université angers

Année : 2010

Propos sur les formules booléennes quantifiées

Thèse d'habilitation à diriger des recherches

Spécialité : Informatique

École Doctorale d'Angers

Présentée et soutenue publiquement

Le 6 juillet 2010

À Angers

Par **Igor Stéphan**

Devant le jury ci-dessous :

Les rapporteurs :	Philippe Besnard,	Directeur de Recherche
	Pierre Marquis,	Professeur d'Université
	Torsten Schaub,	Professeur d'Université
Les examinateurs :	Jin-Kao Hao,	Professeur d'Université
	Pascal Nicolas,	Professeur d'Université

Laboratoire d'Étude et de Recherche en Informatique d'Angers
2, Bd Lavoisier, 49045 Angers Cedex 01

À ma famille, à Alice.

Table des matières

Introduction générale

Pourquoi ce « Propos sur les formules booléennes quantifiées » ? De tous les mots de ce titre, le premier est sans doute celui qui justifie le mieux cet ouvrage. Selon le dictionnaire, un « propos » est, en première définition, « une suite de paroles », et en deuxième, « ce que l'on se propose ». Ce qui suit tient des deux et principalement du second en tant que présentation de ma contribution au domaine des formules booléennes quantifiées. Une autre raison m'attache à « propos » : les formules booléennes quantifiées prennent racine dans la logique *propos*itionnelle.

Alors pourquoi les formules booléennes quantifiées ? Ce formalisme a été pour la première fois défini par Meyer et Stockmeyer [146, 218] dans le domaine de la complexité comme une « hiérarchie » de langages dont le premier intérêt est de mettre en évidence des problèmes « naturels » (c'est-à-dire non artificiellement construits à ce propos) dont les solutions sont calculables de manière non polynomiale en fonction de la taille de ceux-ci ; l'article « The Polynomial-Time Hierarchy » [217] parfait cette « hiérarchie polynomiale » dont les formules booléennes quantifiées sont une représentation. Le formalisme des formules booléennes quantifiées a donc eu comme fées penchées sur son berceau la théorie de la complexité qui en jette ses fondements, mais aussi la logique des prédicats dont il en reprend de très nombreuses propriétés, la logique propositionnelle dont il puise les intuitions et la plupart des méthodes de calcul et enfin la logique du second ordre dont il est une restriction.

De complexité considérée comme déraisonnable puisque exponentielle, il a fallu plus de vingt ans, une éternité dans la science si jeune qu'est l'informatique, et un bond fabuleux de la puissance de calcul des ordinateurs, pour que Cadoli, Giovanardi et Schaerf [46, 47] présentent les premiers résultats pratiques. Voici donc la raison principale de notre intérêt pour les formules booléennes quantifiées : ce formalisme fournit un « raccourci saisissant », comme en peinture, de l'informatique, théorique et pratique, science et technique. De morphologie (la manière d'écrire les formules) et de sémantique (ce que ces formules signifient) élémentaires, les formules booléennes quantifiées offrent un espace de jeu (et de « je » pour cet ouvrage masqué en « nous de majesté ») prodigieux et, lorsque nous avons commencé à nous y intéresser, c'est-à-dire en deux-mille-quatre, relativement neuf.

Il s'est fait jour avec les premières procédures de décision que les formules booléennes quantifiées pouvaient être performantes en représentation, car permettant de décrire de manière compacte, c'est-à-dire dans le meilleur des cas avec un codage de taille logarithmique par rapport à celui dans la logique propositionnelle, si ce n'est encore en calcul pour de nombreux domaines [106] dont la planification [186, 170, 188], la vérification selon une borne de propriétés d'un modèle (« bounded model checking ») [37, 16, 71, 117], la vérification formelle de circuit [24, 142, 33], le test pour l'équivalence d'implémentation partielle [195], la localisation et correction de fautes dans des circuits [1, 206], le calcul de la forme factorisée minimale d'une fonction booléenne [226], la définissabilité dans les circuits programmables FPGA [134], la vérification de propriétés dans les logiques de descrip-

tion [124], le calcul de l'éloignement du centre d'un graphe pour un sommet [149], la compilation de langages pour des raisonnements non-monotones (la circonscription, l'abduction, la logique auto-epistémique et autres logiques modales, la logique des défauts de Reiter et la programmation logique disjonctive au sein du système `QUIP` [77, 76, 75, 173, 82]), les logiques paraconsistantes [35], l'argumentation [34] ou d'autre formalismes $PSPACE$ tels que les diagrammes d'influence possibilistes ($PSPACE$-complet pour le calcul de la stratégie pessimiste optimale [87]) et les jeux finis à deux joueurs [172, 70, 94, 3, 190]. Ce dernier domaine d'application est particulièrement intéressant car il est au delà du « simple » problème de décision et demande le calcul d'(au moins) une solution qui peut être vue comme une stratégie, dans un jeu à deux joueurs, dédiée à un joueur dont l'adversaire est « universel » dans le sens où il faut le vaincre quoi qu'il joue ; cette utilisation des formules booléennes quantifiées place ce formalisme au sein du domaine (même *stricto sensu*) de l'Intelligence Artificielle.

Dans la plus grande part des articles de la littérature, la sémantique des formules booléennes quantifiées est présentée soit sous un angle *ad hoc* à une problématique, soit de façon très sommaire ne permettant pas d'avoir un cadre formel incluant tous les travaux de la communauté. Cette remarque s'applique à nos propres travaux et un effort d'unification est nécessaire pour exprimer dans un unique formalisme l'ensemble de nos contributions. Nous proposons donc une sémantique hautement expressive ainsi qu'un calcul syntaxique mettant en exergue des propriétés qui seront à la base de notre approche du parcours de l'espace de recherche.

Bien après que les bases des formules booléennes quantifiées furent jetées dans la théorie de la complexité, une communauté naissante s'est d'abord intéressée aux aspects calculatoires du problème de décision car celui-ci prolonge le problème de la satisfiabilité d'une formule propositionnelle amenant à une recherche dans un espace combinatoire, précédent thème de recherche pour de nombreux pionniers du jeune domaine.

Nous nous sommes aussi penché sur la spécification de procédures de décision mais en s'affranchissant pour deux raisons de la norme morphologique historique imposant que le fragment traité doit être celui des formules (prénexes) sous forme normale conjonctive : en premier lieu, bien que cette norme soit adaptée particulièrement au problème de satisfiabilité d'une formule propositionnelle, elle est trop dissymétrique par rapport à la sémantique des formules booléennes quantifiées pour en refléter la structure de l'espace de recherche ; en second lieu, bien que ce fragment soit complet dans le sens où toute formule booléenne quantifiée peut se traduire en une formule de même sémantique mais soumise à la norme, ce fragment induit une perte de la structure de la formule ainsi qu'un impact sur l'espace de recherche. En conséquence, nous avons proposé l'abandon pur et simple de la norme et le traitement du langage des formules booléennes quantifiées dans son expressivité maximale selon deux approches diamétralement opposées : d'un côté, la mise au point d'une nouvelle procédure de décision, d'un autre côté, la compilation vers un autre formalisme disposant d'une procédure de décision propre.

La communauté s'est aussi employée à définir et calculer les solutions d'une formule booléenne quantifiée soit pour répondre à un besoin de résultat, soit pour vérifier la per-

tinence du calcul par l'émission d'un certificat. Dans cette optique, nous avons proposé un formalisme plus souple que la simple construction d'une stratégie, dans l'interprétation du jeu à deux joueurs, qui permet entre-autres d'ouvrir un nouvel espace de réflexion : la compilation des formules booléennes quantifiées qui combine l'ensemble des solutions en une forme qui permet de les extraire plus efficacement que dans la formule originale.

Mais la raison de ma rencontre avec ce petit (en apparence) pré carré est évidemment historique : ma filiation académique est celle de la « programmation *logique* » et les équipes de recherche au sein du LERIA (Laboratoire de Recherche en Informatique d'Angers) étudiaient et étudie les *logiques* non-classiques ou non-monotones. Ces dernières, par une transformation qui conserve la complexité, se codent en des problèmes sur les formules booléennes quantifiées, celles-ci se retrouvant comme point commun algorithmique.

Ce sont donc des concepts et des outils venant de la *logique* qui guident mon approche, et ce, dans une démarche complète allant de l'approfondissement des fondements jusqu'à l'implantation soit sous forme de spécifications exécutables en Prolog, langage déclaratif phare de la programmation logique, soit en des outils compétitifs en C++, langage objet impératif.

Cette approche, nous avons déjà eu l'occasion de la pratiquer, sous l'égide du professeur Pascal Nicolas, dans le cadre de travaux sur les langages logiques non-monotones possibilistes [156, 154, 155, 158, 157] et l'application des techniques metaheuristiques à la recherche de solutions pour des langages logiques non-monotones [161, 160, 159, 166, 164, 162, 163, 165, 165]. Le terme « allant » est, dans le cadre de cet ouvrage, plus technique que la simple maîtrise des aspects théoriques et pratiques, il doit être compris dans le sens de « dérivant » : partant de la définition et faisant découler, le plus automatiquement possible (c'est-à-dire à l'aide de programmes), les implantations finales par une exploitation des propriétés algébriques du domaine.

Cet ouvrage reprend six années de recherche sur les formules booléennes quantifiées et s'articule, hormis cette introduction, en sept chapitres. Le premier explore les fondements morphologiques, sémantiques et syntaxiques du domaine. Le deuxième dépeint un panorama des procédures de décision pour le problème de validité des formules booléennes quantifiées en introduisant une classification basée sur un « code génétique » reflétant leurs propriétés ; cela définit une boîte à outils pour le lecteur désireux d'implanter l'une d'elles ou une combinaison nouvelle des outils. Le troisième aborde un nouveau formalisme, celui des « bases littérales », interprétable en termes d'un fragment sémantiquement complet des formules booléennes quantifiées, candidat à être un langage cible pour la compilation de celles-ci. Le quatrième présente la dérivation pas à pas, à partir des propriétés algébriques du domaine des formules booléennes quantifiées, vers une procédure de décision en langage déclaratif puis impératif. Le cinquième traite de la compilation des formules booléennes quantifiées en des programmes logiques non monotones. Le sixième offre une réflexion sur l'intérêt, aussi bien guidé par des considérations d'expressivité que d'efficacité, de ne pas mettre sous forme prénexe et de conserver la bi-implication ainsi que le ou-exclusif dans l'ensemble des connecteurs des fragments en entrée et en interne des procédures de

décision. Le septième et ultime chapitre dresse une conclusion et évoque des travaux en cours et les perspectives qui en découlent.

Une seconde partie sous forme de glossaire définit, pour les non-initiés, des termes utiles à la compréhension du cœur du document sans pour autant faisant partie de notre recherche. Ces termes définis sont en italique dans la première partie et accompagnés entre parenthèses du numéro de page dans le celui-ci. (Cette structure, je l'emprunte au document d'habilitation du professeur Olivier Ridoux [185].)

Enfin, une troisième partie, annexée sous forme de livret indépendant, collecte des compléments techniques et l'ensemble des lemmes qui sont évoqués dans les quatre premiers chapitres mais non détaillés, les démonstrations de ces lemmes et des théorèmes qui sont en exergue ainsi que des définitions et notations complémentaires et des lemmes techniques et leurs démonstrations ; les énoncés et démonstrations des lemmes et théorèmes évoqués dans les chapitres 5 et 6 étant publiés, nous renvoyons le lecteur désireux d'un approfondissement aux références bibliographiques en tête de ces chapitres.

Que le lecteur me pardonne si le premier chapitre est si long et technique : « le Diable se cache dans les détails ». J'ai choisi aussi de conserver, pour les chapitres 3 et 4, une présentation particulièrement rigoureuse et détaillée car elle nous sert et servira dans nos collaborations en vue d'étendre ces résultats. Plus le lecteur progresse dans cet ouvrage, plus le « nous de majesté » devient un « nous de communauté ». Cette évolution est mise en évidence par l'ajout de références bibliographiques en tête des sujets abordés mais aussi par le grain technique apporté, les sujets plus personnels étant plus formalisés et ceux plus partagés, plus « littéraires ». Un lecteur pressé peut ne pas lire les exemples qui ne font qu'illustrer le propos.

Première partie

Propos sur les formules booléennes quantifiées

Chapitre 1

Morphologie, sémantiques et syntaxe des formules booléennes quantifiées

Il y eut un soir, il y eut un matin : premier jour.

[212] I. Stéphan. Sémantique et calcul syntaxique pour les formules booléennes quantifiées. Dans *Quatrièmes Journées Francophones de Programmation par Contraintes*, 2009.

Sommaire

Nous présentons le formalisme des formules booléennes quantifiées dans un cadre formel rigoureux. Cette formalisation est particulièrement importante car elle établit un langage commun dans lequel les définitions sont sans ambiguïté et les lemmes et théorèmes indiscutablement démontrés. Comme pour tout langage il nous faut en définir son alphabet, son vocabulaire (ici élémentaire) ainsi que sa syntaxe que nous appelons « morphologie » et sa sémantique qui permet d'attribuer à chaque formule construite sur le langage un sens. Nous préservons la dichotomie chère au domaine de la logique entre la sémantique qui attribue à toute formule une valeur de vérité selon un processus directement lié à la sémantique des éléments du langage et la syntaxe qui ne s'intéresse qu'aux propriétés structurelles de cette sémantique et basée sur un simple jeu de « copier-coller » sans faire appel à la sémantique elle-même. Le titre de ce chapitre met au pluriel le mot « sémantique » : la sémantique classique, présente dans la plus grande part de la littérature, n'est que « décisionnelle », elle ne permet pas de calculer les solutions ou de traiter des symboles propositionnels libres, nous l'étendons donc à une sémantique, uniquement pour des formules booléennes quantifiées dites « prénexes », qui permet de vérifier ou de construire des solutions. Nous explorons particulièrement les relations d'équivalence qui partitionnent les formules booléennes quantifiées selon ces sémantiques. Nous présentons un système syntaxique, démontré correct et complet par rapport aux sémantiques qui nous sert de fondement à l'approche algorithmique du chapitre 4. Enfin, nous présentons l'algorithmique élémentaire des formules booléennes quantifiées directement issue de la sémantique.

1.1 Morphologie des formules booléennes quantifiées

L'alphabet des *formules booléennes quantifiées* emprunte celui de la *logique propositionnelle* et y ajoute les deux quantificateurs de la *logique des prédicats.*

Définition 1 (alphabet des formules booléennes quantifiées) *L'alphabet des formules booléennes quantifiées est constitué :*

- *d'un ensemble (dénombrable) de* symboles propositionnels $\mathcal{SP}$,
- *des* connecteurs logiques *:* $\neg$ *(négation),* $\wedge$ *(et, conjonction),* $\vee$ *(ou, disjonction),* $\rightarrow$ *(implication),* $\leftrightarrow$ *(bi-implication) et* $\oplus$ *(ou-exclusif),*
- *deux* constantes logiques *:*$\top$ *(top, « ce qui est toujours vrai ») et* $\bot$ *(bottom, « ce qui est toujours faux »),*
- *de deux quantificateurs :* $\forall$ *(« quelque soit », quantificateur de* polarité *universelle) et* $\exists$ *(« il existe », quantificateur de polarité existentielle),*
- *d'une parenthèse ouvrante « (» et d'une parenthèse fermante «) ».*

L'ensemble **QBF** des formules booléennes quantifiées (ou « QBF » pour « Quantified Boolean Formulae ») est défini inductivement sur son alphabet[1] ajoutant aux formules

[1]Cette définition peut être plus formelle en considérant que l'ensemble **QBF** est la clôture de l'ensemble $\mathcal{SP} \cup \{\top, \bot\}$ selon les fonctions définies ainsi ($A, B \in$ **QBF** et $x \in \mathcal{SP}$) :$C_\neg(A) = \neg A$, $C_\wedge(A, B) = (A \wedge B)$, $C_\vee(A, B) = (A \vee B)$, $C_\rightarrow(A, B) = (A \rightarrow B)$, $C_\leftrightarrow(A, B) = (A \leftrightarrow B)$, $C_\oplus(A, B) = (A \oplus B)$, $C_\forall(x, A) = (\forall x\ A)$ et $C_\exists(x, A) = (\exists x\ A)$

propositionnelles la possibilité de quantifier les symboles propositionnels et non les variables comme en logique des prédicats.

Définition 2 (formule booléenne quantifiée) *L'ensemble* **QBF** *des formules booléennes quantifiées est défini inductivement par :*

- *tout élément de* $\mathcal{SP} \cup \{\bot, \top\}$ *est un élément de* **QBF** *;*
- *si* G *est un élément de* **QBF** *alors* $\neg G$ *est un élément de* **QBF** *;*
- *si* G *et* H *sont des éléments de* **QBF** *alors* $(G \wedge H)$, $(G \vee H)$, $(G \rightarrow H)$, $(G \leftrightarrow H)$ *et* $(G \oplus H)$ *sont des éléments de* **QBF** *;*
- *si* G *est un élément de* **QBF** *et* x *est un élément de* $\mathcal{SP}$ *alors* $(\forall x\ G)$ *et* $(\exists x\ G)$ *sont des éléments de* **QBF**.

Exemple 1 (formule booléenne quantifiée) *Les expressions*

$$(\forall a\ (\exists d\ (\forall e\ (\exists f\ \neg\neg\neg((e \rightarrow d) \rightarrow \neg(f \rightarrow a))))))$$

et

$$(\forall a\ (\exists d\ (\forall e\ \neg\neg\neg((e \rightarrow d) \rightarrow (\forall f\ \neg(f \rightarrow a))))))$$

sont des QBF.

Nous définissons inductivement l'ensemble $sf(F)$ des sous-formules de F, une QBF, ainsi que l'ensemble étendue $sfe(F)$ des sous-formules. La notion d'« ensemble étendu des sous-formules » intervient dans la définition du calcul syntaxique (cf. § 1.4).

Définition 3 (sous-formule d'une QBF) *Soit* F *une QBF, l'ensemble des sous-formules de* F, $sf(F)$, *est défini inductivement ainsi :*

- $sf(F) = \{F\}$ *si* $F = x \in \mathcal{SP}$ *ou* $F \in \{\top, \bot\}$ *;*
- $sf(F) = \{F\} \cup sf(G)$ *si* $F = \neg G$ *et* $G \in$ **QBF** *;*
- $sf(F) = \{F\} \cup sf(G) \cup sf(H)$ *si* $F = (G \circ H)$, $\circ \in \{\wedge, \vee, \rightarrow, \leftrightarrow, \oplus\}$ *et* $G, H \in$ **QBF** *;*
- $sf(F) = \{F\} \cup sf(G)$ *si* $F = (qx\ G)$, $q \in \{\forall, \exists\}$, $G \in$ **QBF** *et* $x \in \mathcal{SP}$.

La fonction $\overline{(.)}$ *associe à une QBF son* complémentaire *ou* conjugué *: si* $F = \neg G$ *alors* $\overline{F} = G$ *sinon* $\overline{F} = \neg F$.

L'ensemble des sous-formules et de leurs complémentaires $sfc(F)$, *est défini inductivement ainsi :*

- $sfc(F) = \{F, \overline{F}\}$ *si* $F = x \in \mathcal{SP}$ *;*
- $sfc(F) = \{\top, \overline{\top}\}$ *si* $F \in \{\top, \bot\}$ *;*
- $sfc(F) = \{F\} \cup sfc(G)$ *si* $F = \neg G$ *et* $G \in$ **QBF** *;*
- $sfc(F) = \{F, \overline{F}\} \cup sfc(G) \cup sfc(H)$ *si* $F = (G \circ H)$, $\circ \in \{\wedge, \vee, \rightarrow, \leftrightarrow, \oplus\}$ *et* $G, H \in$ **QBF** *;*
- $sfc(F) = \{F, \overline{F}\} \cup sfc(G)$ *si* $F = (qx\ G)$, $q \in \{\forall, \exists\}$, $G \in$ **QBF** *et* $x \in \mathcal{SP}$.

L'ensemble étendu des sous-formules de F, $sfe(F)$, *est défini par*

$$sfe(F) = sfc(F) \cup \{\top, \overline{\top}\}.$$

L'ensemble étendu des sous formules faisant intervenir les complémentaires élimine de fait les successions de négations. Dans cet ensemble n'apparaissent ni $\bot$, ni $\overline{\top}$: la sémantique (cf. § 1.2) démontre que $\top$ et $\overline{\bot}$ sont deux écritures d'une même réalité sémantique et qu'il en est de même pour $\bot$ et $\overline{\top}$.

Exemple 2 (sous-formule d'une QBF) *En continuant l'exemple 1, et en posant*

$$\xi = (\forall a\ (\exists d\ (\forall e\ (\exists f\ \neg\neg\neg((e{\rightarrow}d){\rightarrow}\neg(f{\rightarrow}a))))))$$

alors

$$\begin{array}{l} sfe(\xi) = \\ \{\quad \xi, \neg\xi, (\exists d\ (\forall e\ (\exists f\ \neg\neg\neg((e{\rightarrow}d){\rightarrow}\neg(f{\rightarrow}a))))), \\ \quad \neg(\exists d\ (\forall e\ (\exists f\ \neg\neg\neg((e{\rightarrow}d){\rightarrow}\neg(f{\rightarrow}a))))), (\forall e\ (\exists f\ \neg\neg\neg((e{\rightarrow}d){\rightarrow}\neg(f{\rightarrow}a)))) \\ \quad \neg(\forall e\ (\exists f\ \neg\neg\neg((e{\rightarrow}d){\rightarrow}\neg(f{\rightarrow}a)))), (\exists f\ \neg\neg\neg((e{\rightarrow}d){\rightarrow}\neg(f{\rightarrow}a))), \\ \quad \neg(\exists f\ \neg\neg\neg((e{\rightarrow}d){\rightarrow}\neg(f{\rightarrow}a))), \neg\neg\neg((e{\rightarrow}d){\rightarrow}\neg(f{\rightarrow}a)), \\ \quad \neg\neg((e{\rightarrow}d){\rightarrow}\neg(f{\rightarrow}a)), \neg((e{\rightarrow}d){\rightarrow}\neg(f{\rightarrow}a)), ((e{\rightarrow}d){\rightarrow}\neg(f{\rightarrow}a)), \\ \quad (e{\rightarrow}d), \neg(e{\rightarrow}d), (f{\rightarrow}a), \neg(f{\rightarrow}a), e, \neg e, d, \neg d, f, \neg f, a, \neg a, \top, \neg\top\} \end{array}$$

Comme pour la logique des prédicats, un symbole propositionnel peut être soit *lié* à un quantificateur dans une formule, soit *libre* voire les deux à la fois, mais pour des occurrences différentes.

Définition 4 (symbole propositionnel lié ou libre et QBF close ou polie) *Un symbole propositionnel x qui apparaît dans une QBF en dehors de $\forall x$ et $\exists x$ est une* occurrence *du symbole dans la QBF. Dans la QBF $(qx\ F)$, q un quantificateur quelconque, la QBF F est la portée du quantificateur q associé au symbole propositionnel x. Une occurrence d'un symbole propositionnel x est liée à un quantificateur de portée F si le symbole associé à ce quantificateur est x, et cette occurrence n'est liée à aucun autre quantificateur d'une sous-formule de F ; si le quantificateur est existentiel (resp. universel) alors l'occurrence du symbole est existentiellement quantifiée (resp. universellement quantifiée) ; l'*ensemble des symboles propositionnels liés[2] *d'une QBF F est noté $\mathcal{SPB}(F)$. Une occurrence de symbole propositionnel qui n'est lié à aucun quantificateur est* libre *; l'*ensemble des symbole propositionnels libres *d'une QBF F est noté $\mathcal{SPL}(F)$. Une QBF est* polie *si tous les symboles propositionnels associés à des quantificateurs et ceux qui sont libres sont différents. Une QBF F est* close *si $\mathcal{SPL}(F) = \emptyset$.*

*L'*ensemble des symboles propositionnels existentiellement quantifiés, $\mathcal{SP}_{\exists}(F)$ *(resp.* universellement quantifiés, $\mathcal{SP}_{\forall}(F)$*) d'une QBF F est la restriction au collectage des symboles propositionnels associés à un quantificateur existentiel (resp. universel). L'*ensemble des symboles propositionnels *d'une QBF F, $\mathcal{SP}(F)$, est l'union des ensembles $\mathcal{SPB}(F)$ et $\mathcal{SPL}(F)$.*

[2] « lié » se traduit en anglais par « bound »

Nous définissons le fragment des QBF prénexes qui est de premier intérêt dans la définition de la sémantique grâce à l'ensemble **PROP** des formules propositionnelles (qui peut être défini par induction sur les trois premiers items de la définition 2 en remplaçant **QBF** par **PROP**).

Définition 5 (formule booléenne quantifiée prénexe) *Une QBF est prénexe si elle est constituée, en préfixe, d'un* lieur*, une chaîne de caractères composée d'une suite de quantificateurs associés chacun à un symbole propositionnel, notée ε si elle est vide, et d'une* matrice*, une formule propositionnelle. L'ensemble $\Sigma_k - \mathbf{QBF}$ (resp. $\Pi_k - \mathbf{QBF}$) est l'ensemble des QBF prénexes closes telles que le lieur est une alternance sur k ensembles homogènes de quantificateurs dont le plus à gauche est un quantificateur existentiel (resp. universel). La restriction d'un lieur Q à un ensemble de symboles propositionnels S est noté $Q|_S$.*

Une QBF prénexe peut être réécrite en omettant les parenthèses du lieur.

Exemple 3 (formule booléenne quantifiée prénexe) *En continuant l'exemple 1, la QBF $\xi = \forall a \exists d \forall e \exists f \mu$ avec $\mu = \neg\neg\neg((e \rightarrow d) \rightarrow \neg(f \rightarrow a))$ est une QBF prénexe, $\forall a \exists d \forall e \exists f$ étant le lieur et μ la matrice. Cette QBF signifie la même chose que la QBF $(\forall a\ (\exists d\ (\forall e\ (\exists f\ \mu))))$ qui explicite toutes les parenthèses ($\xi \in \Pi_4 - \mathbf{QBF}$). Par contre, la QBF*

$$(\forall a\ (\exists d\ (\forall e\ \neg\neg\neg((e \rightarrow d) \rightarrow (\forall f\ \neg(f \rightarrow a))))))$$

n'est pas prénexe.

A toute QBF polie est associée une relation d'ordre partiel sur les symboles propositionnels qui devient totale si la QBF est prénexe. Nous considérons par la suite que toute QBF prénexe est polie.

Définition 6 (relation d'ordre associée à une QBF) *Soit F, une QBF polie, la relation $\prec$ est définie sur $\mathcal{SPB}(F) \times \mathcal{SPB}(F)$. Soient $x, y \in \mathcal{SPB}(F)$, q, q' deux quantificateurs et $(qx\ G), (q'y\ H)$ des sous-formules de F : $x \prec y$ si $(q'y\ H)$ est une sous-formule de $(qx\ G)$. Dans une QBF prénexe, le* quantificateur le plus externe *est le quantificateur le plus petit selon la relation d'ordre $\prec$ et le* quantificateur le plus interne *le plus grand.*

Exemple 4 (relation d'ordre associée à une QBF) *La relation d'ordre induite par la QBF polie $(\forall a\ ((\exists d\ (a \rightarrow d)) \wedge (\forall e\ (e \leftrightarrow a))))$ est $\{a \prec d, a \prec e\}$.*

La notion de « littéral » de la *logique propositionnelle* et de la *logique des prédicats* s'étend naturellement aux QBF.

Définition 7 (littéral) *Un* littéral *est un symbole propositionnel (*littéral positif*) ou sa négation (*littéral négatif*). La fonction* $|.|$ *associe à un littéral son symbole propositionnel sous-jacent. Les littéraux* l *et* $\overline{l}$ *sont des* littéraux complémentaires *ou encore des* littéraux conjugués*. Un littéral est* universellement quantifié *(resp.* existentiellement quantifié*) si le symbole propositionnel sous-jacent est lui-même* universellement quantifié *(resp.* existentiellement quantifié*). Un littéral positif est de* polarité *positive et un littéral négatif est de polarité négative.*

Cette notion de « littéral » est en lien avec la notion de « complémentaire » de la définition 3 : si l est un littéral alors $\overline{l}$ est aussi un littéral et $\overline{\overline{l}} = l$.

Les formes normales classiques de la logique propositionnelle et de la logique des prédicats qui sont la *forme normale négative*, la *forme normale conjonctive* et la *forme normale disjonctive* s'étendent naturellement aux QBF.

Définition 8 (formes normales négative, conjonctive et disjonctive) *Une QBF est sous* forme normale négative *(ou* FNN*) si c'est une QBF restreinte sur les connecteurs à la conjonction, la disjonction et à la négation ne formant que des littéraux. Une QBF est sous* forme normale conjonctive *(ou* FNC*) si c'est une QBF sous FNN et formant une conjonction de disjonctions de littéraux ; une QBF est sous* forme normale disjonctive *(ou* FND*) si c'est une QBF sous FNN et formant une disjonction de conjonctions de littéraux. Si la QBF est sous forme prénexe alors elle est sous FNN (resp. FNC, FND) si sa matrice est sous* forme normale négative *(resp.* forme normale conjonctive*,* forme normale disjonctive*).*

La définition ci-dessus ne contraint pas la QBF a être prénexe pour être sous forme normale négative (ou conjonctive ou disjonctive).

Exemple 5 (forme normale disjonctive) *La QBF* $(\forall a\ (\exists d\ ((\forall e\ ((a \wedge \neg e) \wedge \neg d)) \vee (a \wedge d))))$ *est sous forme normale disjonctive (et donc négative).*

La *fonction de substitution* (ou plus simplement *substitution*) établit la manière de manipuler les formules.

Définition 9 (fonction de substitution) *Une* fonction de substitution *est une fonction de l'ensemble des symboles propositionnels* $\mathcal{SP}$ *dans l'ensemble des QBF. L'ensemble support d'une fonction de substitution* σ *est l'ensemble des symboles propositionnels* x *tels que* $\sigma(x) \neq x$*. Soit* $\{x_1, \ldots, x_n\}$ *l'ensemble support d'une fonction de substitution* σ *alors la fonction* σ *est notée* $[x_1 \leftarrow \sigma(x_1), \ldots, x_n \leftarrow \sigma(x_n)]$*. Par abus de langage la fonction de substitution (ou* substitution*) dénote aussi son extension à toute QBF (*$G_1, \ldots, G_n$ *un ensemble de QBF et* $\sigma' = [x_1 \leftarrow G_1, \ldots, x_{i-1} \leftarrow G_{i-1}, x_{i+1} \leftarrow G_{i+1}, \ldots, x_n \leftarrow G_n]$*) :*

$$
\begin{array}{rcll}
\sigma(\bot) & = & \bot & \\
\sigma(\top) & = & \top & \\
\sigma(x) & = & x & \textit{si } \sigma = [x_1 \leftarrow G_1, \ldots, x_n \leftarrow G_n] \\
 & & & \textit{et } x \in \mathcal{SP}, x \notin \{x_1, \ldots, x_n\} \\
\sigma(x_i) & = & G_i & \textit{si } \sigma = [x_1 \leftarrow G_1, \ldots, x_n \leftarrow G_n] \textit{ et } x_i \in \{x_1, \ldots, x_n\} \\
\sigma(\neg G) & = & \neg\sigma(G) & \textit{si } G \in \mathbf{QBF} \\
\sigma((G \circ H)) & = & (\sigma(G) \circ \sigma(H)) & \textit{si } G, H \in \mathbf{QBF} \textit{ et } \circ \in \{\wedge, \vee, \rightarrow, \leftrightarrow, \oplus\} \\
\sigma((q x_i\ G)) & = & (q x_i\ \sigma'(G)) & \textit{si } G \in \mathbf{QBF} \textit{ et } q \in \{\forall, \exists\} \\
\sigma((q x\ G)) & = & (q x\ \sigma(G)) & \textit{si } G \in \mathbf{QBF}, q \in \{\forall, \exists\} \textit{ et } x \notin \{x_1, \ldots, x_n\}
\end{array}
$$

La formule propositionnelle $\bigwedge_{1 \leq i \leq n} (x_i \leftrightarrow G_i)$ *est la* formule associée à la substitution $[x_1 \leftarrow G_1, \ldots, x_n \leftarrow G_n]$.

L'ensemble des substitutions avec pour codomaine l'union de l'ensemble des littéraux et $\{\top, \bot\}$ *est noté* $\mathcal{R}$.

Comme pour la logique des prédicats, le fait que les symboles propositionnels soient nommés et non simplement manipulés comme des références rend nécessaire l'introduction de la notion de « QBF librement substituable pour un symbole propositionnel dans une QBF » pour éviter des aberrations sémantiques (cf. § 1.2).

Définition 10 (librement substituable) *Une QBF F est* librement substituable pour un symbole propositionnel x dans une QBF G *si, pour tout symbole propositionnel y libre de F, x n'a pas d'occurrence libre dans la QBF G sous la portée d'un quantificateur portant sur y.*

1.2 Sémantiques des formules booléennes quantifiées

À chaque QBF nous attribuons une *valeur de vérité* qui en sera sa *sémantique*. Par extension, la *sémantique* est une fonction qui associe à une QBF cette valeur de vérité.

Définition 11 (valeur de vérite) *L'ensemble* $\mathbf{BOOL} = \{\mathbf{vrai}, \mathbf{faux}\}$ *des* booléens *est l'ensemble des* valeurs de vérité. *Une* fonction booléenne *est une fonction de* $\mathbf{BOOL}^n$, $n \geq 0$, *dans* **BOOL**.

La littérature présente plusieurs définitions morphologiques des QBF : de la définition la plus restreinte des QBF polies prénexes sous FNC à la définition des QBF dans la plus large acception. En conséquence il existe plusieurs niveaux de sémantique. Nous présentons dans un premier temps la sémantique des QBF dans sa forme « décisionnelle » selon une *valuation propositionnelle* : le seul objectif de la sémantique est de *décider* si « oui » ou « non » la QBF est évaluée à **vrai**. Une valuation propositionnelle établit, pour le symbole propositionnel, le lien entre la syntaxe et la sémantique.

Définition 12 (valuation propositionnelle) *Une* valuation *est une fonction qui associe à chaque symbole propositionnel une valeur de vérité ; un symbole propositionnel est alors* valué *à une valeur de vérité. L'ensemble des valuations propositionnelles est noté* **VAL_PROP**. *Nous notons* $v[y := b](x) \begin{cases} = & b & si\ y = x \\ = & v(x) & sinon. \end{cases}$

Si la QBF est considérée comme codant les règles d'un jeu fini à deux joueurs et les conditions de victoire du joueur « existentiel » face à un joueur « universel », alors la sémantique de la QBF est si « oui » ou « non » le joueur existentiel dispose d'(au moins) une stratégie pour vaincre quoi que le joueur universel joue. Mais une définition de la sémantique sous une forme uniquement décisionnelle, comme celle des quantificateurs en logique des prédicats, pour une QBF quelconque ne renseigne en rien sur le comportement des existentielles, c'est-à-dire sur les mouvements du joueur existentiel pour le mener à la victoire : tout ce que sait le joueur existentiel est si « oui » ou « non » il existe une telle stratégie. Nous proposons donc une autre définition de la sémantique des QBF mais restreinte au fragment prénexe (poli) pour pallier au manque de la sémantique décisionnelle.

1.2.1 Sémantique décisionnelle des QBF

Les définitions suivantes pour la sémantique décisionnelle des QBF reprennent l'organisation de la définition de la *sémantique de la logique des prédicats* proposée dans [86] : une définition de la sémantique des connecteurs et constantes logiques au niveau propositionnel identique à celle de la *sémantique de la logique propositionnelle*, une définition de la sémantique des connecteurs, constantes et quantificateurs au niveau QBF comme une restriction de celle de la sémantique de la logique des prédicats, une définition de la sémantique des QBF comme extension aux formules de la sémantique des connecteurs, constantes et quantificateurs.

À chaque connecteur logique est associée une fonction à valeur dans **BOOL** qui en définit sa sémantique au niveau propositionnel.

Définition 13 (sémantique propositionnelle des connecteurs logiques)

$$i_\top, i_\bot :\rightarrow \mathbf{BOOL}$$

$$i_\top = \mathbf{vrai} \qquad i_\bot = \mathbf{faux}$$

$$i_\neg : \mathbf{BOOL} \rightarrow \mathbf{BOOL}$$

x	$i_\neg(x)$
vrai	**faux**
faux	**vrai**

$$i_\wedge, i_\vee, i_\rightarrow, i_\leftrightarrow, i_\oplus : \mathbf{BOOL} \times \mathbf{BOOL} \rightarrow \mathbf{BOOL}$$

x	y	$i_\wedge(x,y)$	$i_\vee(x,y)$	$i_\rightarrow(x,y)$	$i_\leftrightarrow(x,y)$	$i_\oplus(x,y)$
vrai	**vrai**	**vrai**	**vrai**	**vrai**	**vrai**	**faux**
vrai	**faux**	**faux**	**vrai**	**faux**	**faux**	**vrai**
faux	**vrai**	**faux**	**vrai**	**vrai**	**faux**	**vrai**
faux	**faux**	**faux**	**faux**	**vrai**	**vrai**	**faux**

Uniquement pour $\top$ *et* $\bot$ *sont définies deux fonctions* $i : \{\top, \bot\} \rightarrow \mathbf{BOOL}$ *définie par* $i(\top) = \mathbf{vrai}$ *et* $i(\bot) = \mathbf{faux}$*, et* $i^{-1} : \mathbf{BOOL} \rightarrow \{\top, \bot\}$ *définie par* $i^{-1}(\mathbf{vrai}) = \top$ *et* $i^{-1}(\mathbf{faux}) = \bot$*.*

Nous définissons la sémantique décisionnelle des connecteurs et constantes logiques ainsi que des quantificateurs. La sémantique décisionnelle prend en paramètre une valuation propositionnelle qui associe à chaque symbole propositionnel libre une valeur de vérité ; la valuation propositionnelle étant une fonction, à tout symbole propositionnel est associée une valeur de vérité.

Définition 14 (sémantique décisionnelle des connecteurs et quantificateurs)

$$I_\top, I_\bot : \mathbf{VAL\_PROP} \rightarrow \mathbf{BOOL}$$

$$I_\bot(v) = i_\bot \quad et \quad I_\top(v) = i_\top$$

$$I_\neg : (\mathbf{VAL\_PROP} \rightarrow \mathbf{BOOL}) \times (\mathbf{VAL\_PROP}) \rightarrow \mathbf{BOOL}$$

$$I_\neg(f)(v) = i_\neg(f(v))$$

$$I_\circ : (\mathbf{VAL\_PROP} \rightarrow \mathbf{BOOL})^2 \times (\mathbf{VAL\_PROP}) \rightarrow \mathbf{BOOL}$$

$$I_\circ(f,g)(v) = i_\circ(f(v), g(v)) \text{ pour tout connecteur } \circ \in \{\wedge, \vee, \rightarrow, \leftrightarrow, \oplus\}$$

$$I^x_\exists, I^x_\forall : (\mathbf{VAL\_PROP} \rightarrow \mathbf{BOOL}) \times (\mathbf{VAL\_PROP}) \rightarrow \mathbf{BOOL}$$

$$I^x_\exists(f)(v) = i_\vee(f(v[x := \mathbf{vrai}]), f(v[x := \mathbf{faux}]))$$
$$I^x_\forall(f)(v) = i_\wedge(f(v[x := \mathbf{vrai}]), f(v[x := \mathbf{faux}]))$$

*Ces deux dernières définitions forment l'*élimination des quantificateurs*.*

Nous sommes en mesure de définir la sémantique décisionnelle des QBF comme une extension de la sémantique décisionnelle des connecteurs, constantes et quantificateurs.

Définition 15 (sémantique décisionnelle des QBF) *La sémantique décisionnelle d'une QBF F est une fonction*

$$I^*(F) : \mathbf{VAL\_PROP} \rightarrow \mathbf{BOOL}$$

définie inductivement par :
- $I^*(F)(v) = I_\bot(v)$ *si* $F = \bot$;
- $I^*(F)(v) = I_\top(v)$ *si* $F = \top$;
- $I^*(F)(v) = v(x)$ *si* $F = x$ *et* $x \in \mathcal{SP}$;
- $I^*(F)(v) = I_\circ(I^*(G), I^*(H))(v)$ *si* $F = (G \circ H)$, $G, H \in \mathbf{QBF}$ *et* $\circ \in \{\wedge, \vee, \rightarrow, \leftrightarrow, \oplus\}$;
- $I^*(F)(v) = I_\neg(I^*(G))(v)$ *si* $F = \neg G$ *et* $G \in \mathbf{QBF}$;
- $I^*(F)(v) = I^x_\exists(I^*(G))(v)$ *si* $F = (\exists x\ G)$, $G \in \mathbf{QBF}$ *et* $x \in \mathcal{SP}$;
- $I^*(F)(v) = I^x_\forall(I^*(G))(v)$ *si* $F = (\forall x\ G)$, $G \in \mathbf{QBF}$ *et* $x \in \mathcal{SP}$.

Exemple 6 (sémantique décisionnelle des QBF) *En continuant l'exemple 1, soit la QBF*

$$\psi = (\forall a\ (\exists d\ (\forall e\ \neg\neg\neg((e \rightarrow d) \rightarrow (\forall f\ \neg(f \rightarrow a))))))$$

Alors, pour une valuation propositionnelle quelconque v,

$$\begin{array}{ll} & I^*(\psi)(v) \\ = & I^a_\forall(I^*((\exists d\ (\forall e\ \neg\neg\neg((e \rightarrow d) \rightarrow (\forall f\ \neg(f \rightarrow a)))))))(v) \\ = & i_\wedge(I^*((\exists d\ (\forall e\ \neg\neg\neg((e \rightarrow d) \rightarrow (\forall f\ \neg(f \rightarrow a))))))(v[a := \mathbf{vrai}]), \\ & I^*((\exists d\ (\forall e\ \neg\neg\neg((e \rightarrow d) \rightarrow (\forall f\ \neg(f \rightarrow a))))))(v[a := \mathbf{faux}])) \end{array}$$

Détaillons, en posant $v_a = v[a := \mathbf{vrai}]$,

$$\begin{array}{ll} & I^*((\exists d\ (\forall e\ \neg\neg\neg((e \rightarrow d) \rightarrow (\forall f\ \neg(f \rightarrow a))))))(v_a) \\ = & I^d_\exists(I^*((\forall e\ \neg\neg\neg((e \rightarrow d) \rightarrow (\forall f\ \neg(f \rightarrow a))))))(v_a) \\ = & i_\vee(I^*((\forall e\ \neg\neg\neg((e \rightarrow d) \rightarrow (\forall f\ \neg(f \rightarrow a)))))(v_a[d := \mathbf{vrai}]), \\ & I^*((\forall e\ \neg\neg\neg((e \rightarrow d) \rightarrow (\forall f\ \neg(f \rightarrow a)))))(v_a[d := \mathbf{faux}])) \end{array}$$

et, en posant $v_{\overline{d}} = v_a[d := \mathbf{faux}]$,

$$\begin{array}{ll} & I^*((\forall e\ \neg\neg\neg((e \rightarrow d) \rightarrow (\forall f\ \neg(f \rightarrow a)))))(v_{\overline{d}}) \\ = & I^e_\forall(I^*(\neg\neg\neg((e \rightarrow d) \rightarrow (\forall f\ \neg(f \rightarrow a)))))(v_{\overline{d}}) \\ = & i_\wedge(I^*(\neg\neg\neg((e \rightarrow d) \rightarrow (\forall f\ \neg(f \rightarrow a))))(v_{\overline{d}}[e := \mathbf{vrai}]), \\ & I^*(\neg\neg\neg((e \rightarrow d) \rightarrow (\forall f\ \neg(f \rightarrow a))))(v_{\overline{d}}[e := \mathbf{faux}])) \end{array}$$

puis, en posant $v_e = v_{\overline{d}}[e := \mathbf{vrai}]$,

$$\begin{array}{ll} & I^*(\neg\neg\neg((e \rightarrow d) \rightarrow (\forall f\ \neg(f \rightarrow a))))(v_e) \\ = & i_\neg(I^*(\neg\neg((e \rightarrow d) \rightarrow (\forall f\ \neg(f \rightarrow a))))(v_e)) \\ = & i_\neg(i_\neg(I^*(\neg((e \rightarrow d) \rightarrow (\forall f\ \neg(f \rightarrow a))))(v_e))) \\ = & I^*(\neg((e \rightarrow d) \rightarrow (\forall f\ \neg(f \rightarrow a))))(v_e) \\ = & i_\neg(I^*(((e \rightarrow d) \rightarrow (\forall f\ \neg(f \rightarrow a))))(v_e)) \\ = & i_\neg(i_\rightarrow(I^*((e \rightarrow d))(v_e), I^*((\forall f\ \neg(f \rightarrow a)))(v_e))) \end{array}$$

enfin

$$\begin{array}{ll} & I^*((e{\rightarrow}d))(v_e) \\ = & i_{\rightarrow}(I^*(e)(v_{\overline{d}}[e := \mathbf{vrai}]), I^*(d)(v_{\overline{d}}[e := \mathbf{vrai}])) \\ = & i_{\rightarrow}(v_{\overline{d}}[e := \mathbf{vrai}](e), v_{\overline{d}}[e := \mathbf{vrai}](d)) \\ = & i_{\rightarrow}(\mathbf{vrai}, v_{\overline{d}}(d)) = i_{\rightarrow}(\mathbf{vrai}, \mathbf{faux}) = \mathbf{faux} \end{array}$$

et sans plus de détails $I^*(\psi)(v) = \mathbf{vrai}$.

Nous terminons par la définition classique de validité d'une QBF et du problème de décision associé[3].

Définition 16 (Validité et Problème de décision) *Une QBF F est* valide *si, pour toute valuation propositionnelle v, $I^*(F)(v) = \mathbf{vrai}$. Le* problème de validité *d'une QBF est un problème de décision énoncé ainsi :*

- Instance *: Une QBF F.*
- Question *: La QBF F est-elle valide ?*

Décider de la validité d'une QBF est le thème central d'une grande partie de la littérature sur les QBF et des chapitres 2 et 4.

La validité est sensible à l'ordre des quantificateurs : la QBF $(\forall d\ (\exists a\ (\exists c\ ((c{\vee}a){\leftrightarrow}d))))$ est valide tandis que la QBF $(\exists a\ (\exists c\ (\forall d\ ((c{\vee}a){\leftrightarrow}d))))$ ne l'est pas. En l'absence totale de quantificateur, la sémantique des QBF est la même que la sémantique de la logique propositionnelle (cf. Démonstrations, lemme 1) ; décider de la satisfiabilité d'une formule propositionnelle revient donc à décider de la validité de la clôture existentielle d'une QBF prénexe, ce qui correspond à quantifier existentiellement le plus à l'extérieur les symboles propositionnels libres. Décider de la validité d'une instance du fragment $\Sigma_k - \mathbf{QBF}$ (resp. $\Pi_k - \mathbf{QBF}$) est Σ_k^p*-complet* (resp. Π_k^p*-complet*) ; en vertu des propriétés des quantificateurs vis-à-vis de la négation, les classes Σ_k^p et Π_k^p sont *complémentaires*. La classe de complexité Σ_{k+1}^p-complet, pour $k > 0$ recouvre (sous des hypothèses classiques de non effondrement de la hiérarchie polynomiale [172, 88]) des instances plus difficiles à résoudre que celles de complexité Σ_k^p-complet. La *hiérarchie polynomiale* (qui est l'union infinie des classes Σ_k^p, $k \geq 0$) est incluse dans $PSPACE$ ce qui justifie les algorithmes basés sur le *retour-arrière* (cf. § 1.5.1).

Exemple 7 (Validité et Problème de décision) *En continuant les exemples 6 et 3, la QBF ψ est valide et il en est de même de la QBF $\xi = \forall a \exists d \forall e \exists f \mu$ avec*

$$\mu = \neg\neg\neg((e{\rightarrow}d){\rightarrow}\neg(f{\rightarrow}a))$$

ce qui peut être aisément déduit de la table de vérité puisque la QBF est prénexe (polie et close).

[3]Ce problème est parfois appelé $QSAT$ dans la littérature par extension du problème SAT

$v(a)$	$v(d)$	$v(e)$	$v(f)$	$I^*(\mu)(v)$
vrai	**vrai**	**vrai**	**vrai**	**vrai**
vrai	**vrai**	**vrai**	**faux**	**vrai**
vrai	**vrai**	**faux**	**vrai**	**vrai**
vrai	**vrai**	**faux**	**faux**	**vrai**
vrai	**faux**	**vrai**	**vrai**	**faux**
vrai	**faux**	**vrai**	**faux**	**faux**
vrai	**faux**	**faux**	**vrai**	**vrai**
vrai	**faux**	**faux**	**faux**	**vrai**
faux	**vrai**	**vrai**	**vrai**	**faux**
faux	**vrai**	**vrai**	**faux**	**vrai**
faux	**vrai**	**faux**	**vrai**	**faux**
faux	**vrai**	**faux**	**faux**	**vrai**
faux	**faux**	**vrai**	**vrai**	**faux**
faux	**faux**	**vrai**	**faux**	**faux**
faux	**faux**	**faux**	**vrai**	**faux**
faux	**faux**	**faux**	**faux**	**vrai**

Les quatre valuations v_1, v_2, v_3 et v_4 telles que
- $v_1(a) = \mathbf{vrai}$, $v_1(d) = \mathbf{vrai}$, $v_1(e) = \mathbf{vrai}$ *et* $v_1(f) = \mathbf{vrai}$
- $v_2(a) = \mathbf{vrai}$, $v_2(d) = \mathbf{vrai}$, $v_2(e) = \mathbf{faux}$ *et* $v_2(f) = \mathbf{vrai}$
- $v_3(a) = \mathbf{faux}$, $v_3(d) = \mathbf{vrai}$, $v_3(e) = \mathbf{vrai}$ *et* $v_3(f) = \mathbf{faux}$
- $v_4(a) = \mathbf{faux}$, $v_4(d) = \mathbf{vrai}$, $v_4(e) = \mathbf{faux}$ *et* $v_4(f) = \mathbf{faux}$

pour une valuation quelconque v, sont respectivement les valuations
- $v[a := \mathbf{vrai}][d := \mathbf{vrai}][e := \mathbf{vrai}][f := \mathbf{vrai}]$,
- $v[a := \mathbf{vrai}][d := \mathbf{vrai}][e := \mathbf{faux}][f := \mathbf{vrai}]$,
- $v[a := \mathbf{faux}][d := \mathbf{vrai}][e := \mathbf{vrai}][f := \mathbf{faux}]$ *et*
- $v[a := \mathbf{faux}][d := \mathbf{vrai}][e := \mathbf{faux}][f := \mathbf{faux}]$

qui interviennent dans le calcul détaillé de $I^*(\psi)(v) = \mathbf{vrai}$.

1.2.2 Sémantique des QBF prénexes

Nous revisitons la sémantique des QBF pour le fragment prénexe poli en définissant ce qu'est une solution à une QBF et comment sa vérification ou son calcul sont réalisés.

Cette définition s'appuie sur les fonctions booléennes et le lien qui peut être établi entre celles-ci et les formules propositionnelles : à toute fonction booléenne (et sur une suite de symboles propositionnels qui est omise quand il n'y a pas d'ambiguïté) peuvent être associées deux formules propositionnelles.

Définition 17 (formules associées à une fonction) *Soit $f : \mathbf{BOOL}^n \to \mathbf{BOOL}$ une fonction booléenne et $x_1, \ldots, x_n$ une suite de symboles propositionnels. À la fonction f sont*

associées respectivement positivement et négativement deux (classes de) formules propositionnelles ϕ_f et ν_f définies sur $x_1, \ldots, x_n$ qui sont telles que, pour toute valuation v, $I^*(\phi_f)(v) =$ **vrai** *si et seulement si $f(v(x_1), \ldots, v(x_n)) =$* **vrai** *et $I^*(\nu_f)(v) =$* **vrai** *si et seulement si $f(v(x_1), \ldots, v(x_n)) =$* **faux**.

La *sémantique d'une QBF prénexe* en définit la valeur de vérité selon une *valuation QBF* composée d'une valuation propositionnelle et d'une *valuation fonctionnelle*. Nous conservons la notation de la sémantique décisionnelle, l'arité ôtant toute ambiguïté.

Définition 18 (valuation QBF) *Une fonction partielle sk de l'ensemble des symboles propositionnels dans l'ensemble des fonctions booléennes est une* valuation fonctionnelle *pour une QBF prénexe polie si pour tout symbole propositionnel existentiellement quantifié x il existe un (unique) couple $(x \mapsto \hat{x}) \in sk$ tel que la fonction booléenne $\hat{x}$ a pour arité le nombre de symboles propositionnels universellement quantifiés qui précèdent x dans le lieur. L'ensemble des valuations fonctionnelles est noté* **VAL_FONC**.

Une valuation QBF est un couple constitué d'une composante propositionnelle, la valuation propositionnelle et d'une composante fonctionnelle, la valuation fonctionnelle. La valuation QBF est vide *si l'ensemble des fonctions booléennes est vide. L'ensemble des valuations QBF est noté* **VAL_QBF** $=$ **VAL_PROP** $\times$ **VAL_FONC**. *Une valuation QBF est* partielle *si tous les symboles propositionnels existentiellement quantifiés de la QBF prénexe ne sont pas présents dans la valuation fonctionnelle.*

Une valuation propositionnelle v est dans *une valuation fonctionnelle sk, pour une QBF prénexe QM telle que les symboles propositionnels existentiellement quantifiés x_i du lieur sont précédés par les symboles propositionnels universellement quantifiés $\{y_j\}_{j<j_i}$, si, pour tout symbole propositionnel existentiellement quantifié x_i, $v(x_i) = sk(v(y_1), \ldots, v(y_{j_i}))$.*

De même, une substitution σ est obtenue à partir *d'une valuation fonctionnelle sk, pour une QBF prénexe QM telle que les symboles propositionnels existentiellement quantifiés x_i du lieur sont précédés par les symboles propositionnels universellement quantifiés $\{y_j\}_{j<j_i}$, si, le support de la substitution est un préfixe du lieur et pour tout symbole propositionnel existentiellement quantifié x_i substitué dans σ, x_i est substitué par $i^{-1}(sk(i(C_1), \ldots, i(C_{j_i})))$ sachant y_j substitué par $C_j \in \{\top, \bot\}$, pour tout j, $1 \leq j \leq j_i$.*

Une valuation QBF qui n'est pas partielle peut être qualifiée, pour insister, de « valuation QBF totale ».

Une valuation fonctionnelle est un ensemble de fonctions booléennes qui sont très souvent appelées *fonctions de Skolem*. Dans la littérature, la valuation (propositionnelle) est une fonction qui associe à tout symbole propositionnel une valeur de vérité ; pour la valuation fonctionnelle, une fonction partielle est préférée à une fonction totale.

Exemple 8 (valuation QBF) *En continuant l'exemple 7, pour la QBF $\xi = \forall a \exists d \forall e \exists f \mu$ avec*

$$\mu = \neg\neg\neg((e \rightarrow d) \rightarrow \neg(f \rightarrow a))$$

La fonction partielle $\{(d \mapsto \hat{d}), (f \mapsto \hat{f})\}$, $\hat{d}$ d'arité 1 et $\hat{f}$ d'arité 2, est une valuation fonctionnelle qui forme avec toute valuation propositionnelle une valuation QBF (totale)

pour la QBF ξ ; les valuations fonctionnelles $\emptyset$, $\{(d \mapsto \hat{d})\}$ et $\{(f \mapsto \hat{f})\}$ en forment avec toute valuation propositionnelle des valuations QBF partielles. En posant $sk = \{(d \mapsto \hat{d}), (f \mapsto \hat{f})\}$ avec $\hat{d} = \{(\mathbf{vrai} \mapsto \mathbf{vrai}), (\mathbf{faux} \mapsto \mathbf{vrai})\}$ et

$$\hat{f} = \{((\mathbf{vrai}, \mathbf{vrai}) \mapsto \mathbf{vrai}), ((\mathbf{vrai}, \mathbf{faux}) \mapsto \mathbf{vrai}),\\ ((\mathbf{faux}, \mathbf{vrai}) \mapsto \mathbf{faux}), ((\mathbf{faux}, \mathbf{faux}) \mapsto \mathbf{faux})\}$$

les quatre valuations v_1, v_2, v_3 et v_4 sont dans la valuation fonctionnelle sk et la substitution

$$[a \leftarrow i^{-1}(\mathbf{faux})][d \leftarrow i^{-1}(\mathbf{vrai})][e \leftarrow i^{-1}(\mathbf{vrai})], [f \leftarrow i^{-1}(\mathbf{faux})] =\\ [a \leftarrow \bot][d \leftarrow \top][e \leftarrow \top][f \leftarrow \bot]$$

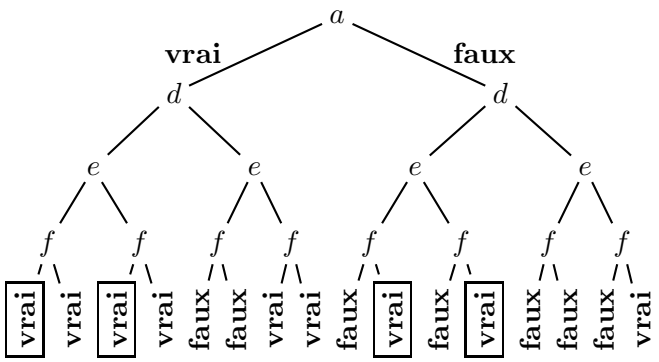
est une substitution obtenue à partir de la valuation fonctionnelle sk ($\hat{d}(i(\bot)) = \hat{d}(\mathbf{faux}) = \mathbf{vrai}$ et $\hat{f}(i(\bot), i(\top)) = \hat{f}(\mathbf{faux}, \mathbf{vrai}) = \mathbf{faux}$).

De part son alternance de quantificateurs universels et existentiels, une QBF prénexe peut donc être vue comme un jeu fini à deux joueurs dont la matrice encode les règles et conditions de victoire. Renforçant le caractère séquentiel du lieur, la valuation fonctionnelle représente alors les mouvements à jouer pour le joueur existentiel en fonction de tout mouvement joué par son adversaire, le joueur universel. Cela s'exprime alors avec élégance par une *politique* [55] (ou *stratégie* [40]) dont la représentation graphique est un arbre de décision. Une valuation (propositionnelle) dans une valuation fonctionnelle représente alors une branche d'un tel arbre et une substitution obtenue à partir d'une valuation fonctionnelle un chemin de la racine vers un nœud, permettant d'exprimer les mouvements déjà joués. À la simplicité de lecture d'une politique par rapport à une valuation fonctionnelle s'oppose une rigidité du formalisme.

Exemple 9 (politique) *En continuant l'exemple 8, la politique (ou stratégie) suivante peut être associée à la valuation fonctionnelle sk :*

$$\{a \mapsto d; \{e \mapsto f, \neg e \mapsto f\}, \neg a \mapsto d; \{e \mapsto \neg f, \neg e \mapsto \neg f\}\}$$

C'est une restriction de l'arbre sémantique pour la matrice :



qui peut être interprétée ainsi dans le cadre d'un jeu à deux joueurs : « Quoi que le joueur universel choisisse, le joueur existentiel choisit de jouer d. Si le joueur universel

a choisi de jouer a alors le joueur existentiel choisit de jouer f quelque soit le choix du joueur universel pour e ; de même, si le joueur universel a choisi de jouer $\neg a$ alors le joueur existentiel choisit de jouer $\neg f$ quelque soit le choix du joueur universel pour e. » Les quatre feuilles dans des rectangles correspondent de gauche à droite aux valuations v_1, v_2, v_3 et v_4 ; la troisième en partant de la gauche correspond à la substitution $[a \leftarrow \bot][d \leftarrow \top][e \leftarrow \top][f \leftarrow \bot]$.

Nous définissons la sémantique « prénexe » des connecteurs et constantes logiques ainsi que des quantificateurs. La différence majeure avec la sémantique décisionnelle est la possibilité de manipuler une valuation QBF et, par la même, d'offrir une réponse aux symboles propositionnels existentiellement quantifiés.

Définition 19 (sémantique « prénexe » des connecteurs et quantificateurs) *La sémantique des connecteurs et quantificateurs est définie par (v une valuation propositionnelle et sk une valuation fonctionnelle) :*

$$I_\top, I_\bot : \mathbf{VAL\_QBF} \to \mathbf{BOOL}$$

$$I_\bot(v, sk) = i_\bot \quad et \quad I_\top(v, sk) = i_\top$$

$$I_\neg : (\mathbf{VAL\_QBF} \to \mathbf{BOOL}) \times \mathbf{VAL\_QBF} \to \mathbf{BOOL}$$

$$I_\neg(f)(v, sk) = i_\neg(f(v, sk))$$

$$I_\circ : (\mathbf{VAL\_QBF} \to \mathbf{BOOL})^2 \times \mathbf{VAL\_QBF} \to \mathbf{BOOL}$$

$$I_\circ(f, g)(v, sk) = i_\circ(f(v, sk), g(v, sk)) \text{ pour tout connecteur } \circ \in \{\wedge, \vee, \to, \leftrightarrow, \oplus\}$$

$$I^x_\exists, I^x_\forall : (\mathbf{VAL\_QBF} \to \mathbf{BOOL}) \times \mathbf{VAL\_QBF} \to \mathbf{BOOL}$$

$$I^x_\exists(f)(v, sk) = i_\vee(f(v[x := \mathbf{vrai}], sk), f(v[x := \mathbf{faux}], sk))$$
$$I^x_\forall(f)(v, sk) = i_\wedge(f(v[x := \mathbf{vrai}], sk(\mathbf{vrai})), f(v[x := \mathbf{faux}], sk(\mathbf{faux})))$$

Nous sommes en mesure de définir la sémantique des QBF prénexes comme une extension de la sémantique prénexe des connecteurs, constantes et quantificateurs.

Définition 20 (sémantique des QBF prénexes) *La sémantique d'une QBF prénexe F est une fonction*

$$I^*(F) : \mathbf{VAL\_QBF} \to \mathbf{BOOL}$$

définie inductivement par :

- $I^*(\bot)(v, sk) = I_\bot(v, sk)$;
- $I^*(\top)(v, sk) = I_\top(v, sk)$;
- $I^*(x)(v, sk) = v(x)$ *si* $x \in \mathcal{SP}$;

- $I^*((G \circ H))(v, sk) = I_\circ(I^*(G), I^*(H))(v, sk)$ *si* $G, H \in \mathbf{QBF}$ *et* $\circ \in \{\wedge, \vee, \rightarrow, \leftrightarrow, \oplus\}$ *;*
- $I^*(\neg G)(v, sk) = I_\neg(I^*(G))(v, sk)$ *si* $G \in \mathbf{QBF}$ *;*
- $I^*((\exists x\ G))(v, sk) = I^*(G)(v[x := \hat{x}], sk \setminus \{(x \mapsto \hat{x})\})$ *si* $G \in \mathbf{QBF}$*,* $x \in \mathcal{SP}$ *et* $(x \mapsto \hat{x}) \in sk$ *;*
- $I^*((\exists x\ G))(v, sk) = I^x_\exists(I^*(G))(v, sk)$ *si* $G \in \mathbf{QBF}$*,* $x \in \mathcal{SP}$ *et il n'existe pas de couple* $(x \mapsto \hat{x}) \in sk$ *(*$\hat{x}$ *fonction booléenne) ;*
- $I^*((\forall x\ G))(v, sk) = I^x_\forall(I^*(G))(v, sk)$ *si* $G \in \mathbf{QBF}$ *et* $x \in \mathcal{SP}$*.*

Exemple 10 (sémantique des QBF prénexes) *En continuant l'exemple 8 et pour la valuation QBF partielle* $sk = \{(f \mapsto \hat{f})\}$ *(et une valuation propositionnelle* v *quelconque) avec*

$$\hat{f} = \{((\mathbf{vrai}, \mathbf{vrai}) \mapsto \mathbf{vrai}), ((\mathbf{vrai}, \mathbf{faux}) \mapsto \mathbf{vrai}),\\ ((\mathbf{faux}, \mathbf{vrai}) \mapsto \mathbf{faux}), ((\mathbf{faux}, \mathbf{faux}) \mapsto \mathbf{faux})\}$$

dans laquelle nous retrouvons les quatre valuations de l'exemple 8

$$\begin{array}{l} I^*(\forall a \exists d \forall e \exists f \mu)(v, sk) \\ = \quad I^a_\forall(I^*(\exists d \forall e \exists f \mu))(v, sk) \\ = \quad i_\wedge(\ I^*(\exists d \forall e \exists f \mu)(v[a := \mathbf{vrai}], sk(\mathbf{vrai})), \\ \qquad\quad I^*(\exists d \forall e \exists f \mu)(v[a := \mathbf{faux}], sk(\mathbf{faux}))) \end{array}$$

Détaillons, avec $v_a = v[a := \mathbf{vrai}]$

$$\begin{array}{l} I^*(\exists d \forall e \exists f \mu)(v_a, sk(\mathbf{vrai})) \\ = \quad I^d_\exists(I^*(\forall e \exists f \mu))(v_a, sk(\mathbf{vrai})) \\ = \quad i_\vee(\ I^*(\forall e \exists f \mu)(v_a[d := \mathbf{vrai}], sk(\mathbf{vrai})), \\ \qquad\quad I^*(\forall e \exists f \mu)(v_a[d := \mathbf{faux}], sk(\mathbf{vrai}))) \end{array}$$

avec $v_d = v_a[d := \mathbf{vrai}]$
et $sk(\mathbf{vrai}) = \{(f \mapsto \{(\mathbf{vrai} \mapsto \mathbf{vrai}), (\mathbf{faux} \mapsto \mathbf{vrai})\})\}$

$$\begin{array}{l} I^*(\forall e \exists f \mu)(v_d, \{(f \mapsto \hat{f}(\mathbf{vrai}))\}) = \\ i_\wedge(\ I^*(\exists f \mu)(v_d[e := \mathbf{vrai}], \{(f \mapsto \hat{f}(\mathbf{vrai})(\mathbf{vrai}))\}), \\ \qquad I^*(\exists f \mu)(v_d[e := \mathbf{faux}], \{(f \mapsto \hat{f}(\mathbf{vrai})(\mathbf{faux}))\})) \end{array}$$

enfin en posant $v_e = v_d[e := \mathbf{vrai}]$

$$\begin{array}{l} I^*(\exists f \mu)(v_e, \{(f \mapsto \hat{f}(\mathbf{vrai})(\mathbf{vrai}))\}) \\ = \quad I^*(\mu)(v_e[f := \hat{f}(\mathbf{vrai})(\mathbf{vrai})], \emptyset) \\ = \quad I^*(\mu)(v_e[f := \mathbf{vrai}], \emptyset) = \mathbf{vrai} \end{array}$$

Sans plus de détails, $I^*(\forall a \exists d \forall e \exists f \mu)(v, sk) = \mathbf{vrai}$*.*

Cette sémantique est particulièrement souple puisqu'elle fait cohabiter le symbole propositionnel libre dont la valeur de vérité est déterminée par la valuation propositionnelle, le symbole propositionnel existentiellement quantifié associé à une fonction booléenne dont la valeur de vérité est déterminée lorsque, dans le processus de calcul

inductif de la sémantique, le symbole propositionnel devient libre et enfin le symbole propositionnel existentiellement quantifié qui n'est pas associé à une fonction booléenne et qui s'élimine comme une disjonction sur les deux valeurs de vérité, retrouvant la sémantique du problème de décision. Ainsi, la sémantique décisionnelle n'est qu'un point de vue restreint au problème de décision de la sémantique prénexe pour les QBF prénexes (cf. Démonstrations, lemme 3) : si une valuation fonctionnelle est vide, la sémantique décisionnelle est la même que la sémantique prénexe.

Nous retrouvons aussi que si la QBF prénexe est close alors la valuation propositionnelle est sans importance (cf. Démonstrations, lemme 4).

En logique propositionnelle, la notion de solution (ou « modèle ») à une formule est définie comme étant une valuation (propositionnelle) qui rend **vrai** la formule (qui la satisfait) ; il en est de même pour le fragment prénexe des QBF où une solution (ou « modèle QBF » ou plus simplement « modèle ») est une valuation QBF qui rend **vrai** la formule.

Définition 21 (modèle QBF prénexe) *Une valuation QBF totale constituée d'une valuation propositionnelle v et d'une valuation fonctionnelle sk pour une QBF prénexe QM est un modèle QBF pour QM si $I^*(QM)(v, sk) = \mathbf{vrai}$. Une valuation propositionnelle dans la composante fonctionnelle d'un modèle QBF est un* scénario.

Cette définition de la notion de modèle QBF n'est autre pour notre description de la sémantique que celle pour les QBF sous FNC introduite dans [122, 121]. Lorsque la QBF prénexe est close alors la valuation n'importe pas et, dans la définition précédente, seule la valuation fonctionnelle définit le modèle QBF.

Exemple 11 (modèle QBF prénexe) *En continuant l'exemple 10, la valuation fonctionnelle $sk = \{(d \mapsto \hat{d}), (f \mapsto \hat{f})\}$ avec $\hat{d} = \{(\mathbf{vrai} \mapsto \mathbf{vrai}), (\mathbf{faux} \mapsto \mathbf{vrai})\}$ permet quelle que soit la valuation propositionnelle associée d'obtenir un modèle QBF pour la QBF prénexe $\forall a \exists d \forall e \exists f \mu$. « Si a est choisi par le joueur universel alors le joueur existentiel, en jouant d a la garantie qu'il existe pour lui un moyen de gagner. »*

La clôture existentielle d'une QBF prénexe admet un modèle QBF facilement déductible du modèle de la QBF prénexe en choisissant d'associer aux fonctions constantes des symboles propositionnels nouvellement quantifiés la valeur de vérité associée à ces symboles dans la valuation propositionnelle du modèle.

Les modèles d'une formule propositionnelle et les modèles d'une QBF close prénexe quantifiée uniquement existentiellement dont la matrice est précisément la formule propositionnelle sont en bijection (cf. Démonstrations, lemme 5).

La sémantique *attendue* des quantificateurs pour les QBF par rapport à la sémantique des quantificateurs en logique des prédicats (c'est-à-dire une conjonction sur l'ensemble des éléments du domaine pour la quantification universelle et une disjonction sur l'ensemble des éléments du domaine pour la quantification existentielle) est retrouvée sous la forme (cf. Démonstrations, lemme 7) :

$$I^*((\forall x\ G))(v, \emptyset) = I^*(([x \leftarrow \top](G) \wedge [x \leftarrow \bot](G)))(v, \emptyset)$$

et

$$I^*((\exists x\ G))(v,\emptyset) = I^*(([x \leftarrow \top](G) \vee [x \leftarrow \bot](G)))(v,\emptyset).$$

Nous redéfinissons le problème de validité selon la sémantique pour les QBF prénexes et introduisons la notion de « satisfiabilité » d'une QBF prénexe comme expression de l'existence d'un modèle.

Définition 22 (validité et satisfiabilité) *Une QBF prénexe F est* valide *si, pour toute valuation propositionnelle v, $I^*(F)(v,\emptyset) = \mathbf{vrai}$. Une QBF prénexe F est* satisfiable *s'il existe une valuation propositionnelle v telle que $I^*(F)(v,\emptyset) = \mathbf{vrai}$.*

La clôture universelle d'une QBF prénexe, ce qui correspond à quantifier universellement le plus à l'extérieur les symboles propositionnels libres, est valide si la QBF prénexe est valide ; la clôture existentielle d'une QBF prénexe est valide si la QBF prénexe est satisfiable. Si la QBF prénexe est close alors les notions de validité et de satisfiabilité se recouvrent.

Comme pour la logique propositionnelle, pour laquelle rechercher la satisfiabilité est équivalent à rechercher un modèle, la question du problème de satisfiabilité d'une QBF prénexe peut être reformulée ainsi : « Existe-t-il un modèle QBF pour la QBF prénexe F ? ». Si la QBF prénexe n'est pas close alors la validité implique l'existence d'un modèle.

Exemple 12 (validité et modèle) *La QBF prénexe $(\exists a\ (a \wedge d))$ admet pour modèle QBF la valuation QBF $(v[d := \mathbf{vrai}], \{(a \mapsto \mathbf{vrai})\})$ (v une valuation propositionnelle quelconque) mais n'est pas valide (pas de modèle avec $v[d := \mathbf{faux}]$), et sa clôture existentielle $(\exists d\ (\exists a\ (a \wedge d)))$ admet pour modèle QBF la valuation QBF $(v, \{(a \mapsto \mathbf{vrai}), (d \mapsto \mathbf{vrai})\})$ mais sa clôture universelle $(\forall d\ (\exists a\ (a \wedge d)))$ n'admet pas de modèle QBF.*

Un modèle QBF pour une QBF prénexe close étant un ensemble de fonctions booléennes, par la définition 17 qui associe positivement à une fonction booléenne une formule, la formule propositionnelle née de la substitution dans la matrice des symboles existentiellement quantifiés par les formules associées uniquement constituées des symboles universellement quantifiés est une *tautologie* (cf. Démonstrations, lemme 9). Si les formules associées sont étendues aux symboles propositionnels libres, par le même procédé, un modèle QBF pour une QBF prénexe non close peut être défini [122]. Le test pour décider si une valuation QBF est un modèle QBF ou non est donc $co - \mathcal{NP}$-complet [123].

Exemple 13 (modèle QBF et tautologie) *En continuant l'exemple 11 et en considérant les formules propositionnelles $\phi_{\hat{d}} = \mathbf{vrai}$ et $\phi_{\hat{f}} = a$, associées (positivement) aux fonctions booléennes $\hat{d}$ et $\hat{f}$, la formule propositionnelle $[d \leftarrow \phi_{\hat{d}}][f \leftarrow \phi_{\hat{f}}](\mu)$ est une tautologie.*

La forme normale conjonctive sert très souvent de définition même à la notion de QBF. Le modèle QBF a alors une autre définition possible dans cette restriction [25] : les fonctions booléennes du modèle QBF sont associées positivement et négativement ; la formule propositionnelle née de la substitution dans la matrice des littéraux positifs existentiellement quantifiés par les formules associées positivement et des littéraux négatifs existentiellement quantifiés par les formules associées négativement est une tautologie (cf. Démonstrations, lemme 10).

1.3 Deux relations d'équivalence

Les deux sémantiques des QBF suggèrent deux *relations d'équivalence*[234] : une extension de la relation d'équivalence classique ($\equiv$) sur la préservation des modèles propositionnelles et une relation d'équivalence ($\cong$) sur la préservation des modèles QBF, partitionnant les classes de l'équivalence classique. Nous redéfinissons la première de ces deux relations d'équivalence et retrouvons pour les QBF des résultats de la logique des prédicats. La définition d'une relation d'équivalence pour les QBF prénexes sur la préservation des modèles QBF offre de multiples choix dont nous explorons une variante. Nous retrouvons alors les résultats de la logique des prédicats pour les formes normales classiques.

1.3.1 Relation d'équivalence préservant les modèles propositionnels

La relation d'équivalence classique de la logique propositionnelle, qui préserve les modèles propositionnels, partitionne pour un ensemble de n symboles propositionnels l'espace des formules propositionnelles en 2^{2^n} classes. Cette relation s'étend de manière naturelle à l'ensemble **QBF**. Cette relation est définie sur la sémantique décisionnelle mais peut être trivialement étendue à la sémantique prénexe.

Définition 23 (Relation $\equiv$) *Soient F et G deux QBF. $F \equiv G$ si, pour toute valuation propositionnelle v, $I^*((F \leftrightarrow G))(v) =$* **vrai**.

Cette définition met en lien la bi-implication ($\leftrightarrow$) au niveau objet et la relation de préservation des modèles propositionnels ($\equiv$) mais elle peut aussi de part la sémantique de la bi-implication (sous les mêmes conditions et de manière équivalente) être définie par

$$I^*(F)(v) = I^*(G)(v).$$

La relation $\equiv$ est bien une relation d'équivalence ce qui est immédiat car l'égalité est une relation d'équivalence (cf. Démonstrations, lemme 11).

La sémantique pour des QBF sans quantificateur étant la même que celle de la logique propositionnelle, la relation $\equiv$ sur les QBF sans quantificateur est identique à celle sur les formules propositionnelles (d'où l'abus de notation).

Le résultat classique de la logique des prédicats sur l'anonymat des variables quantifiées s'étend aux QBF : nous pouvons disposer (de manière cohérente) des noms des symboles propositionnels liés, par contre, ceux qui sont libres « ne nous appartiennent pas », et ne peuvent être renommés.

Nous récapitulons quelques résultats classiques utiles pour les chapitres suivants (cf. Démonstrations, lemme 13) :

($Q^{\equiv}$.1) si $F \equiv G$ et x lié ni dans F ni dans G alors $qxF \equiv qxG$;
($Q^{\equiv}$.2) $\exists x Q(x \wedge G) \equiv Q[x \leftarrow \top](G)$;
($Q^{\equiv}$.3) $\exists x Q(\neg x \wedge G) \equiv Q[x \leftarrow \bot](G)$;
($Q^{\equiv}$.4) $\forall x Q(x \vee G) \equiv Q[x \leftarrow \bot](G)$;
($Q^{\equiv}$.5) $\forall x Q(\neg x \vee G) \equiv Q[x \leftarrow \top](G)$;
($Q^{\equiv}$.6) $\forall x x \equiv \bot$;

($Q^{\equiv}$.7) $\forall x \neg x \equiv \bot$;
($Q^{\equiv}$.8) $\exists x x \equiv \top$;
($Q^{\equiv}$.9) $\exists x \neg x \equiv \top$.
(F et G deux QBF, x et y deux symboles propositionnels, Q un lieur et q un quantificateur.)

De nombreuses *équivalences logiques* dues principalement aux propriétés des connecteurs et à celles des quantificateurs entre-eux sont conservées pour les QBF : les équivalences propositionnelles 0.1 *à* 0.30 et les équivalences du prédicatif 1.1 *à* 1.14. Nous remarquons, qu'en général, $(\exists x\ (\forall y\ F)) \not\equiv (\forall x\ (\exists y\ F))$ comme par exemple

$$(\forall d\ (\exists a\ (\exists c\ ((c \vee a) \leftrightarrow d)))) \not\equiv (\exists a\ (\exists c\ (\forall d\ ((c \vee a) \leftrightarrow d)))).$$

Les équivalences $Q^{\equiv}$.2 et $Q^{\equiv}$.3 forment le principe de la *propagation de clauses unitaires*. L'ensemble des QBF closes est partitionné en uniquement deux classes : la classe des QBF valides $[\top]_{\equiv}$ et la classe des QBF non valides $[\bot]_{\equiv}$.

Exemple 14

$$\begin{array}{ll} & (\forall a\ (\exists d\ (\forall e\ (\exists f\ \neg\neg\neg((e \rightarrow d) \rightarrow \neg(f \rightarrow a)))))) \\ \equiv & (\forall a\ (\exists d\ (\forall e\ \neg\neg\neg((e \rightarrow d) \rightarrow (\forall f\ \neg(f \rightarrow a)))))) \end{array}$$

1.3.2 Relation d'équivalence préservant les modèles QBF

La relation $\equiv$ ne préserve que la composante propositionnelle de la valuation QBF des QBF prénexes mais n'informe pas sur la préservation de la composante fonctionnelle ce qui est crucial pour l'interprétation d'une QBF prénexe comme un jeu fini à deux joueurs ou dans la compilation de celle-ci. Nous proposons une relation d'équivalence préservant les modèles QBF pour les QBF *prénexes*.

Définition 24 (relation $\cong$)
Soient F et G deux QBF prénexes de lieurs identiques jusqu'à la dernière existentielle. $F \cong G$ si, pour toute valuation propositionnelle v et pour toute valuation fonctionnelle sk pour F et G, $i_{\leftrightarrow}(I^*(F)(v, sk), I^*(G)(v, sk)) = \mathbf{vrai}$.

La définition est plus complexe que celle de la définition 23 car la QBF $(F \leftrightarrow G)$, pour deux QBF prénexes F et G, n'est pas une QBF prénexe (sauf si bien sûr F et G sont sans quantificateur) ; elle est aussi différente de celle présentée initialement dans [207] car elle autorise à réunir dans une même classe d'équivalence des QBF prénexes ayant des lieurs qui diffèrent sur les universelles les plus internes prenant ainsi en compte le fait qu'un modèle QBF est constitué de fonctions booléennes dépendant des symboles propositionnels universellement quantifiés qui précèdent le symbole propositionnel existentiellement quantifié associé à la fonction. Ainsi, bien que les lieurs soient différents $\exists x x \cong \exists x \forall y((x \vee y) \wedge (y \rightarrow x))$ car ces deux QBF prénexes ont un unique modèle QBF (dont la composante fonctionnelle est) $\{(x \mapsto \mathbf{vrai})\}$; mais $\exists x x \ncong \forall y \exists x((x \vee y) \wedge (y \rightarrow x))$ car l'unique modèle QBF de cette dernière QBF prénexe est de valuation fonctionnelle $\{(x \mapsto \{(\mathbf{vrai} \mapsto \mathbf{vrai}), (\mathbf{faux} \mapsto \mathbf{vrai})\})\}$.

De la même manière que pour la relation d'équivalence classique, cette définition met en lien la bi-implication ($\leftrightarrow$) au niveau objet et la relation de préservation des modèles mais elle peut aussi de part la sémantique de la bi-implication (sous les mêmes conditions et de manière équivalente) être définie par

$$I^*(F)(v, sk) = I^*(G)(v, sk).$$

La relation d'équivalence préservant les modèles QBF partitionne les classes de l'équivalence classique (sauf celle de $\bot$) en une infinité de classes. Ainsi $\exists xx \equiv \top$ mais $\exists xx \not\cong \top$ car l'unique modèle QBF de $\top$ est la valuation QBF vide alors que l'unique modèle QBF de $\exists xx$ est la valuation QBF (dont la composante fonctionnelle est) $\{(x \mapsto \mathbf{vrai})\}$. Les symboles propositionnels existentiellement quantifiés ne sont pas anonymes pour la relation d'équivalence préservant les modèles QBF en effet $\exists xx \equiv \exists yy$ mais $\exists xx \not\cong \exists yy$ tandis que les symboles propositionnels universellement quantifiés sont anonymes pour la relation d'équivalence préservant les modèles QBF puisque ne figurant pas dans la représentation choisie de la valuation QBF[4].

La relation $\cong$ est bien une relation d'équivalence ce qui est encore une fois immédiat car l'égalité est une relation d'équivalence (cf. Démonstrations, lemme 15). La relation $\cong$ partitionne la classe $[\top]_\equiv$ et donc (Démonstrations, lemme 16) :

- Si $F \cong F'$ alors $F \equiv F'$.
- De plus si F et F' sont deux QBF sans quantificateur alors $F \equiv F'$ si et seulement si $F \cong F'$.

Les résultats classiques ne s'étendent que partiellement à la relation d'équivalence préservant les modèles QBF (cf. Démonstrations, lemme 18) :

($Q^\cong$.1) $\exists x \exists y F \cong \exists y \exists x F$;
($Q^\cong$.2) $I^*(\forall x \forall y F)(v, sk) = I^*(\forall y \forall x F)(v, sk')$ avec sk et sk' identiques à ceci près que les deux premières colonnes sont permutées ;
($Q^\cong$.3) $\exists x Q(x \wedge H) \cong \exists x Q(x \wedge [x \leftarrow \top](H))$;
($Q^\cong$.4) $\exists x Q(\neg x \wedge H) \cong \exists x Q(\neg x \wedge [x \leftarrow \bot](H))$;
($Q^\cong$.5) $I^*(\forall x Q(x \vee H))(v, sk) = I^*(Q[x \leftarrow \bot](H))(v, sk(\mathbf{faux}))$;
($Q^\cong$.6) $I^*(\forall x Q(\neg x \vee H))(v, sk) = I^*(Q[x \leftarrow \top](H))(v, sk(\mathbf{vrai}))$;
($Q^\cong$.7) $\forall xx \cong \bot$;
($Q^\cong$.8) $\forall x \neg x \cong \bot$.

(F et G deux QBF prénexes, H une formule propositionnelle, x et y deux symboles propositionnels et Q un lieur.)

La définition 24 est extrêmement discutable dans la mesure où elle introduit une dissymétrie entre la manipulation algébrique du quantificateur existentiel et celle du quantificateur universel : $\forall x \forall y F \cong \forall y \forall x F$ n'est pas un théorème ; pour qu'il le soit, il n'est pas difficile de modifier la définition mais les conséquences techniques sont fastidieuses.

[4]La perte de l'anonymat des symboles propositionnels existentiellement quantifiés n'est pas vraiment une difficulté puisqu'il est possible de l'obtenir en choisissant une notation par référence comme celle des indices de de Bruijn pour le λ-calcul [185].

1.3.3 Mises sous forme prénexe et sous formes normales

La plus part des procédures de décision pour les QBF prennent en format d'entrée des formules prénexes sous forme normale conjonctive (cf. chapitre 2). Si nous nous attardons sur les formes normales, c'est que leur choix, en tant que format d'entrée ou structure interne aux procédures de décisions QBF, et leur obtention, à partir d'une représentation d'une application, influencent grandement l'efficacité (cf. chapitre 2) et la manière de programmer (cf. chapitre 6).

En *logique des prédicats*, la mise sous forme prénexe et la mise sous forme normale conjonctive sont combinées en une mise sous forme normale clausale[5]. La forme normale clausale a été particulièrement étudiée [169] car elle est le format d'entrée de la *résolution*. Après l'obtention d'une forme normale conjonctive prénexe, la forme normale clausale est obtenue par *skolémisation* (qui élimine les variables existentiellement quantifiées pour les substituer par des symboles de fonction ayant pour arguments les variables universellement quantifiées qui précèdent la variable substituée) puis par redistribution des quantificateurs universels pour obtenir une conjonction de *clauses*. La définition des QBF étant sans symbole de fonction, la skolémisation est impossible (seule un *ersatz* par *skolémisation propositionnelle* existe).

Pour les QBF, comme pour la logique des prédicats, la première étape de la mise sous forme prénexe ou de la mise sous FNC est toujours la *linéarisation*, qui élimine les bi-implications et les ou-exclusifs par les équivalences $(F{\leftrightarrow}G) \overset{0.3}{\equiv} ((F{\rightarrow}G){\wedge}(G{\rightarrow}F))$ et $(F{\oplus}G) \overset{0.30}{\equiv} ((F{\vee}G){\wedge}\neg(F{\wedge}G))$. Cette étape est très souvent négligée et omise dans le traitement des QBF comme ayant déjà été traitée mais elle ne doit pas l'être car elle peut avoir un effet exponentiel sur la taille de la formule (cf. chapitre 6).

La mise sous forme prénexe d'une QBF se réalise alors en deux étapes supplémentaires :

1. le renommage des symboles propositionnels apparaissant associées à des quantificateurs différents en de nouveaux symboles propositionnels (c'est-à-dire n'apparaissant pas dans la formule) pour atteindre une forme polie grâce à l'anonymat des symboles propositionnels liés ;
2. l'extraction des quantificateurs grâces aux équivalences (pour q un quantificateur quelconque) $((qx\ F){\vee}G) \overset{1.5}{\equiv} (qx\ (F{\vee}G))$ et $((qx\ F){\wedge}G) \overset{1.6}{\equiv} (qx\ (F{\wedge}G))$ utilisées de gauche à droite (et à la commutativité de la disjonction et de la conjonction).

Pour obtenir la forme normale conjonctive (ou disjonctive) prénexe ou non, il faut en calculer la forme normale négative à partir de la QBF linéarisée. La mise sous FNN s'opère selon l'application des étapes suivantes :

1. l'élimination des implications par les équivalences (au choix) $(F{\rightarrow}G) \overset{0.1}{\equiv} (\neg F{\vee}G)$ et $(F{\rightarrow}G) \overset{0.2}{\equiv} \neg(F{\wedge}\neg G)$;
2. la « descente » des négations sur les littéraux par les lois de de Morgan $\neg(F{\vee}G) \overset{0.5}{\equiv}$

[5]Dans la littérature anglophone, en logique du premier ordre, CNF signifie généralement « clausal normal form » tandis qu'en QBF et logique propositionnelle, cet acronyme signifie plutôt « conjunctive normal form ».

$(\neg F \wedge \neg G)$ et $\neg(F \wedge G) \overset{0.6}{\equiv} (\neg F \vee \neg G)$, la traversée des quantificateurs par les équivalences $\neg(\forall x\ F) \overset{1.3}{\equiv} (\exists x\ \neg F)$ et $\neg(\exists x\ F) \overset{1.4}{\equiv} (\forall x\ \neg F)$ ainsi que l'élimination des doubles négations par l'involution de la négation $\neg\neg F \overset{0.4}{\equiv} F$.

Les équivalences étant toutes utilisées de gauche à droite.

Exemple 15 (mise sous forme normale negative) *En continuant l'exemple 5,*

$$\begin{array}{ll} & (\forall a\ (\exists d\ ((\exists e\ ((a \wedge \neg e) \rightarrow d)) \rightarrow (a \wedge d)))) \\ \overset{0.1}{\equiv} & (\forall a\ (\exists d\ (\neg(\exists e\ (\neg(a \wedge \neg e) \vee d)) \vee (a \wedge d)))) \\ \overset{1.4}{\equiv} & (\forall a\ (\exists d\ ((\forall e\ \neg(\neg(a \wedge \neg e) \vee d)) \vee (a \wedge d)))) \\ \overset{0.5}{\equiv} & (\forall a\ (\exists d\ ((\forall e\ (\neg\neg(a \wedge \neg e) \wedge \neg d)) \vee (a \wedge d)))) \\ \overset{0.4}{\equiv} & (\forall a\ (\exists d\ ((\forall e\ ((a \wedge \neg e) \wedge \neg d)) \vee (a \wedge d)))) \end{array}$$

À partir de la forme normale négative, la mise sous FNC se réalise par une mise sous forme prénexe suivie par deux méthodes possibles : l'« académique » [169] par distributivité selon l'équivalence $(F \vee (G \wedge H)) \overset{0.24}{\equiv} ((F \vee G) \wedge (F \vee H))$ (et la commutativité de la disjonction) et celle dite « par renommage de formules » qui s'appuie sur l'introduction de nouveaux symboles propositionnels [223, 177, 43, 74]. La première fait dans le pire des cas croître exponentiellement la taille de la formule tandis que la seconde augmente le nombre de symboles propositionnels (un par occurrence de connecteurs binaires) et donc l'espace de recherche. La mise sous FND est similaire à la mise sous FNC « académique » mais en optant pour l'équivalence logique duale à 0.24 : $(F \wedge (G \vee H)) \overset{0.23}{\equiv} ((F \wedge G) \vee (F \wedge H))$.

Exemple 16 (mise sous FNC académique) *En continuant l'exemple 15,*

$$\begin{array}{ll} & (\forall a\ (\exists d\ ((\forall e\ ((a \wedge \neg e) \wedge \neg d)) \vee (a \wedge d)))) \\ \overset{1.5}{\equiv} & (\forall a\ (\exists d\ (\forall e\ (((a \wedge \neg e) \wedge \neg d) \vee (a \wedge d))))) \\ \overset{0.24}{\equiv} & (\forall a\ (\exists d\ (\forall e\ ((((a \wedge \neg e) \wedge \neg d) \vee a) \wedge (((a \wedge \neg e) \wedge \neg d) \vee d))))) \\ \overset{0.20}{\equiv} & (\forall a\ (\exists d\ (\forall e\ ((a \vee ((a \wedge \neg e) \wedge \neg d)) \wedge (d \vee ((a \wedge \neg e) \wedge \neg d)))))) \\ \overset{0.24}{\equiv} & (\forall a\ (\exists d\ (\forall e\ (((a \vee (a \wedge \neg e)) \wedge (a \vee \neg d)) \wedge ((d \vee (a \wedge \neg e)) \wedge (d \vee \neg d)))))) \\ \overset{0.24}{\equiv} & (\forall a\ (\exists d\ (\forall e\ ((((a \vee a) \wedge (a \vee \neg e)) \wedge (a \vee \neg d)) \wedge (((d \vee a) \wedge (d \vee \neg e)) \wedge (d \vee \neg d)))))) \end{array}$$

Cette dernière QBF est sous FNC mais des simplifications peuvent être opérées pour obtenir :

$$\begin{array}{ll} & (\forall a\ (\exists d\ ((\forall e\ ((a \wedge \neg e) \wedge \neg d)) \vee (a \wedge d)))) \\ \equiv & (\forall a\ (\exists d\ (\forall e\ (((a \wedge (a \vee \neg e)) \wedge (a \vee \neg d)) \wedge ((d \vee a) \wedge (d \vee \neg e)))))) \end{array}$$

Pour les QBF, la mise sous FNC par « renommage de formules » est basée sur l'équivalence (G une QBF prénexe, x un symbole propositionnel, F une QBF librement substituable pour x dans G) $(\exists x\ ((x \leftrightarrow F) \wedge G)) \equiv [x \leftarrow F](G)$ qui permet d'introduire un nouveau symbole existentiellement quantifié mis en équivalence avec une sous formule

(cf. Démonstrations, lemme 20). Dans l'équivalence précédente, l'introduction du symbole propositionnel existentiellement quantifié par une implication et non par une bi-implication n'est pas suffisante pour la mise sous FNC *a contrario* de la logique des prédicats [169] $((\exists x\ ((x{\rightarrow}F){\wedge}G)) \not\equiv [x \leftarrow F](G)$[6]) .

Si l'équivalence est restreinte aux deux types de formules pour F : $(l{\vee}l')$ et $(l{\wedge}l')$, l et l' deux littéraux, alors des clauses propositionnelles sont introduites :

$$(x{\leftrightarrow}(l{\vee}l')) \equiv (\neg x{\vee}l{\vee}l'){\wedge}(x{\vee}\neg l){\wedge}(x{\vee}\neg l') \text{ et } (x{\leftrightarrow}(l{\wedge}l')) \equiv (x{\vee}\neg l{\vee}\neg l'){\wedge}(\neg x{\vee}l){\wedge}(\neg x{\vee}l')$$

ce qui nous mène à l'algorithme 1, $msfnc\_renommage\_formules$, qui permet de calculer la FNC d'une QBF sous FNN prénexe, QM, par « renommage de formules » par introduction dans la matrice passée en premier argument de nouveaux symboles propositionnels indicés par le second argument : si $(Q_{\exists}, D) = msfnc\_renommage\_formules(M, 0)$ alors $QQ_{\exists}D$ est sous FNC.

Exemple 17 (mise sous FNC par renommage) *En continuant l'exemple 16,*

$$\begin{array}{ll} & (\forall a\ (\exists d\ ((\forall e\ ((a{\wedge}\neg e){\wedge}\neg d)){\vee}(a{\wedge}d)))) \\ \overset{1.5}{\equiv} & \forall a\exists d\forall e(((a{\wedge}\neg e){\wedge}\neg d){\vee}(a{\wedge}d)) \end{array}$$

puis en suivant l'algorithme 1 :

$$\begin{array}{ll} & \forall a\exists d\forall e((\forall e\ ((a{\wedge}\neg e){\wedge}\neg d)){\vee}(a{\wedge}d)) \\ \equiv & \forall a\exists d\forall e\exists z_1(((z_1{\wedge}\neg d){\vee}(a{\wedge}d)){\wedge}(z_1{\leftrightarrow}(a{\wedge}\neg e))) \\ \equiv & \forall a\exists d\forall e\exists z_1\exists z_2 \\ & (((z_2{\vee}(a{\wedge}d)){\wedge}(z_1{\leftrightarrow}(a{\wedge}\neg e))){\wedge}(z_2{\leftrightarrow}(z_1{\wedge}\neg d))) \\ \equiv & \forall a\exists d\forall e\exists z_1\exists z_2\exists z_3 \\ & ((((z_2{\vee}z_3){\wedge}(z_1{\leftrightarrow}(a{\wedge}\neg e))){\wedge}(z_2{\leftrightarrow}(z_1{\wedge}\neg d))){\wedge}(z_3{\leftrightarrow}(a{\wedge}d))) \\ \equiv & \forall a\exists d\forall e\exists z_1\exists z_2\exists z_3\exists z_4 z_4{\wedge} \\ & (z_1{\vee}\neg a{\vee}e){\wedge}(\neg z_1{\vee}a){\wedge}(\neg z_1{\vee}\neg e){\wedge} \\ & (z_2{\vee}\neg z_1{\vee}d){\wedge}(\neg z_2{\vee}z_1){\wedge}(\neg z_2{\vee}\neg d){\wedge} \\ & (z_3{\vee}\neg a{\vee}\neg d){\wedge}(\neg z_3{\vee}a){\wedge}(\neg z_3{\vee}d){\wedge} \\ & (\neg z_4{\vee}z_2{\vee}z_3){\wedge}(z_4{\vee}\neg z_2){\wedge}(\neg z_4{\vee}\neg z_3) \\ & \textit{ce qui peut être réduit en} \\ \overset{Q.2}{\equiv} & \forall a\exists d\forall e\exists z_1\exists z_2\exists z_3 \\ & (z_1{\vee}\neg a{\vee}e){\wedge}(\neg z_1{\vee}a){\wedge}(\neg z_1{\vee}\neg e){\wedge} \\ & (z_2{\vee}\neg z_1{\vee}d){\wedge}(\neg z_2{\vee}z_1){\wedge}(\neg z_2{\vee}\neg d){\wedge} \\ & (z_3{\vee}\neg a{\vee}\neg d){\wedge}(\neg z_3{\vee}a){\wedge}(\neg z_3{\vee}d){\wedge}(z_2{\vee}z_3) \end{array}$$

Que ce soit l'équivalence qui préserve les modèles propositionnels pour une QBF quelconque ou l'équivalence qui préserve les modèles QBF pour une QBF prénexe, la linéarisation termine et calcule une QBF équivalente sans bi-implication, ni ou-exclusif

[6] pour $G = \neg x$ et $F = \top$ alors $[x \leftarrow F](G) \equiv \bot$ mais $(\exists x\ ((x{\rightarrow}F){\wedge}G)) \equiv \top$ (tandis que $(\exists x\ ((x{\leftrightarrow}F){\wedge}G)) \equiv \bot$).

Algorithme 1 $msfnc\_renommage\_formules$

Entrée: Une formule propositionnelle F
Entrée: Un entier n
Sortie: Une paire composée d'un lieur et d'une formule propositionnelle
  si $F = \top$ ou $F = \bot$ ou $F = \neg x$ ou $F = x$ avec $x \in \mathcal{SP}$ **alors**
    retourner (ε, F)
  sinon
    $F = [z_n \leftarrow (x \circ y)](G)$, avec $z_n, |x|, |y| \in \mathcal{SP}$
    $(Q, D) := msfnc\_renommage\_formules(G, n+1)$
    si $\circ = \wedge$ **alors**
      $cl := (z_n \vee \overline{x} \vee \overline{y}) \wedge (\neg z_n \vee x) \wedge (\neg z_n \vee y)$
    sinon
      $cl := (\neg z_n \vee x \vee y) \wedge (z_n \vee \overline{x}) \wedge (z_n \vee \overline{y})$
    fin si
    retourner $(Q \exists z_n, (cl \wedge D))$
  fin si

(cf. Démonstrations, lemmes 21 et 22). De même, pour toute QBF, il existe une QBF sous FNN (resp. sous FND, sous FNC) qui lui est équivalente et la mise sous FNN (resp. sous FND, sous FNC « académique ») termine et calcule une QBF sous FNN équivalente (resp. sous FND, sous FNC) (cf. Démonstrations, lemmes 21 et 22).

Pour l'équivalence qui préserve les modèles propositionnels, la mise sous forme prénexe termine et calcule une QBF équivalente sous forme prénexe et la mise sous FNC par la méthode par « renommage de formules » termine et calcule une QBF sous FNC équivalente (cf. Démonstrations, lemme 21). Ces résultats de complétude ne peuvent pas s'étendre directement à l'équivalence qui préserve les modèles QBF pour la mise sous FNC par « renommage des formules » puisqu'il y a l'introduction de nouveaux symboles propositionnels existentiellement quantifiés mais un résultat similaire peut être obtenu si les modèles de la QBF sous FNC sont « purgés » de ces nouveaux symboles (cf. Démonstrations, lemme 23).

1.4 Calcul syntaxique pour les QBF prénexes

La *logique des prédicats* marche sur deux jambes : la sémantique qui est définie sur une interprétation des symboles non-logiques, étudiée dans la « théorie des modèles », et de nombreux systèmes syntaxiques, étudiés dans la « théorie de la preuve », définis par des ensembles de règles purement symboliques qui ignorent la signification associée aux symboles non-logiques. Dans un système syntaxique, à partir d'une formule et en suivant scrupuleusement les règles, il est possible de construire des preuves. Ces systèmes de règles ne peuvent être quelconques pour être élus, ils doivent nécessairement garantir qu'à toute formule vraie pour toute interprétation une preuve peut-être associée (la complétude) et que toute preuve ne peut émaner que d'une formule vraie pour toute interprétation (la correction). Face à l'infinité des interprétations possibles, l'approche syntaxique est la

voie algorithmique principale pour faire de la logique un moyen de calcul avec des succès incontestables tels que les langages de la *programmation logique*, héritiers de la *résolution* de Robinson, qui est l'archétype de la méthode syntaxique. La *logique propositionnelle*, de part sa définition enchâssée dans celle de la logique des prédicats, bénéficie aussi de ces deux jambes mais, du fait de la simplicité de la sémantique, la frontière algorithmique des approches est assez floue. Il en est de même pour les QBF : à l'intérieur d'une méthode sémantique basée sur la définition des quantificateurs peuvent être insérés des éléments de résolution (cf. chapitre 2). Il existe néanmoins des méthodes syntaxiques qui ne font pas référence à la définition de la sémantique des quantificateurs, comme par exemple la *Q-résolution*, extension de la résolution propositionnelle, elle-même restriction de la résolution de Robinson.

Nous présentons notre calcul syntaxique comme une extension, pour les QBF prénexes, de la justification de l'*algorithme de Stålmarck* [202], dans le cadre de la théorie de la preuve, qui a pour objectif de démontrer qu'une formule propositionnelle est une *tautologie*. L'idée de cet algorithme est triple : de part le caractère fonctionnel de la logique propositionnelle, toute sous-formule d'une formule (y compris la formule elle-même et les symboles propositionnels qui la composent) est *in fine* équivalente soit à $\top$, soit à $\bot$; les propriétés algébriques des connecteurs logiques induisent que pour que la formule soit équivalente à $\top$ (pour que la formule soit une tautologie), certaines de ses sous formules soient nécessairement équivalentes, au sein d'une « relation de formules », à d'autres de ses sous-formules (ou à $\top$ ou à $\bot$) ; lorsque les deux idées précédentes sont insuffisantes pour décider complètement, il est toujours possible de forcer l'équivalence d'un symbole propositionnel à $\top$ et de démontrer que cette nouvelle formule est une tautologie puis de faire de même avec $\bot$ et d'en déduire alors que la formule est une tautologie.

Nous présentons tout d'abord l'extension immédiate de la relation de formules aux QBF ; nous définissons le système syntaxique S_{QBF} le plus élémentaire basé sur la relation de formules ; pour en démontrer la correction et la complétude, nous exprimons le lien entre ce système et la sémantique des QBF ; nous évoquons des extensions possibles que nous développerons plus encore dans le chapitre 4 ; enfin, nous reconstruisons le modèle QBF attaché à toute preuve dans ce système.

1.4.1 Relation de formules

Une *relation de formules* [202] (propositionnelle) R pour une formule F est définie comme étant une relation d'équivalence sur l'ensemble étendu des sous-formules, $sfe(F)$ (cf. définition 3), avec pour contrainte que pour tout élément $A, B \in sfe(F)$, si $(A, B) \in R$ alors $(\overline{A}, \overline{B}) \in R$. Une classe d'une relation de formules R contenant un élément A est notée $[A]_R$ (et plus simplement $[A]$ lorsque la relation de formules est clairement explicitée par le contexte). La sémantique attendue est que tous les éléments d'une même classe d'équivalence ont la même valeur de vérité. Nous étendons ce formalisme aux QBF prénexes (closes et polies).

Définition 25 (relation de formules) *Une relation de formules $R = (Q\ ;\ P)$ pour une QBF prénexe QM est constituée du lieur Q et d'une partition P de $sfe(M)$ sous la*

contrainte que, pour tout $A, B \in sfe(M)$*, si* $(A, B) \in R$ *alors* $(\overline{A}, \overline{B}) \in R$*.*

La relation de formules la plus fine pour une QBF prénexe QM est la relation identité Id_{QM} qui est la partition des singletons de $sfe(M)$ associée au lieur Q. La relation de formules $R([A]_R = [B]_R)$ (notée plus simplement par la suite $R([A] = [B])$) pour une QBF prénexe QM est la relation de formules la plus fine pour QM telle qu'elle est un raffinement de R et $([A]_R = [B]_R)$. La relation de formules $Id_{QM}([M] = [\top])$ nous servira de racine à l'arbre de déduction de notre calcul syntaxique. Dans ce qui suit seule une des deux classes symétriques $[A]$ et $[\overline{A}]$, sauf mention, est explicitée.

Exemple 18 *Soit la QBF*

$$\xi = \forall a \exists d \forall e \exists f \neg\neg\neg((e \rightarrow d) \rightarrow \neg(f \rightarrow a))$$

alors

$$Id_\xi([\neg\neg\neg\nu] = [\top]) = \\ (\forall a \exists d \forall e \exists f \ ; \ [\top, \neg\neg\neg\nu], [a], [d], [e], [f], [\nu_0], [\nu_1], [\neg\neg\nu], [\neg\nu], [\nu])$$

avec $\nu = (\nu_0 \rightarrow \neg\nu_1)$*,* $\nu_0 = (e \rightarrow d)$ *et* $\nu_1 = (f \rightarrow a)$*.*

1.4.2 Système syntaxique

Nous présentons notre système syntaxique S_{QBF} pour les QBF prénexes comme un ensemble de règles $\frac{conclusion}{prémisse}$ qui opèrent sur des relations de formules formant un arbre de déduction dont la racine est la relation de formules $Id_{QM}([M] = [\top])$. Mais avant de présenter le système nous définissons un ensemble de relations de formules qui correspondent à des relations de formules à partir desquelles aucune déduction n'est possible.

Définition 26 (relation de formules explicitement contradictoire) *Une relation de formules est* explicitement contradictoire *si une des conditions suivantes est vérifiée :*

1. *il existe une formule* F *dans la relation de formules telle que* $[F] = [\overline{F}]$ *;*
2. *il existe une classe qui contient au moins deux symboles propositionnels universellement quantifiés du lieur ;*
3. *il existe une classe qui contient un symbole propositionnel universellement quantifié* u *du lieur et un symbole propositionnel existentiellement quantifié* e *tel que* $e \prec u$*.*

La condition 1 maintient intuitivement qu'une formule et son complémentaire ne peuvent avoir la même valeur de vérité. La condition 2 exprime que deux symboles universellement quantifiés ne sont jamais liés fonctionnellement. La condition 3 correspond dans le cadre de l'*unification* de *termes* de la *logique des prédicats* au classique *test d'occurrence*. Une relation de formules peut n'être pas explicitement contradictoire et contenir des classes qui le sont.

Nous ne présentons les règles du système S_{QBF} que pour le fragment $\{\rightarrow, \neg, \top\}$. Pour simplifier, dans la description de la prémisse et de la conclusion de la règle, seuls le lieur et les classes pertinentes sont notées, les classes invariantes ne sont pas décrites (la classe $[\top]$ est implicitement toujours présente).

Définition 27 (système S_{QBF} pour $\{\rightarrow,\neg,\top\}$) *Le système S_{QBF} est constitué d'une règle d'élimination des doubles négations ($F \in \mathbf{PROP}_{\{\rightarrow,\neg,\top\}}$) :*

$$\neg\neg : \quad \frac{(Q\ ;\ [F]{=}[\neg\neg F])}{(Q\ ;\ [F],[\neg\neg F])}$$

de quatre règles d'élimination du quantificateur existentiel ($x \in \mathcal{SP}$) :

$$\exists\top : \quad \frac{(Q\ ;\ [x]{=}[\top])}{(\exists xQ\ ;\ [x])} \qquad \exists\bot : \quad \frac{(Q\ ;\ [\overline{x}]{=}[\top])}{(\exists xQ\ ;\ [\overline{x}])}$$

$$\exists+ : \quad \frac{(QQ'\ ;\ [x]{=}[\top])}{(Q\exists xQ'\ ;\ [x]{=}[\top])} \qquad \exists- : \quad \frac{(QQ'\ ;\ [\overline{x}]{=}[\top])}{(Q\exists xQ'\ ;\ [\overline{x}]{=}[\top])}$$

d'une règle d'élimination du quantificateur universel ($x \in \mathcal{SP}$) :

$$\forall : \quad \frac{(Q\ ;\ [x]{=}[\top]) \quad (Q\ ;\ [\overline{x}]{=}[\top])}{(\forall xQ\ ;\ [x])}$$

et neuf règles de fusion des classes ($F_Y, F_X \in \mathbf{PROP}_{\{\rightarrow,\neg,\top\}}$) :

$$1 : \quad \frac{(Q\ ;\ [\overline{F_X}]{=}[\top],[\overline{F_Y}]{=}[\top])}{(Q\ ;\ [(F_X{\rightarrow}F_Y)]{=}[\overline{F_Y}])} \qquad 2 : \quad \frac{(Q\ ;\ [F_X]{=}[\top],[F_Y]{=}[\top])}{(Q\ ;\ [(F_X{\rightarrow}F_Y)]{=}[\overline{F_X}])}$$

$$3 : \quad \frac{(Q\ ;\ [F_X]{=}[\top],[\overline{F_Y}]{=}[\top])}{(Q\ ;\ [\overline{(F_X{\rightarrow}F_Y)}]{=}[\top])} \qquad 4 : \quad \frac{(Q\ ;\ [(F_X{\rightarrow}F_Y)]{=}[\overline{F_X}])}{(Q\ ;\ [(F_X{\rightarrow}F_Y)],[\overline{F_Y}]{=}[\top])}$$

$$5 : \quad \frac{(Q\ ;\ [(F_X{\rightarrow}F_Y)]{=}[\overline{F_X}])}{(Q\ ;\ [(F_X{\rightarrow}F_Y)],[F_X]{=}[\overline{F_Y}])} \qquad 6 : \quad \frac{(Q\ ;\ [(F_X{\rightarrow}F_Y)]{=}[\top])}{(Q\ ;\ [(F_X{\rightarrow}F_Y)],[\overline{F_X}]{=}[\top])}$$

$$7 : \quad \frac{(Q\ ;\ [(F_X{\rightarrow}F_Y)]{=}[\top])}{(Q\ ;\ [(F_X{\rightarrow}F_Y)],[F_Y]{=}[\top])} \qquad 8 : \quad \frac{(Q\ ;\ [(F_X{\rightarrow}F_Y)]{=}[\top])}{(Q\ ;\ [(F_X{\rightarrow}F_Y)],[F_X]{=}[F_Y])}$$

$$9 : \frac{(Q\ ;\ [(F_X{\rightarrow}F_Y)]{=}[F_Y])}{(Q\ ;\ [(F_X{\rightarrow}F_Y)],[F_X]{=}[\top])}$$

Exemple 19 *En continuant l'exemple 18. Nous explicitons ici toutes les classes d'équivalences.*

$$\frac{\begin{array}{ll}(\forall a\exists d\forall e\exists f\ ; & [\top,\neg\neg\neg\nu,\neg\nu],[a],[d],[e],[f],[\nu_0],[\nu_1] \\ & [\neg\top,\neg\neg\nu,\nu],[\neg a],[\neg d],[\neg e],[\neg f],[\neg\nu_0],[\neg\nu_1])\end{array}}{\begin{array}{ll}(\forall a\exists d\forall e\exists f\ ; & [\top,\neg\neg\neg\nu],[a],[d],[e],[f],[\nu_0],[\nu_1],[\nu], \\ & [\neg\top,\neg\neg\nu],[\neg a],[\neg d],[\neg e],[\neg f],[\neg\nu_0],[\neg\nu_1],[\neg\nu])\end{array}}\ \neg\neg$$

Exemple 20 *La règle 9 de la définition précédente exprime que si la relation de formules R est telle que son lieur est Q et la partition est telle qu'une classe contienne une formule $(F_X{\rightarrow}F_Y)$ et que les formules F_X et $\top$ appartiennent à la même classe alors est inférée par la règle 9 une relation de formules $R([(F_X{\rightarrow}F_Y)] = [F_Y])$ (en particulier, cette nouvelle relation de formules est toujours telle que $[F_X] = [\top]$).*

Nous sommes à même de décrire ce qu'est un arbre de déduction et une preuve pour le système S_{QBF}.

Définition 28 (arbre de déduction) *Un* arbre de déduction *est défini inductivement ainsi :*

- *toute relation de formules qui est non explicitement contradictoire est un arbre de déduction pour le système S_{QBF} ;*
- *si R est une relation de formules, r une règle du système S_{QBF} telle que la prémisse soit satisfaite par R et que le résultat R', unique conclusion de l'application de la règle à la prémisse soit non explicitement contradictoire et si ∇' est un arbre de déduction de racine R' alors $\frac{\nabla'}{R}r$ est un arbre de déduction de racine R ;*
- *si R est une relation de formules, r une règle du système S_{QBF} telle que la prémisse soit satisfaite par R et que les résultats R' et R'', conclusions de l'application de la règle r à la prémisse soient non explicitement contradictoires et si ∇' et ∇'' sont des arbres de déduction de racine respectivement R' et R'' alors $\frac{\nabla' \quad \nabla''}{R}r$ est un arbre de déduction de racine R.*

Dans les exemples qui suivent, seule l'application des règles de fusion est indicée par le numéro de la règle.

Exemple 21 *En continuant l'exemple 19, l'arbre ci-dessous est un arbre de déduction pour le système S_{QBF} ($\nu = (\nu_0 \rightarrow \neg\nu_1)$, $\nu_0 = (e \rightarrow d)$ et $\nu_1 = (f \rightarrow a)$).*

$$\dfrac{\dfrac{\dfrac{\nabla \quad \nabla'}{(\forall a \exists d \forall e \exists f \ ; \ [\top, \neg\neg\neg\nu, \neg\nu, \nu_0, \nu_1], [a], [d], [e], [f])}}{(\forall a \exists d \forall e \exists f \ ; \ [\top, \neg\neg\neg\nu, \neg\nu], [a], [d], [e], [f], [\nu_0], [\nu_1])}\,{}^{3}}{Id_\xi([\neg\neg\neg\nu] = [\top])}$$

avec ∇ tel que

$$\dfrac{\dfrac{(\varepsilon \ ; \ [\top, \neg\neg\neg\nu, \neg\nu, \nu_0, \nu_1, a, d, e, f])}{(\exists f \ ; \ [\top, \neg\neg\neg\nu, \neg\nu, \nu_0, \nu_1, a, d, e], [f])} \quad \dfrac{(\varepsilon \ ; \ [\top, \neg\neg\neg\nu, \neg\nu, \nu_0, \nu_1, a, d, \neg e, f])}{(\exists f \ ; \ [\top, \neg\neg\neg\nu, \neg\nu, \nu_0, \nu_1, a, d, \neg e], [f])}}{\dfrac{(\forall e \exists f \ ; \ [\top, \neg\neg\neg\nu, \neg\nu, \nu_0, \nu_1, a, d], [e], [f])}{(\exists d \forall e \exists f \ ; \ [\top, \neg\neg\neg\nu, \neg\nu, \nu_0, \nu_1, a], [d], [e], [f])}}$$

avec ∇' tel que

$$\dfrac{\dfrac{\dfrac{(\exists f \ ; \ [\top, \neg\neg\neg\nu, \neg\nu, \nu_0, \nu_1, \neg a, \neg f, d, e]) \quad (\exists f \ ; \ [\top, \neg\neg\neg\nu, \neg\nu, \nu_0, \nu_1, \neg a, \neg f, d, \neg e])}{(\forall e \exists f \ ; \ [\top, \neg\neg\neg\nu, \neg\nu, \nu_0, \nu_1, \neg a, \neg f, d], [e])}}{(\exists d \forall e \exists f \ ; \ [\top, \neg\neg\neg\nu, \neg\nu, \nu_0, \nu_1, \neg a, \neg f], [d], [e])}}{(\exists d \forall e \exists f \ ; \ [\top, \neg\neg\neg\nu, \neg\nu, \nu_0, \nu_1, \neg a], [d], [e], [f])}\,{}^{4}$$

La relation de formules $(\exists f \ ; \ [\top, \neg\neg\neg\nu, \neg\nu, \nu_0, \nu_1, \neg a, \neg f, d, e])$ n'est pas explicitement contradictoire bien que a et e soient des symboles propositionnels universellement quantifiés car ils n'appartiennent plus au lieur qui est présentement $\exists f$.

Le fait d'avoir ramené dans la classe $\top$ un des deux littéraux pour tout symbole existentiel, ni même pour toute sous formule, ne signifie par pour autant qu'une telle relation de formules admette un modèle.

Définition 29 (axiome) *Un* axiome *pour une QBF prénexe close, QM, est une relation de formules $R = (\varepsilon\ ;\ P)$ non explicitement contradictoire (P une partition de $sfe(M)$) ne contenant que deux classes telles que $[M]_R = [\top]_R$ et telle qu'aucune règle de fusion ne puisse plus être appliquée.*

Nous sommes en mesure de définir ce qu'est une preuve pour une QBF dans le système S_{QBF}.

Définition 30 (preuve) *Un* arbre de preuve *(ou* preuve*) pour une QBF QM dans le système S_{QBF} est un arbre de déduction de racine $Id_{QM}([M] = [\top])$ tel que toute feuille soit un axiome.*

Exemple 22 *L'arbre de déduction de l'exemple 21 est une preuve dans le système S_{QBF} de la QBF $\forall a \exists d \forall e \exists f \neg\neg\neg((e \rightarrow d) \rightarrow \neg(f \rightarrow a))$.*

La règle d'élimination de la double négation est une règle qui peut être omise si la QBF à la racine ne contient pas de double négation. Les règles d'élimination $\exists\top$ et $\exists\bot$ expriment la sémantique du quantificateur existentiel, de même la règle d'élimination $\forall$ exprime la sémantique du quantificateur universel : en cela ce système n'est pas purement syntaxique. La règle d'élimination $\exists+$ (resp. $\exists-$) exprime que seule, lors de l'élimination du quantificateur existentiel, la règle $\exists\top$ (resp. $\exists\bot$) pourra être appliquée pour poursuivre le calcul ; les règles $\exists+$ et $\exists-$ peuvent être ôtées du système sans en altérer la complétude ; elles seront démontrées correctes en faisant appel à la sémantique ce qui est beaucoup plus simple que de passer par une démonstration dans la tradition de la théorie de la preuve par une manipulation des preuves. Les règles de fusion expriment les propriétés de l'implication ; le lien sémantique entre la prémisse et la conclusion des règles de fusion est un lien d'équivalence de préservation des modèles qui sera explicité dans ce qui suit.

La définition de l'axiome prévient la contradiction dans une classe. Cette définition se réduit uniquement à la première condition si l'utilisation de la règle d'élimination du quantificateur n'est utilisée que lorsque aucune autre règle ne peut plus l'être (car alors nécessairement la contradiction aurait été déduite des règles).

Exemple 23 *La relation de formules*

$$(\varepsilon\ ;\ [(a \rightarrow d), a, \overline{d}, \top], [\overline{(a \rightarrow d)}, \overline{a}, d, \overline{\top}])$$

(dont exceptionnellement toutes les classes ont été explicitées) n'est pas explicitement contradictoire mais n'est pas un axiome car :

$$\frac{(\varepsilon\ ;\ [(a \rightarrow d), a, \overline{d}, \top, \overline{(a \rightarrow d)}, \overline{a}, d, \overline{\top}])}{(\varepsilon\ ;\ [(a \rightarrow d), a, \overline{d}, \top], [\overline{(a \rightarrow d)}, \overline{a}, d, \overline{\top}])}\ 1$$

*Ceci interdit une preuve fausse pour la QBF prénexe $\exists a \exists d (a \rightarrow d)$ (en posant, d'un point de vue sémantique, $v(a) =$ **vrai** et $v(d) =$ **faux**) n'utilisant que la règle d'élimination du*

quantificateur existentiel telle que :

$$\dfrac{\dfrac{(\varepsilon\ ;\ [(a{\rightarrow}d), a, \overline{d}, \top], [\overline{(a{\rightarrow}d)}, \overline{a}, d, \overline{\top}])}{(\exists d\ ;\ [(a{\rightarrow}d), a, \top], [\overline{(a{\rightarrow}d)}, \overline{a}, \overline{\top}], [d], [\overline{d}])}}{(\exists a \exists d\ ;\ [(a{\rightarrow}d), \top], [\overline{(a{\rightarrow}d)}, \overline{\top}], [a], [\overline{a}], [d], [\overline{d}])}$$

La définition du système induit nullement l'utilisation des règles de fusion au plus tôt dans la preuve et permet en particulier des preuves « naïves » éliminant complètement les quantificateurs puis appliquant la règle de la double négation et celles de fusion.

1.4.3 Sémantique

Nous établissons les théorèmes de correction et complétude du système S_{QBF} par rapport à la sémantique des QBF, pour cela nous interprétons le système S_{QBF} dans la sémantique des QBF prénexes. En premier lieu, nous remarquons que tout axiome pour une QBF peut être aisément mis en correspondance avec un modèle propositionnel de la matrice de cette QBF (cf. Démonstrations, lemme 25).

Nous introduisons une fonction d'interprétation qui explicite la sémantique d'une relation de fonctions en une QBF.

Définition 31 (fonction d'interprétation) *La fonction d'interprétation d'une relation de formules* $(\cdot\ ;\ \cdot)^*$ *est une fonction de l'ensemble des relations de formules dans les QBF définie par*

$$(Q\ ;\ P)^* = Q \bigwedge_{C \in P} \Big(\bigwedge_{F, F' \in C} (F {\leftrightarrow} F') \Big).$$

Exemple 24 (fonction d'interpretation) *En continuant l'exemple 21*

$$\begin{array}{ll} & (\exists d \forall e \exists f\ ;\ [\top, \neg\neg\neg\nu, \neg\nu, \nu_0, \nu_1, \neg a, \neg f], [d], [e])^* \\ = & \exists d \forall e \exists f ((\bigwedge_{F, F' \in \{\top, \neg\neg\neg\nu, \neg\nu, \nu_0, \nu_1, \neg a, \neg f\}} (F {\leftrightarrow} F')) \wedge (d {\leftrightarrow} d) \wedge (e {\leftrightarrow} e)) \\ \cong & \exists d \forall e \exists f (\neg\nu \wedge \nu_0 \wedge \nu_1 \wedge \neg a \wedge \neg f) \end{array}$$

grâce à l'équivalence propositionnelle :

$$\bigwedge_{F, F' \in \{\top, \neg\neg\neg\nu, \neg\nu, \nu_0, \nu_1, \neg a, \neg f\}} (F {\leftrightarrow} F') \equiv \neg\nu \wedge \nu_0 \wedge \nu_1 \wedge \neg a \wedge \neg f$$

L'équivalence $(Q\ ;\ Id_{QM}([M] = [\top]))^* \cong QM$ dont la preuve est immédiate justifie que la racine de la preuve soit $Id_{QM}([M] = [\top])$ pour une QBF QM (cf. Démonstrations, lemme 26).

La correction du système S_{QBF} par rapport à la sémantique des QBF est une conséquence du fait que pour toute relation de formules R' le résultat de l'application d'une règle de fusion sur une relation de formules R est tel que $R'^* \cong R^*$ (cf. Démonstrations, lemme 27).

Il est immédiat qu'à partir du calcul de la validité directement par la définition de la sémantique des quantificateurs il est possible de construire une preuve dans le système S_{QBF} : la preuve, de bas en haut, élimine tous les quantificateurs puis fusionne les classes grâce aux règles de double négation et de fusions 4, 6, 7 et 9, d'où la complétude.

Théorème 1 (correction et complétude du système S_{QBF} par rapport à la sémantique des QBF) *Une QBF QM est valide si et seulement si la relation de formules $Id_{QM}([M] = [\top])$ admet une preuve dans le système S_{QBF}.*

Ce théorème de décision est étendu dans le paragraphe 1.4.5 à une version plus « forte » qui permet de construire n'importe quel modèle d'une QBF prénexe valide.

La plupart des systèmes pour décider des QBF prénexes sont basés sur l'algorithme de *Davis, Logemann et Loveland* (cf. chapitre 2) lui-même basé sur la *propagation de clauses unitaires* qui n'est autre que l'application des équivalences : $\exists x Q(x \wedge G) \overset{Q.2}{\equiv} Q[x \leftarrow \top](G)$ et $\exists x Q(\neg x \wedge G) \overset{Q.3}{\equiv} Q[x \leftarrow \bot](G)$, et sur la sémantique des quantificateurs. Par là même, le DLL ne construit que deux classes d'équivalence au lieu de la relation de formules. Cette dernière introduit, d'un point de vue théorie de la preuve, l'utilisation de la *règle de la coupure* mais avec une propriété de sous-formule (coupure « analytique ») qui permet de ne pas avoir à deviner une nouvelle formule (coupure « non-analytique ») pour pouvoir l'appliquer ; cette règle permet de réduire dans le meilleur des cas exponentiellement la taille des preuves [86]. Dans [202] une hiérarchie de complexité, basée sur le nombre nécessaire d'utilisations de la sémantique du quantificateur universel, est introduite. En pratique, et pour $TAUT$, les problèmes industriels ne dépasse guère le niveau deux de complexité. Ce résultat s'explique par la totale liberté sur l'ordre des quantificateurs, puisque tous universels, en application de l'équivalence logique $(\forall x\ (\forall y\ F)) \overset{1.2}{\equiv} (\forall y\ (\forall x\ F))$; il n'en va pas de même avec les QBF puisque les quantificateurs existentiels et universels ne permutent généralement pas.

1.4.4 Une extension au système S_{QBF}

Le système S_{QBF} présenté jusqu'ici est assez pauvre : il n'élimine les quantificateurs qu'en employant explicitement leur sémantique et ne tire aucun avantage des propriétés algébriques des quantificateurs entre-eux. L'extension proposée ci-dessous y pallie en partie en rajoutant de nouvelles règles de fusion ; l'extension préserve la correction, la complétude l'étant puisque nous ne faisons qu'étendre le système de règles.

Définition 32 (système S_{QBF} enrichi pour le $\{\rightarrow, \neg, \top\}$) *Le système S_{QBF} enrichi est constitué des règles du système S_{QBF} augmenté des nouvelles règles de fusion suivantes :*

$$10 : \frac{(QQ'\forall x Q'' \ ; \ [y]=[\top])}{(Q\exists y Q'\forall x Q'' \ ; \ [(x\rightarrow y)]=[\top])} \qquad 11 : \frac{(QQ'\forall x Q'' \ ; \ [y]=[\top])}{(Q\exists y Q'\forall x Q'' \ ; \ [(x\rightarrow y)]=[y])}$$

$$12 : \frac{(QQ'\forall y Q'' \ ; \ [x]=[\top])}{(Q\exists x Q'\forall y Q'' \ ; \ [(x\rightarrow y)]=[y])} \qquad 13 : \frac{(QQ'\forall x Q'' \ ; \ [\overline{y}]=[\top])}{(Q\exists y Q'\forall x Q'' \ ; \ [(x\rightarrow y)]=[\overline{x}])}$$

$$14 : \frac{(QQ'\forall y Q'' \ ; \ [\overline{x}]=[\top])}{(Q\exists x Q'\forall y Q'' \ ; \ [(x\rightarrow y)]=[\top])} \qquad 15 : \frac{(QQ'\forall y Q'' \ ; \ [\overline{x}]=[\top])}{(Q\exists x Q'\forall y Q'' \ ; \ [(x\rightarrow y)]=[\overline{x}])}$$

La définition 28 d'arbre de déduction est étendue en y intégrant les nouvelles règles de fusion. Les autres définitions portant sur l'axiome et la preuve ne sont pas modifiées.

Les règles enrichissant le système S_{QBF} dans la définition ci-dessus ne s'appliquent qu'aux symboles propositionnels et non aux sous-formules.

La construction du système S_{QBF} est orientée par la validité : la non-validité n'est exclue que par la définition 26 ; à l'instar des règles de fusion 10 à 15, les propriétés algébriques des quantificateurs vis-à-vis des connecteurs induisent des « règles » qui permettent de conclure que développer un arbre de déduction à partir d'une relation de formules est d'ors et déjà voué à l'échec ; la procédure de décision du chapitre 4 étend le système S_{QBF} en ce sens.

L'axiome choisit pour le système S_{QBF} est élémentaire puisqu'il réclame l'élimination complète du lieur ; la procédure de décision du chapitre 6 étend à nouveau le système S_{QBF} par le choix d'un axiome plus « efficace » : le lieur peut être non-vide si toutes les sous-formules, qui ne sont pas des littéraux, sont dans la classe de $\top$ ou celle de $\bot$.

Exemple 25 *En continuant l'exemple 21. Extrait de la preuve pour le système S_{QBF} enrichi (avec $\nu_0 = (e \rightarrow d)$) :*

$$\dfrac{\dfrac{(\forall a \forall e \exists f \ ; \ [\top, \neg\neg\neg\nu, \neg\nu, \nu_0, \nu_1, d], [a], [e], [f])}{(\forall a \exists d \forall e \exists f \ ; \ [\top, \neg\neg\neg\nu, \neg\nu, \nu_0, \nu_1], [a], [d], [e], [f])}\ {}_{3}}{Id_{\xi}([\neg\neg\neg\nu] = [\top])}\ {}^{10}$$

Nous obtenons, à l'instar du théorème 1, la correction et complétude pour le système S_{QBF} enrichi simplement en démontrant pour les nouvelles règles que les interprétations de la prémisse et de conclusion sont toujours équivalentes au sens de la préservation des modèles QBF (cf. Démonstrations, lemme 28).

1.4.5 Calcul de modèle dans le système S_{QBF} enrichi

Nous annotons notre système S_{QBF} pour les QBF prénexes en y ajoutant la construction d'un modèle QBF : nos règles opèrent maintenant sur un couple $V \Rightarrow (Q \ ; \ P)$, $(Q \ ; \ P)$ une relation de formules et V une valuation QBF de QM, avec P une partition de $sfe(M)$. Nous étendons les notions d'axiome, de dérivation et de preuve pour un nouveau système S^a_{QBF}, extension annotée du système S_{QBF} enrichi.

Définition 33 (système S^a_{QBF} pour le $\{\rightarrow, \neg, \top\}$) *Le système S^a_{QBF} est constitué d'une règle d'élimination des doubles négations ($F \in \mathbf{PROP}_{\{\rightarrow,\neg,\top\}}$) :*

$$\neg\neg : \quad \frac{V \Rightarrow (Q \ ; \ [F] = [\neg\neg F])}{V \Rightarrow (Q \ ; \ [F], [\neg\neg F])}$$

de quatre règles d'élimination du quantificateur existentiel (dans les règles $\exists +^a$ et $\exists -^a$, les fonctions $\hat{x}$ sont d'arité le nombre de quantificateurs universels du lieur Q.) :

$$\exists \top^a : \quad \frac{(v[x{:=}\mathbf{vrai}], sk) \Rightarrow (Q \ ; \ [x] = [\top])}{(v, \{(x \mapsto \mathbf{vrai})\} \cup sk) \Rightarrow (\exists x Q \ ; \ [x])} \qquad \exists \bot^a : \quad \frac{(v[x{:=}\mathbf{faux}], sk) \Rightarrow (Q \ ; \ [\overline{x}] = [\top])}{(v, \{(x \mapsto \mathbf{faux})\} \cup sk) \Rightarrow (\exists x Q \ ; \ [x])}$$

$$\exists +^a : \quad \frac{(v[x{:=}\mathbf{vrai}], sk) \Rightarrow (QQ' \ ; \ [x] = [\top])}{(v, \{(x \mapsto \mathbf{vrai})\} \cup sk) \Rightarrow (Q \exists x Q' \ ; \ [x] = [\top])} \qquad \exists -^a : \quad \frac{(v[x{:=}\mathbf{faux}], sk) \Rightarrow (QQ' \ ; \ [\overline{x}] = [\top])}{(v, \{(x \mapsto \mathbf{faux})\} \cup sk) \Rightarrow (Q \exists x Q' \ ; \ [\overline{x}] = [\top])}$$

d'une règle d'élimination du quantificateur universel :

$$\forall^a : \frac{(v[x{:=}\mathbf{vrai}],sk(\mathbf{vrai}))\Rightarrow(Q\ ;\ [x]{=}[\top])\quad (v[x{:=}\mathbf{faux}],sk(\mathbf{faux}))\Rightarrow(Q\ ;\ [\overline{x}]{=}[\top])}{(v,sk)\Rightarrow(\forall xQ\ ;\ [x])}$$

et chaque règle de fusion de 1 à 9 de la définition 27 de structure $\frac{R'}{R}$ *est augmentée en une règle* $\frac{V\Rightarrow R'}{V\Rightarrow R}$.

Les règles de fusion 10 à 15 du système S_{QBF} *enrichi sont étendues ainsi :*

$$10^a : \frac{(v[y{:=}\mathbf{vrai}],sk)\Rightarrow(QQ'\forall xQ''\ ;\ [y]{=}[\top])}{(v,\{(y{\mapsto}\mathbf{vrai})\}{\cup}sk)\Rightarrow(Q\exists yQ'\forall xQ''\ ;\ [(x{\rightarrow}y)]{=}[\top])}$$

$$11^a : \frac{(v[y{:=}\mathbf{vrai}],sk)\Rightarrow(QQ'\forall xQ''\ ;\ [y]{=}[\top])}{(v,\{(y{\mapsto}\mathbf{vrai})\}{\cup}sk)\Rightarrow(Q\exists yQ'\forall xQ''\ ;\ [(x{\rightarrow}y)]{=}[y])}$$

$$12^a : \frac{(v[x{:=}\mathbf{vrai}],sk)\Rightarrow(QQ'\forall yQ''\ ;\ [x]{=}[\top])}{(v,\{(x{\mapsto}\mathbf{vrai})\}{\cup}sk)\Rightarrow(Q\exists xQ'\forall yQ''\ ;\ [(x{\rightarrow}y)]{=}[y])}$$

$$13^a : \frac{(v[y{:=}\mathbf{faux}],sk)\Rightarrow(QQ'\forall xQ''\ ;\ [\overline{y}]{=}[\top])}{(v,\{(y{\mapsto}\mathbf{faux})\}{\cup}sk)\Rightarrow(Q\exists yQ'\forall xQ''\ ;\ [(x{\rightarrow}y)]{=}[\overline{x}])}$$

$$14^a : \frac{(v[x{:=}\mathbf{faux}],sk)\Rightarrow(QQ'\forall yQ''\ ;\ [\overline{x}]{=}[\top])}{(v,\{(x{\mapsto}\mathbf{faux})\}{\cup}sk)\Rightarrow(Q\exists xQ'\forall yQ''\ ;\ [(x{\rightarrow}y)]{=}[\top])}$$

$$15^a : \frac{(v[x{:=}\mathbf{faux}],sk)\Rightarrow(QQ'\forall yQ''\ ;\ [\overline{x}]{=}[\top])}{(v,\{(x{\mapsto}\mathbf{faux})\}{\cup}sk)\Rightarrow(Q\exists xQ'\forall yQ''\ ;\ [(x{\rightarrow}y)]{=}[\overline{x}])}$$

Un axiome *pour une QBF* QM *dans le système* S^a_{QBF} *est un couple* $V \Rightarrow R$ *constitué d'une relation de formules* $R = (\varepsilon\ ;\ P)$, *non explicitement contradictoire ne contenant que deux classes telles que* $[M]_R = [\top]_R$ *et telle qu'aucune règle de fusion ne puisse plus être appliquée, et d'une valuation QBF,* $V = (v, \emptyset)$, *telle que pour tout symbole propositionnel* x, $v(x) = \mathbf{vrai}$ *si et seulement si* $x \in [\top]$ *et* $v(x) = \mathbf{faux}$ *si et seulement si* $x \in [\overline{\top}]$.

Si la preuve se construit de bas en haut pour la recherche des axiomes, la construction du modèle QBF se réalise de haut en bas.

Exemple 26 *En continuant l'exemple 21 et pour la valuation QBF de composante fonctionnelle* $sk = \{(d \mapsto \hat{d}), (f \mapsto \hat{f})\}$ *(et de composante propositionnelle* v *quelconque) avec*

$$\begin{array}{l}\hat{d} = \{(\mathbf{vrai} \mapsto \mathbf{vrai}), (\mathbf{faux} \mapsto \mathbf{vrai})\} = \mathbf{vrai}\ \textit{et}\\ \hat{f} = \{((\mathbf{vrai}, \mathbf{vrai}) \mapsto \mathbf{vrai}), ((\mathbf{vrai}, \mathbf{faux}) \mapsto \mathbf{vrai}),\\ \quad ((\mathbf{faux}, \mathbf{vrai}) \mapsto \mathbf{faux}), ((\mathbf{faux}, \mathbf{faux}) \mapsto \mathbf{faux})\}\end{array}$$

Extrait de la preuve pour le système S^a_{QBF} *avec calcul du modèle (*$\nu = (\nu_0{\rightarrow}\neg\nu_1)$, $\nu_0 = (e{\rightarrow}d)$ *et* $\nu_1 = (f{\rightarrow}a)$*) :*

$$\dfrac{\dfrac{\dfrac{\nabla_{\mathbf{vrai}}\quad \nabla_{\mathbf{faux}}}{(v[d := \mathbf{vrai}], \{(f \mapsto \hat{f})\}) \Rightarrow (\forall a \forall e \exists f\ ;\ [\top, \neg\neg\neg\nu, \neg\nu, \nu_0, \nu_1, d], [a], [e], [f])}\,{}_{10}}{(v, sk) \Rightarrow (\forall a \exists d \forall e \exists f\ ;\ [\top, \neg\neg\neg\nu, \neg\nu, \nu_0, \nu_1], [a], [d], [e], [f])}}{(v, sk) \Rightarrow (\forall a \exists d \forall e \exists f\ ;\ [\top, \neg\neg\neg\nu, \neg\nu], [\nu_0], [\nu_1], [a], [d], [e], [f])}\,{}_{3}}{(v, sk) \Rightarrow Id_\xi([\neg\neg\neg\nu] = [\top])}$$

avec $\nabla_{\mathbf{vrai}}$ *de racine*

$$(v[d := \mathbf{vrai}][a := \mathbf{vrai}], \{(f \mapsto \hat{f}(\mathbf{vrai}))\})$$
$$\Rightarrow \quad (\forall e \exists f \ ; \ [\top, \neg\neg\neg\nu, \neg\nu, \nu_0, \nu_1, d, a], [e], [f])$$

et $\nabla_{\mathbf{faux}}$ *de racine*

$$(v[d := \mathbf{vrai}][a := \mathbf{faux}], \{(f \mapsto \hat{f}(\mathbf{faux}))\})$$
$$\Rightarrow \quad (\forall e \exists f \ ; \ [\top, \neg\neg\neg\nu, \neg\nu, \nu_0, \nu_1, d, \neg a], [e], [f])$$

Nous obtenons, à l'instar du théorème 1 et de la correction et complétude du système S_{QBF} enrichi, la correction et complétude pour le système S^a_{QBF} comme corollaire (cf. Démonstrations, lemme 29) ; ce résultat est plus fort que les deux précédents puisque non seulement il décide du problème de validité mais il permet de construire un modèle QBF pour toute QBF prénexe close valide.

1.5 Algorithmique des QBF

L'algorithmique des QBF s'appuient principalement sur des règles de simplification issues des équivalences préservant les modèles propositionnels (cf. § 1.3.1) et des *équivalences logiques* propositionnelles, et pour le traitement des quantificateurs sur les deux équivalences suivantes qui étendent la sémantique attendue (F et G deux QBF, x et y deux symboles propositionnels, y ayant une occurrence libre dans G) :

$$[y \leftarrow (\exists x \ F)](G) \equiv [y \leftarrow ([x \leftarrow \top](F) \vee [x \leftarrow \bot](F))](G) \tag{1.1}$$

et

$$[y \leftarrow (\forall x \ F)](G) \equiv [y \leftarrow ([x \leftarrow \top](F) \wedge [x \leftarrow \bot](F))](G) \tag{1.2}$$

et qui expriment qu'un quelconque quantificateur peut être éliminé par son *expansion*, respectivement, soit existentielle, soit universelle. La formule F (et ses quantificateurs) est dupliquée et les symboles propositionnels associés à ses quantificateurs doivent être renommés dans une des deux parties de la nouvelle formule si elle doit être mise sous forme prénexe.

Deux versions radicales s'offrent à nous : la version « algorithme de recherche » qui applique les équivalences du quantificateur le plus externe vers le plus interne (ou encore « expansion top-down ») et la version « élimination de quantificateurs » qui applique les équivalences du quantificateur le plus interne vers le plus externe (ou encore « expansion bottom-up »).

Le problème de décision pour les QBF est complet pour la classe de complexité $PSPACE$ ce qui signifie que l'espace nécessaire pour résoudre une QBF est au pire des cas polynomial par rapport à la taille de la QBF. A moins que $\mathcal{P} = \mathcal{NP}$, il n'existe pas d'algorithme polynomial en temps par rapport à la taille d'une QBF quelconque pour décider de la validité celle-ci [172, 88].

Algorithme 2 *recherche_prenexe*

Entrée: Q : le lieur d'une QBF
Entrée: M : la matrice d'une QBF
Sortie: une valuation fonctionnelle d'un modèle QBF ou *non_valide*

```
si Q = qx alors
 si q = ∃ alors
  selon M faire
   cas ⊤ : retourner {(x ↦ vrai)}
   cas ⊥ : retourner non_valide
   cas x : retourner {(x ↦ vrai)}
   cas ¬x : retourner {(x ↦ faux)}
  fin selon
 sinon
  si M ≡ ⊤ alors
   retourner ∅
  sinon
   retourner non_valide
  fin si
 fin si
sinon
 Q = qxQ'
 π^⊤ := recherche_prenexe(Q', [x ← ⊤](M))
 si π^⊤ = non_valide alors
  si q = ∃ alors
   π^⊥ := recherche_prenexe(Q', [x ← ⊥](M))
   si π^⊥ = non_valide alors retourner non_valide
   sinon retourner {(x ↦ faux)} ∪ π^⊥ fin si
  sinon
   retourner non_valide
  fin si
 sinon
  si q = ∃ alors
   retourner {(x ↦ vrai)} ∪ π^⊤
  sinon
   π^⊥ := recherche_prenexe(Q', [x ← ⊥](M))
   si π^⊥ = non_valide alors
    retourner non_valide
   sinon
    retourner {(y ↦ ŷ) | ŷ(vrai) = ŷ^⊤, ŷ^⊤ ∈ π^⊤, ŷ(faux) = ŷ^⊥, ŷ^⊥ ∈ π^⊥}
   fin si
  fin si
 fin si
fin si
```

1.5.1 Recherche pour les QBF prénexes

Nous nous focalisons pour l'algorithme de recherche sur le cas des QBF prénexes closes car elles seront au cœur des chapitres suivants. La QBF G dans les équivalences ci-dessus est réduite à y et le caractère $PSPACE$ est assuré par une pile de *retour-arrière* qui stocke l'état de $[x \leftarrow \bot](F)$ pendant que $[x \leftarrow \top](F)$ est traité. L'algorithme 1.5.1 *recherche_prenexe* réalise cette recherche avec $(\exists x\ F) = QM$ et calcule une valuation fonctionnelle d'un modèle QBF, la pile de retour-arrière étant implicite : si le lieur est réduit à un unique quantificateur associé à un symbole propositionnel alors si c'est un quantificateur existentiel quatre cas sont possibles, correspondant dans l'ordre de l'algorithme à : $\exists x \top \equiv \top$ alors, arbitrairement, la valuation fonctionnelle $\{(x \mapsto \textbf{vrai})\}$ est retournée[7] $\exists x \bot \equiv \bot$ alors *non_valide* est retourné, $\exists x x \equiv \top$ alors l'unique valuation fonctionnelle $\{(x \mapsto \textbf{vrai})\}$ est retournée et $\exists x \neg x \equiv \top$ alors l'unique valuation fonctionnelle $\{(x \mapsto \textbf{faux})\}$ est retournée, sinon le quantificateur est universel alors si $M \equiv \top$ alors $\forall x M \equiv \top$ et la valuation fonctionnelle vide est retournée sinon $\forall x x \equiv \forall x \neg x \equiv \forall x \bot \equiv \bot$ et *non_valide* est retourné ; s'il y a plus d'un quantificateur, puisque l'algorithme est un algorithme de recherche, le quantificateur le plus externe est considéré en premier, si ce quantificateur est existentiel alors si un des deux appels récursifs pour la substitution par $\top$ (resp. par $\bot$) sur le symbole propositionnel x retourne un résultat différent de *non_valide* (la valuation fonctionnelle $\pi^{\top}$, resp. $\pi^{\bot}$) alors la QBF est valide et la valuation fonctionnelle $\{(x \mapsto \textbf{vrai})\} \cup \pi^{\top}$, resp. $\{(x \mapsto \textbf{faux})\} \cup \pi^{\bot}$ est retournée, si le quantificateur est universel alors si au moins un des appels récursifs pour les substitutions par $\top$ ou $\bot$ pour le symbole propositionnel x retourne *non_valide* alors la QBF est non-valide et *non_valide* est retourné sinon la QBF est valide et la valuation fonctionnelle sk telle que pour chaque symbole propositionnel y, $(y \mapsto \hat{y}^{\top}) \in \pi^{\top}$, $(y \mapsto \hat{y}^{\bot}) \in \pi^{\bot}$, $(y \mapsto \hat{y}) \in sk$, $\hat{y}(\textbf{vrai}) = \hat{y}^{\top}$ et $\hat{y}(\textbf{faux}) = \hat{y}^{\bot}$, est retournée.

Dans l'algorithme, il est supposé que la substitution par $\top$ ou $\bot$ s'accompagne de simplifications qui les éliminent. Le cas d'arrêt peut être simplifié en considérant que la récursivité s'arrête sur un lieur vide, dans ce cas l'algorithme retourne la valuation fonctionnelle vide, si la matrice M est égal à $\top$ et *non_valide* sinon.

Dans le cas de QBF prénexes dont la matrice est sous FNC, $[x \leftarrow \bot](F)$ revient à ôter toutes les clauses contenant $\neg x$ et ôter les occurrences de x restantes ; $[x \leftarrow \top](F)$ revient à ôter toutes les clauses contenant x et ôter les occurrences de $\neg x$ restantes dans la formule ; l'algorithme `QDLL`, extension pour les QBF de l'algorithme de *Davis, Logemann et Loveland* qui est « la » procédure de décision pour le problème SAT pour des formules propositionnelles sous FNC, adjoint à l'algorithme de recherche la *propagation de clauses unitaires* et la *propagation de littéraux monotones*.

1.5.2 Élimination de quantificateurs pour des QBF prénexes sous FNC

Nous nous focalisons pour l'algorithme d'élimination des quantificateurs sur le cas des QBF prénexes closes sous FNC car elles rendent l'algorithme plus aisé à présenter. La QBF

[7] deux modèles QBF sont possibles : l'un de valuation fonctionnelle $\{(x \mapsto \textbf{vrai})\}$ et l'autre, $\{(x \mapsto \textbf{faux})\}$

G dans les équivalences (1.1) et (1.2) est réduite à Qy et la QBF F à qxM pour une QBF prénexe close $QqxM$. Si nous considérons la matrice M sous FNC et que M_+ représente l'ensemble des clauses de M contenant le symbole propositionnel x sous sa forme de littéral positif, M_- représente l'ensemble des clauses de M contenant x sous sa forme de littéral négatif et M' représente l'ensemble des clauses de M ne contenant pas x alors M peut être réécrite en $((x \vee M_+) \wedge (\neg x \vee M_-) \wedge M')$ et (cf. Démonstrations, lemme 30)

$$Q\exists x((x \vee M_+) \wedge (\neg x \vee M_-) \wedge M') \equiv Q((M_- \vee M_+) \wedge M')$$

et

$$Q\forall x((x \vee M_+) \wedge (\neg x \vee M_-) \wedge M') \equiv Q((M_- \wedge M_+) \wedge M').$$

Il est particulièrement remarquable que $Q((M_- \vee M_+) \wedge M')$ ne soit plus, en général, sous FNC et nécessite l'utilisation de la distributivité $(F \vee (G \wedge H)) \stackrel{0.24}{\equiv} ((F \vee G) \wedge (F \vee H))$ pour obtenir une FNC induisant une croissance exponentielle dans le pire des cas de la taille de la QBF (le problème demeure $PSPACE$-complet mais l'algorithme, lui, n'est pas polynomial en espace) ; tandis que la QBF $Q((M_- \wedge M_+) \wedge M')$ est sous FNC.

L'algorithme 1.5.2 *elimination_prenexe_FNC* réalise cette élimination de quantificateurs par l'appel *elimination_prenexe_FNC*(Q, M) pour une QBF prénexe close sous FNC QM. La fonction fnc prend en entrée une formule propositionnelle et en calcule une FNC par la méthode « académique » (la méthode par « renommage de formules » ne pouvant être appliquée pour une question de terminaison).

Cet algorithme est l'extension pour les QBF prénexes de l'algorithme de *Davis et Putnam* qui est une procédure de décision pour le problème SAT pour des formules propositionnelles sous FNC : seul le cas pour le quantificateur universel y est adjoint.

Algorithme 3 *elimination_prenexe_FNC*

Entrée: Q : le lieur d'une QBF sous FNC
Entrée: M : la matrice d'une QBF sous FNC
Sortie: *valide* ou *non_valide*

```
si Q = ε alors
  si M ≡ ⊤ alors retourner valide sinon retourner non_valide
sinon
  Q = Q'q
  M = ((x∨M+)∧(¬x∨M-)∧M')
  si q = ∃ alors
    retourner elimination_prenexe_FNC(Q', (fnc((M-∨M+))∧M'))
  sinon
    retourner elimination_prenexe_FNC(Q', ((M-∧M+)∧M'))
  fin si
fin si
```

1.5.3 Certificat pour le problème de validité des QBF prénexes

Il y a loin de la coupe aux lèvres entre l'algorithme correct (et complet) pour le problème de validité, quel qu'il soit, et une implantation introduisant de nombreuses optimisations ; les compétitions mettant en concurrence des implantations de procédures de décision pour le problème de validité des QBF montrent des désaccords sur certaines instances [152, 153]. Un *certificat* est une information qui permet de certifier *a posteriori* que la preuve de validité ou de non-validité est correcte. Dans le cas du problème SAT, l'information construite au fur et à mesure de la preuve qui certifie de la satisfiabilité d'une formule propositionnelle peut être un *modèle* propositionnel et de l'*insatisfiabilité*, une réfutation dans le cadre de la *résolution*. Dans le cas du problème de validité des QBF prénexes, la non-validité peut être certifiée par la construction d'une réfutation grâce à la *Q-résolution* ; la validité peut l'être soit par un modèle QBF représenté explicitement par une *politique*, soit par un ensemble de modèles QBF implicitement calculé par l'exécution et représenté par un ***sat**-certificat* [25], soit par des formalismes reflétant le fonctionnement opérationnel *ad hoc* à des procédures de décision spécifiques [118].

Un **sat**-certificat pour une QBF prénexe QM, avec $y_1, \ldots, y_p$ ses symboles propositionnels existentiellement quantifiés, est une séquence de paires de formules $(\phi_1, \nu_1); \ldots ; (\phi_p, \nu_p)$, ϕ_i et ν_i associées au symbole propositionnel y_i et définies sur les symboles propositionnels universellement quantifiés de QM précédant dans le lieur le symbole y_i, $1 \leq i \leq p$. Cela est défini dans [25] seulement pour des QBF sous FNC avec des séquences de paires de *BDD*. Un **sat**-certificat est *consistant* si pour tout i, $1 \leq i \leq p$, $(\phi_i \wedge \nu_i) \equiv \bot$. Si un **sat**-certificat est consistant alors $\phi_1; \ldots ; \phi_p$ et $\neg\nu_1; \ldots ; \neg\nu_p$ sont deux séquences de formules associées aux fonctions de la composante fonctionnelle de deux ensembles de modèles QBF de la QBF certifiée (et pas nécessairement les mêmes). Pour certifier de la validité d'une QBF sous FNC QM avec un **sat**-certificat $(\phi_1, \nu_1); \ldots ; (\phi_p, \nu_p)$, il suffit de vérifier si $[\neg x_1 \leftarrow \nu_1][x_1 \leftarrow \phi_1] \ldots [\neg x_p \leftarrow \nu_p][x_p \leftarrow \phi_p](M)$ est une *tautologie* ; si la vérification échoue alors soit la QBF est non-valide soit le **sat**-certificat est incorrect ; réciproquement, si la vérification est un succès alors la QBF est valide et le **sat**-certificat encode un ensemble de modèles QBF. Bien que le **sat**-certificat soit une notion opérationnelle, elle est indépendante de la procédure de décision et n'est pas lié à la représentation de la QBF mais seulement à sa sémantique ; présentée dans le cas des QBF sous FNC et pour la *skolémisation propositionnelle* [25], nous étendons la portée aux QBF prénexes dans le cas des algorithmes de recherche (cf § 3.2).

Chapitre 2

Un panorama sur le problème de validité des QBF

Il y eut un soir, il y eut un matin : deuxième jour.

Sommaire

Nous désirons dresser dans ce chapitre un panorama des travaux sur le problème de validité des QBF d'un point de vue de l'algorithmique *séquentielle*. Nous concluons ce chapitre par un état de l'art de l'algorithmique parallèle pour le problème de validité des QBF et établissons le lien en conclusion générale avec les travaux de thèse de Benoit Da Mota. Le but de l'ensemble des *procédures de décision* décrites dans cette étude est d'échapper, autant que faire se peut, au caractère exponentiel de ce problème. Pourquoi autant d'effort et ne pas s'abandonner à la « loi de Moore » qui prophétise que le nombre de transistors par circuit de même taille double tous les dix-huit mois ? « Si le programme consomme une ressource de taille exponentielle selon la taille de la tâche, $O(k^n)$, alors pour chaque doublement des ressources accessibles, le gain n'est qu'en $(\ln 2 / \ln k)$ mots (...) traitables. » [141] Ainsi la recherche algorithmique a de belles heures devant elle ; mais pourquoi ne pas se concentrer sur le problème SAT et ramener le problème de validité des QBF prénexe à celui-ci par une utilisation de la sémantique des quantificateurs universels jusqu'à leur élimination complète ? Ce processus est malheureusement exponentiel dans le pire des cas ce qui est conforme à l'esprit de la *hiérarchie polynomiale* et à l'idée que plus la QBF est dans un fragment avec une grande alternance de quantificateurs plus le problème de validité qui lui est associé est complexe.

Nous proposons une tentative de « phylogénie » qui nous paraît nécessaire pour remettre en perspective l'ensemble des travaux présentés dans les chapitres suivants. Elle est aussi utile pour qui voudrait avoir une idée des outils théoriques dont il dispose pour définir une procédure de décision pour le problème de validité des QBF et qu'il doit implanter pour développer un système efficace. Pour se faire nous allons définir un *code génétique* très élémentaire des différentes procédures de décision. Dans la seconde partie, le « Glossaire des formules booléennes quantifiées », ce code génétique marquera chaque procédure de décision. Nombre des techniques qui nous permettront de dépeindre le décor sont compatibles entre-elles et une hiérarchisation est un exercice périlleux. Nous proposons six gènes que nous décrirons dans l'ordre décroissant du caractère, de notre point de vue, fondamental pour les classer. Cela ne signifie pas que les derniers gènes présentés nous semblent moins importants, bien au contraire comme en témoigne les thèmes des différents chapitres abordés, mais cet ordre d'introduction des caractéristiques des procédures de décision permet une classification « naturelle » en ce qu'il offre une grille de lecture qui nous apparaît adaptée à la compréhension du domaine. Nous introduisons aussi trois « espèces » de procédures de décision qui possèdent non seulement les six gènes communs mais aussi des gènes propres. La forme générique du génome et l'ensemble des allèles sont rassemblés dans la table 2.1.

Nous intégrons dans notre étude les procédures suivantes classées chronologiquement : `QKN` [120], `Evaluate` [46], `decide` [187], `QSOLVE` [84], `QUBE` [103], `Semprop` [130], `QSAT` [176], `Quaffle` [230, 228], `QUBOS` [17], `WalkQSAT` [95], `quantor` [36], `QMRES` [171], `QBDD` [171], `ZQSAT` [98], `CLearn` [97], `sKizzo` [23], `S-QBF` [192], `Q-PREZ` [49], `2clsQ` [193], `qpro` [79], `IQTest` [227], `Duaffle` [190], `AB` [28], `QuBIS`[181], `Nenofex` [136], `AIGsolve` [175] ainsi que `CirQit` [112]. Toutes ces procédures font l'objet d'articles plus ou moins détaillés dans le glossaire. D'autres procédures existent mais elles ne sont pas décrites suffisamment dans la littérature pour être intégrées dans cette étude [152, 153]. La procédure de décision `AQME` [180] échappe résolument à cette étude puisque c'est une « meta »-procédure

Nom de la procédure

Procédure :	*Complexité :*
Incomplet	$PSPACE$
Décision	$\mathcal{NP}/co-\mathcal{NP}$

Sémantique ou symbolique :	*Complexité réelle en espace :*	*En entrée :*
Sémantique	Polynomial en espace	FNC (prénexe)
Symbolique	Exponentiel en espace	FNC/FND (prénexe)
		FNN (prénexe)
		QBF (prénexe)

Une des trois espèces ci-dessous et les gènes qui lui sont propres

Monolithique

Elimination des existentielles :	*Traitement des connecteurs :*
$\exists$-expansion$\downarrow$	Résolution
$\exists$-backtrack	Multi-résolution
$\exists$-backjumping	Elimination des connecteurs
$\exists$-learning	Propagation unitaire
$\exists$-expansion$\uparrow$	Hyper résolution binaire
Eliminationdes universelles :	Littéraux monotones
$\forall$-expansion$\downarrow$	Réduction universelle
$\forall$-backtrack	Elimination des littéraux unitaires
$\forall$-backjumping	Propagation don't care
$\forall$-expansion$\uparrow$	Simplification $\top$ et $\bot$
Abstract Branching	Propagation

Représentation interne :
Les allèles *en entrée* plus
FNCA (prénexe)
DAG

À Oracle

Complexité de l'oracle :
$\mathcal{NP}$
$co-\mathcal{NP}$
Utilisation de l'oracle :
Simplification
Agrégation

Transformation

Technique :
$\exists$-expansion
$\forall$-expansion
Skolémisation
Langage :
SAT
ASP

Tab. 2.1 – Forme générique du génome et ensemble des allèles

qui fait appel à différentes procédures présentées ci-dessous en choisissant, par une (série d')heuristique(s), celle considérée comme la mieux adaptée.

2.1 Six gènes communs

Premier gène. Le premier gène détermine si la procédure considérée est bien une procédure de décision ; le problème étant décidable, il est évident qu'il est préférable qu'il en soit ainsi. Néanmoins, et sous les hypothèses que $\mathcal{P} \neq \mathcal{NP}$ et que la *hiérarchie polynomiale* ne s'effondre pas sur elle-même à partir d'un certain niveau [88], la complexité exponentielle inhérente au test de validité d'une QBF rend illusoire la recherche d'une procédure de décision efficace ; cette voie est pourtant celle choisie par la majeure partie de la communauté, le thème premier de ce chapitre et plus ou moins celui de cet ouvrage ; nous marquons ce gène de l'allèle « Décision ». Une autre voie est possible : celle des procédures *incomplètes* ; nous marquons ce gène de l'allèle « Incomplet » ; elles peuvent l'être de deux manières : en posant des hypothèses sur lesquelles la procédure ne revient pas ou en ne parcourant qu'une partie de l'espace de recherche de manière « aléatoire ». Le premier cas relève typiquement des méthodes « gloutonnes » qui font des choix sur les valeurs des existentielles et s'y tiennent. Lorsqu'une telle procédure termine sur un échec, seul subsiste le constat que aucune solution n'a été mise en exergue ; la non-validité, dans le cas des QBF, n'est donc pas décidée. Le second cas relève des méthodes stochastiques et a été exploré avec succès pour le problème SAT, pour la *programmation par ensembles réponses* [92][1], pour la logique des défauts de Reiter [184][2]. Là encore, lorsqu'une procédure stochastique termine sur un échec, seul subsiste le constat que aucune solution n'a été mise en exergue ; la non-validité, dans le cas des QBF, n'est donc pas plus décidée. Seules trois procédures appliquent des méthodes incomplètes pour des QBF d'une complexité quelconque : la procédure `WalkQSAT` [95] qui est basée sur la recherche stochastique locale selon l'algorithme `WalkSAT` [200] et l'instance `QBDD(LS)` [14] pour la recherche locale du schéma `QBDD(X)` qui fait coopérer un générateur de modèles propositionnels sur le problème SAT et un constructeur de modèles QBF (la procédure décrite dans [115] est une recherche locale uniquement pour le fragment des QBF prénexes de lieur $\forall\exists$). Pour une méthode stochastique devant évaluer un modèle QBF potentiel, telle que `WalkQSAT`, la difficulté est que ce test est $co-\mathcal{NP}$*-complet* (cf. § 1.2.2) alors que pour la procédure `WalkSAT` celui-ci est polynomial en temps.

Deuxième gène. Le deuxième gène détermine si la procédure est complète vis-à-vis des QBF et de la classe de complexité $PSPACE$: toute QBF peut être traitée, au besoin

[1] aussi de complexité $\mathcal{NP}$-complet [143] lorsqu'il s'agit du calcul du *modèle stable* [91], l'auteur ayant participé activement à l'exploration de ce domaine [162, 165, 166] dans le cadre de l'optimisation par colonies de fourmis [39]

[2] de complexité Σ_2^p-complet [111], l'auteur ayant aussi participé activement à l'exploration de ce domaine dans [216, 161, 160, 159, 163, 164] dans le cadre de l'algorithmique génétique [147]), en combinaison avec de la recherche locale via la recherche tabou [110] dans [159, 163, 164] et enfin dans [162, 164] dans le cadre de l'optimisation par colonies de fourmis

à travers une série de transformations qui préservent la validité. Bien que cette question ne soit pas uniquement lié à une question de syntaxe, les classes de complexité sont caractérisées par des fragments syntaxiques *complets*. De fait, certains fragments syntaxiques ont été identifiés comme ayant une complexité moindre, les plus évidents de ces fragments étant le fragment des QBF sans symbole propositionnel dont la complexité vis-à-vis du problème de validité est dans la classe $\mathcal{P}$ (polynomiale en temps), le fragment des QBF *prénexes* uniquement existentiellement quantifiées (qui ne sont autres que des instances du problème SAT) qui est $\mathcal{NP}$-complet et le fragment des QBF prénexes uniquement universellement quantifiées (qui ne sont autres que des instances du problème $TAUT$) qui est $co-\mathcal{NP}$-complet. Nous marquons donc ce gène par des allèles se rapportant aux grandes classes de complexité : « $PSPACE$ », « $\mathcal{P}$ », « $\mathcal{NP}$ » et « $co-\mathcal{NP}$ ». Au-delà de ces fragments triviaux, certains ont été simplement caractérisés [123] et d'autres ont été plus algorithmiquement traités. C'est le cas, en particulier, des QBF sous *forme normale conjonctive* dont les *clauses* contiennent au plus deux littéraux : un algorithme décide de ce fragment linéairement en temps [4]. C'est aussi le cas des QBF restreintes aux *clauses de Horn* [45, 57, 129]. Enfin, c'est aussi le cas, selon des syntaxes propres, pour les procédures de décision ou non, qui traitent les logiques non-monotones [111].

Troisième gène. Le troisième gène détermine si la procédure est basée directement sur la définition de la sémantique (allèle « Sémantique ») ou sur une manipulation symbolique (allèle « Symbolique »). Ce dernier allèle se réfère à la théorie de la preuve dont de nombreux systèmes syntaxiques sont issus pour la *logique propositionnelle* comme pour la *logique des prédicats* (voir [86] pour de nombreux exemples) : un système syntaxique démontré *correct* et *complet* vis-à-vis de la sémantique des QBF ; mais pas seulement : la théorie des graphes peut entrer en jeu comme le démontre l'algorithme pour les 2QBF [4].

Quatrième gène. Le quatrième gène détermine si la procédure conserve la complexité polynomiale en espace (allèle « Polynomial en espace ») ce qui correspond à rester dans la classe de complexité du problème évoqué ou si elle a recours à un mécanisme nécessitant un espace de taille exponentielle dans le pire des cas (allèle « Exponentiel en espace »). Certaines techniques d'optimisation transforment une procédure de décision de complexité polynomiale en espace en une procédure de décision exponentielle en espace si elles ne sont pas bridées ; nous ne prendrons pas en compte ces optimisations dans la détermination de ce gène mais uniquement la complexité en espace de la procédure initiale.

Cinquième gène. Le cinquième gène détermine quel fragment syntaxique est en entrée de la procédure. Ce fragment est d'importance car suffisamment (ou trop) restreint, il peut réduire la puissance de la procédure à ne traiter que des niveaux de classes de complexité fixés de la hiérarchie polynomiale, c'est le cas en particulier des procédures uniquement dédiées au problème SAT, au problème $TAUT$ et aux formalismes des logiques non-monotones (selon des syntaxes *ad hoc*) mais aussi des 2QBF déjà évoquées et des QBF prénexes dont la matrice est formée de clauses de Horn. Il est aussi d'importance car il détermine les propriétés des opérateurs permettant de réduire l'espace de recherche.

En outre, même si le fragment est complet, la conversion d'une QBF, en un fragment restrictif sur la portée des quantificateurs ou sur l'ensemble des connecteurs, peut en détruire la structure et faire perdre de nombreuses informations utiles comme nous en donnons une illustration au chapitre 6, ou être tout simplement inadapté. Ce constat est particulièrement vrai pour les hégémoniques QBF sous FNC prénexes qui est le fragment le plus étudié et que nous avons déjà présenté dans le chapitre 1 (allèle « FNC prénexe »). La perte de structure est due à la *distributivité*, si elle est utilisée, qui la dissout complètement, à la *mise sous forme prénexe* et à l'élimination des bi-implications et des ou-exclusifs ; nous revenons au chapitre 6 sur ces effets délétères. Mais ce qui disqualifie le plus le fragment FNC est son inadaptation à la symétrie existentiel/universel qui se reflète dans le couple FNC/*FND*. La porte est donc ouverte aux fragments FNC/FND (allèle « FNC/FND ») et en *forme normale négative* (allèle « FNN ») qui préservent la symétrie. Le dernier allèle « QBF » de ce gène fait référence au langage des QBF tout entier, en particulier, en intégrant la bi-implication et le ou-exclusif.

Sixième gène. Le sixième gène détermine : si le principe même de la procédure se suffit à lui-même (allèle « Monolithique ») ; si la procédure fait appel à un *oracle* pour résoudre des sous-problèmes de complexité moindre (allèle « À Oracle ») ; si la procédure fait appel à une transformation radicale vers un autre formalisme de décision (allèle « Transformation »). Ce sixième gène est capital puisqu'il partitionne en trois grandes « espèces » les procédures de décision que nous dénommons par leurs allèles et qui disposent de leurs gènes propres.

2.2 Trois espèces

2.2.1 L'espèce « À Oracle »

Les procédures de décision relevant de l'espèce « À Oracle » reviennent à l'esprit de la définition même de la *hiérarchie polynomiale* en utilisant un oracle pour résoudre un problème de complexité plus faible.

Premier gène propre à l'espèce « À Oracle ». Le premier gène propre à l'espèce « À Oracle » est donc le niveau de complexité de l'oracle dans la hiérarchie polynomiale et possède les mêmes allèles que le deuxième gène. L'oracle fait un « peu plus » que décider : il rend soit un modèle propositionnel dans le cas du schéma des procédures `QBDD(X)` [13, 14] soit l'ensemble des modèles propositionnels dans le cas de la procédure `QSAT` [176].

Second gène propre à l'espèce « À Oracle ». Le second gène propre à l'espèce « À Oracle » exprime la manière dont le résultat de l'oracle est exploité. La première approche (allèle « Simplification ») réalise l'élimination d'un ensemble de quantificateurs d'une sous-formule en calculant une formule propositionnelle équivalente au sens de la préservation des modèles propositionnels et construite sur les symboles propositionnels libres de la sous-formule pour réinsérer le résultat à la place de la sous-formule et itérer. C'est l'approche adoptée par la procédure de décision `QSAT` avec un oracle qui calcule à partir d'une

sous-formule existentiellement quantifiée contenant des symboles propositionnels libres une forme normale disjonctive équivalente. La seconde approche (allèle « Agrégation ») utilise l'oracle comme un générateur de modèles pour des problèmes de complexité moindre et tente alors de les agréger pour construire (ou mettre en évidence l'existence d') un modèle QBF. C'est l'approche adoptée par le schéma[3] de procédures `QBDD(X)` qui utilise un générateur de modèles propositionnels pour les agréger en un modèle QBF construit sous la forme d'un *BDD* ; ce schéma se décline en deux instances : l'une sous la forme d'une procédure de décision `QBDD(DLL)` avec pour générateur un `QDLL` entrelacé avec la construction du BDD et l'autre sous la forme d'une procédure incomplète, au sens du premier gène, `QBDD(LS)`, avec pour générateur une recherche locale.

2.2.2 L'espèce « Transformation »

L'espèce « Transformation » utilise une transformation qui doit être démontrée correcte et complète vers un autre formalisme qui est, lui, exécuté pour décider.

Premier gène propre à l'espèce « Transformation ». Le premier gène propre à l'espèce « Transformation » détermine qu'elle transformation est utilisée. La première idée qui vient à l'esprit est d'utiliser, pour une QBF prénexe, la sémantique du quantificateur existentiel et de décider de la validité d'une QBF ne contenant uniquement des quantificateurs universels (c'est-à-dire décider d'un problème $TAUT$, allèle « $\exists$-expansion ») ou de manière duale, d'utiliser la sémantique du quantificateur universel et de décider de la validité d'une QBF contenant uniquement des quantificateurs existentiels (c'est-à-dire décider d'un problème SAT, allèle « $\forall$-expansion »). Mais la principale transformation est la *skolémisation* qui élimine d'une QBF prénexe les quantificateurs existentiels (ou universels) au profit de symboles de fonction ; c'est le cas de `sKizzo` et de la transformation proposée au chapitre 5.

Second gène propre à l'espèce « Transformation ». Le second gène propre à l'espèce « Transformation » spécifie le langage vers lequel la QBF est transformée et par conséquent le type de procédure de décision qui permet *in fine* de décider. C'est vers SAT que la procédure `sKizzo` transforme une QBF tandis que nous proposons dans le chapitre 5 de transformer vers un programme ASP ; ces deux transformations utilisent la *skolémisation propositionnelle* et transforment, par un encodage exponentiel, la QBF, sous FNC pour `sKizzo` et sous forme prénexe mais de matrice quelconque pour notre approche, en un problème relevant de la complexité $\mathcal{NP}$-complet.

2.2.3 L'espèce « Monolithique »

La majeure partie des procédures de décision pour le problème de validité des QBF sont des membres de l'espèce « Monolithique », qui contient aussi les procédures qui font appel pour terminer le calcul à une procédure pour les problèmes SAT ou $TAUT$ mais uniquement pour des raisons d'efficacité et non pour des raisons conceptuelles.

[3] « schéma » car l'oracle peut varier dans sa manière de calculer

Premier gène propre à l'espèce « Monolithique ». Le premier gène propre à l'espèce « Monolithique » détermine la manière dont est manipulée la QBF en interne. Les premiers travaux sur les QBF se sont algorithmiquement concentrés sur le fragment FNC prénexe [120, 47, 187] par extension des travaux similaires sur le problème SAT (la procédure de décision `decide` [187] a été implantée à partir du système `sat0` [133] tandis que la procédure de décision `QUBE` [103] a été implantée à partir du système `SIM` [100]). Le fragment FNC a été très étudié, pour le problème SAT, car il est simple à mettre en œuvre mais surtout parce qu'il a d'excellentes propriétés : en premier lieu, comme conjonction, il permet d'expliciter une collection de connaissances ; en second lieu, la détection de l'*insatisfiabilité* est localisée à la *clause* ce qui favorise les procédures de recherche. D'autres bonnes propriétés vis-à-vis du problème SAT seront explicitées ci-dessous lorsque nous traiterons du *retour-arrière*, propriétés qui s'étendent en partie au traitement des symboles propositionnels existentiellement quantifiés des QBF puisque le problème SAT n'est autre que le problème de validité mais pour des QBF prénexes uniquement existentiellement quantifiées. De manière duale, ces excellentes propriétés peuvent se retrouver pour le traitement des symboles propositionnels universellement quantifiés mais pour des QBF sous *forme normale disjonctive* (FND) ; ainsi le fragment FNC peut être augmenté, en interne à la procédure, soit à un fragment *FNCA* [230] qui ne rajoute des *cubes* que lors du processus de calcul (allèle « FNCA »), soit à un fragment FNC/FND [227] (allèle « FNC/FND »). Mais plus largement, le fragment des QBF sous *forme normale négative* peut être utilisé (allèle « FNN ») [136, 80, 17] ; à l'instar du fragment FNC, dont il partage l'algorithme de mise sous forme normale à l'exclusion de la distributivité permettant ainsi de conserver une croissance polynomiale de la formule (de plus, il est complet et facile à mettre en œuvre). Lorsque le fragment en entrée est prénexe, les quantificateurs peuvent être replacés au plus près des occurrences de leurs symboles propositionnels associés dans un processus inverse à la *mise sous forme prénexe* : le *miniscoping* [17]. L'objectif de ce processus est de diminuer l'espace de recherche en retrouvant (ou mettant en évidence) un lieur non linéaire (un *arbre de quantificateurs*) pour lequel les dépendances sont moindre. Provenant du domaine de la représentation des circuits booléens, d'autres représentations ont été investiguées autour des *graphes acycliques orientés* (allèle « DAG ») : les procédures `ZQSAT` et `Q-PREZ` opèrent sur les *ZBDD* qui sont utilisés pour représenter de manière compacte des ensembles de *clauses* ; la procédure `AIGsolve` opérant sur les *AIG*, la procédure `QBDD` opérant sur les *BDD* et la procédure `CirQit` opérant sur une représentation de circuit logique, elles recherchent principalement le partage des sous-circuits.

Deuxième gène propre à l'espèce « Monolithique ». Le deuxième gène propre à l'espèce « Monolithique » détermine la manière dont sont éliminés les quantificateurs. Toutes les procédures de décision n'ont pas recours à l'élimination des quantificateurs : dans le cas des QBF prénexes, et par des transformations qui préserve la validité, la preuve de l'insatisfiabilité de la matrice est suffisante pour démontrer la non-validité de la QBF ; la *Q-résolution* [120] en est le meilleur exemple : le quantificateur disparaît car le symbole propositionnel associé n'a plus d'occurrence dans la matrice. L'idée première d'une élimination de quantificateurs est d'appliquer simplement la sémantique du quantificateur

par son expansion, qu'il soit le plus externe (*top-down*, allèles « ∃-expansion↓ » et « ∀-expansion↓ ») ou le plus interne (*bottom-up*, allèles « ∃-expansion↑ » et « ∀-expansion↑ »).

Le test de validité d'une QBF prénexe, dans le cas de l'élimination du quantificateur le plus externe, est donc basé sur l'algorithme 2 de recherche du chapitre 1 qui revient à tester récursivement la validité de deux nouvelles QBF prénexes structurellement identiques et ne variant que sur la substitution par une constante du symbole propositionnel associé au quantificateur éliminé. En l'absence de machine indéterministe et pour préserver le caractère $PSPACE$, un mécanisme de *retour-arrière* (ou « backtracking »), souvent implanté sous la forme d'une pile de points de choix (entre la substitution d'un symbole propositionnel par $\top$ ou par $\bot$) sur lesquels l'algorithme devra revenir (en cas d'échec pour le quantificateur existentiel et de succès pour le quantificateur universel), assure que la procédure reste polynomiale en espace [48] ; ce mécanisme est reflété par deux allèles (« ∃-backtrack » et « ∀-backtrack ») pour ce retour-arrière dit « chronologique ». La procédure de décision `QDLL`, basée sur celle de *Davis, Logemann et Loveland* (« le » DLL[4]) [68] et dont s'inspirent une bonne part des procédures de décision pour QBF, a donc pour allèle la combinaison « ∃-backtrack » et « ∀-backtrack ». Le retour-arrière, qui est simple à mettre en œuvre, fait reparcourir des parties de l'espace de recherche inutilement, ce qui peut être en partie évité par une analyse des raisons de l'échec, dans le cas existentiel, et du succès, dans le cas universel. Les techniques de *retour-arrière intelligent* réalisent cette analyse pour mettre en lumière les symboles propositionnels sur lesquels il est inutile de revenir et les informations calculées qui ne seront pas remises en cause par le retour-arrière. La technique du *backjumping* (ou « conflict-directed backtracking » pour le problème SAT [144]) met en œuvre cette analyse pour « sauter » dans la pile des points de choix ceux qui se révéleront, de toute manière, inutiles. Dans le cas d'un échec, ces points de choix font références aux symboles existentiellement quantifiés qui ne modifierons pas cet échec (« conflict directed backjumping » ou « dependency-directed backtracking for false subproblems » [130], allèle « ∃-backjumping ») tandis que, dans le cas d'un succès, ces points de choix font références aux symboles universellement quantifiés qui ne modifierons pas ce succès (« solution directed backjumping » ou « dependency-directed backtracking for true subproblems » [130], allèle « ∀-backjumping »). Le backjumping ne retient du travail effectué sur les symboles propositionnels entre la constatation de l'échec, dans le cas existentiel, ou du succès, dans le cas universel, et le point de choix dépilé que ceux qui y ont participés. Le backjumping n'est que la manifestation algorithmique de la permutabilité à l'intérieur du système de preuve sous-jacent à la procédure de décision [80]. Les techniques de retour-arrière intelligent s'appuient en fait sur l'*apprentissage de lemme* (introduit pour le problème SAT sous le terme de « conflict-based equivalence » [144]) dont le backjumping n'est qu'un avatar élémentaire. La problématique de la construction d'un lemme est indépendante de la notion de retour-arrière mais dans le cadre de la définition des procédures de décision pour le problème de validité, elle se confine à s'intégrer au retour-arrière intelligent. Un lemme prend la forme d'une *clause* dans le cas d'un échec pour ne pas revenir inutilement sur un point de choix généré par l'expansion d'un quantificateur existentiel (« conflict learning », allèle « ∃-learning ») et la forme d'un *cube* dans le cas d'un succès pour ne

[4]ou parfois DPLL, avec un « P » pour « Putnam » de la procédure de décision *Davis et Putnam*

pas revenir inutilement sur un point de choix généré par l'expansion d'un quantificateur universel (« solution learning », allèle « ∀-learning »). Le risque majeur de l'application de l'apprentissage de lemme est qu'elle peut se révéler exponentielle en espace si elle n'est pas contrôlée [130] ce qui fait quitter la complexité $PSPACE$ à la procédure de décision qui l'intègre. L'utilisation d'un lemme dans une procédure de décision pour les QBF s'apparente à l'utilisation de la *règle de la coupure* dans le *calcul des séquents* : introduire une formule qui permet de réduire (parfois exponentiellement) le nombre d'inférences. Les heuristiques de choix pour l'élimination du quantificateur le plus externe reprennent celles employées dans les procédures de décision pour le problème SAT comme par exemple celle qui consiste, pour le fragment FNC, à choisir le littéral le plus présent dans les plus petites clauses [38] étendue dans `QSOLVE` [84] aux QBF. Mais ces heuristiques ont dans le cas QBF moins d'impact car le choix du symbole propositionnel qui doit respecter l'ordre partiel du lieur ne peut être aussi libre que celui pour le cas propositionnel. L'apprentissage de lemme élimine *a posteriori* les parties de l'espace de recherche assurément non-valides mais il est possible grâce à l'*abstract branching* de réduire cette espace *a priori* en étendant les *scénarii* à d'autres valuations pour les symboles propositionnels quantifiés universellement.

Passons maintenant à l'élimination du quantificateur le plus interne qui correspond à l'application du principe de Fourier-Motzkin, mis en pratique, par exemple, dans la résolution des systèmes d'inéquations linéaires [196]. L'algorithme de *Davis et Putnam* (« le » DP) [69] pour le problème SAT est basé sur cette élimination du quantificateur existentiel le plus interne. Cette technique s'appelle aussi *multi-résolution* dans le cadre du problème SAT [50] et du problème QBF [171] pour l'élimination des quantificateurs existentiels. La multi-résolution n'est pas linéairement close mais seulement polynomialement close pour le fragment FNC [83]) ; l'élimination par expansion du quantificateur universel le plus interne est linéairement clos pour le fragment FNC. Ainsi les procédures de décision `QMRES` [171], `quantor` [36] et `QuBIS` [181] qui sont basées sur la multi-résolution ne sont plus polynomiales en espace mais, dans le pire des cas, exponentielles en espace. Le fragment FNN demeure linéairement clos pour l'expansion des quantificateurs les plus internes ; ainsi la procédure de décision `Nenofex` [136] qui étend la procédure `quantor` au fragment FNN est polynomiale en espace.

Troisième gène propre à l'espèce « Monolithique ». Le troisième gène propre à l'espèce « Monolithique » détermine les techniques pour la manipulation des connecteurs. La première est liée à la sémantique des quantificateurs qui substitue un symbole propositionnel par $\top$ ou $\bot$: la simplification selon les *équivalences logiques* (allèle « Simplification $\top$ et $\bot$ ») qui exprime les propriétés algébriques des deux constantes propositionnelles (pour la négation, la disjonction et la conjonction : (0.7), (0.8), (0.13), (0.14), (0.15), (0.16) et celles qui sont déductibles pour les autres opérateurs par (0.1), (0.3) et (0.30)). Associées aux équivalences logiques : $(x \wedge F) \overset{0.26}{\equiv} (x \wedge [x \leftarrow \top](F))$ et $(\neg x \wedge F) \overset{0.27}{\equiv} (\neg x \wedge [x \leftarrow \bot](F))$ (x un symbole propositionnel quantifié existentiellement), elles forment, pour le fragment FNC, la *propagation de clauses unitaires* (allèle « Propagation unitaire », c'est-à-dire dans le premier cas : élimination des clauses contenant x plus élimination des occurrences de $\neg x$ et dans l'autre cas : élimination des clauses contenant $\neg x$ plus élimination des

occurrences de x). De ces équivalences, s'obtient aussi une propriété plus globale pour le fragment FNC : la *propagation de littéraux monotones* qui élimine pour les littéraux monotones (c'est-à-dire n'apparaissant que sous une seule *polarité*) qui sont existentiellement quantifiés les clauses qui les contiennent et pour ceux qui sont universellement quantifiés leurs occurrences. Ces deux techniques sont directement héritées de l'algorithme de *Davis, Logemann et Loveland* [68] pour le problème SAT. L'application de la sémantique du quantificateur universel pour une QBF sous FNC apporte aussi une technique simple à mettre en œuvre : la *réduction universelle* (allèle « Réduction universelle ») qui élimine purement et simplement d'une clause le symbole propositionnel associé à ce quantificateur si celui-ci est plus interne que les quantificateurs associés aux autres symboles propositionnels de la clause. Toujours pour le fragment FNC, la *résolution de Robinson* et sa restriction pour la logique propositionnelle, qui est basée sur l'équivalence logique $((\neg x \vee A) \wedge (x \vee B)) \equiv ((\neg x \vee A) \wedge (x \vee B) \wedge (A \vee B))$, s'étendent aux QBF en la *Q-résolution* [120] (le symbole propositionnel est alors nécessairement existentiellement quantifié, allèle « Résolution ») dont la complexité en espace n'est pas polynomiale mais exponentielle sauf si elle est restreinte. L'*hyper résolution binaire*, implantée dans `2clsQ`, qui étend la propagation de clauses unitaires est une technique qui peut se révéler efficace surtout sous la forme d'un prétraitement [194]. Si l'étape de résolution est appliquée systématiquement sur le même symbole propositionnel, le résultat est la *multi-résolution* (que nous avons déjà commentée en tant qu'élimination de quantificateurs). D'autres techniques restées relativement peu usitées ou triviales parsèment la littérature des premières heures sur les QBF prénexe sous FNC ; sans objectif d'exhaustivité, nous citons pour mémoire : l'*échantillonnage*, la *détection des littéraux en échec*, la *falsifiabilité existentielle triviale*, la *falsifiabilité par littéraux complementaires*, la *falsifiabilité universelle triviale* et la *validité universelle triviale*. Par parfaite dualité, les techniques pour le fragment FNC ont leurs reflets pour le fragment FND et une même structure de donnée peut être utilisé pour les deux fragments selon l'interprétation dont elle en est faite. Cette extension FND peut-être vide au départ et les cubes sont alors ajoutés pour l'apprentissage de lemme [104, 230] mais le problème initial peut être converti *ab initio* par une duplication de l'information en FNC/FND [227] voire défini, dans l'esprit d'un jeu à deux joueurs, dans le fragment FNC pour le joueur existentiel et dans le fragment FND pour le joueur universel [190]. Les techniques pour le fragment FNC sont des cas limites des propriétés algébriques des connecteurs qui peuvent s'appliquer aux QBF quelconques. (De même, la propriété globale des littéraux monotones pour le fragment FNC n'en est qu'une restriction [79].) Ainsi la propagation de clauses unitaires s'étend au fragment FNN en l'*élimination des littéraux unitaires globalement* ou *localement* [79] (allèle « Elimination des littéraux unitaires ») et, plus généralement, en la propagation des propriétés algébriques (allèle « Propagation ») que nous aborderons dans le chapitre 4. Enfin, une technique importante lorsque les QBF quelconques sont considérées est la propagation dite « don't care » [220, 112] (allèle « Propagation don't care ») qui propage l'information dans une sous-formule selon laquelle le calcul de la valeur de vérité de celle-ci n'est plus pertinent car il ne peut faire varier le résultat final ; cette technique se retrouve implicitement dans l'utilisation des *règles d'élimination des connecteurs* (allèle « Elimination des connecteurs ») issues du *calcul des séquents* qui est la base du système syntaxique $GQBF$

de la procédure de décision `qpro` [79].

2.3 Une analyse de la population

Attribuer un génome à certaines procédures de décision est délicat, ainsi la procédure `QSAT` que nous identifions comme faisant partie de l'espèce « À Oracle » peut être classée comme faisant partie de l'espèce « Monolithique » [230] puisque la procédure qui sert d'oracle relève de la *résolution* pour éliminer les quantificateurs : nous pensons que l'esprit de la procédure `QSAT` réside plus dans sa relation à l'oracle que par la spécification de l'oracle lui-même.

L'entière population des procédures de décision pour le problème de validité des QBF partage le soucis de réduire dynamiquement l'espace de recherche en ne reparcourant pas plusieurs fois le même sous-espace. Des techniques, en prétraitement de la formule, permettent aussi de réduire plus ou moins selon la procédure de décision qui les suit : lors de la mise sous forme prénexe pour minimiser le nombre d'alternance du lieur [81] ou pour briser les symétries syntaxiques par permutation des symboles propositionnels de la QBF [9, 10, 11][5].

Pour ainsi dire, tout article qui introduit une nouvelle procédure de décision pour le problème de validité des QBF démontre, par des exemples choisis, qu'elle améliore l'ensemble des procédures de l'état de l'art. Cependant, la comparaison est délicate car la plus grande part des exemples de référence sont des QBF sous FNC prénexes et la plus grande part des procédures n'acceptent que celles-ci en entrée. Néanmoins, nous pouvons proposer quelques réflexions. La figure 2.1 trace une « phylogénie » pour les procédures selon le premier gène (« Incomplet »/« Décision »), l'espèce (« À Oracle »/« Transformation »/« Monolithique ») et enfin le deuxième gène propre à l'espèce Monolithique selon que la procédure de décision utilise l'élimination des quantificateurs par expansion bottom-up ou top-down.

L'espèce « À Oracle » est restée peu étudiée et présente des résultats peu satisfaisants, mais nous pensons qu'elle n'a pas livrée toute sa puissance et nous y reviendrons dans la conclusion lorsque nous dresserons quelques perspectives.

Les compétitions sur le problème de validité des QBF ont révélé des champions différents selon les années [174] ; le champion de la dernière compétition [153] qui avait déjà brillé lors des précédentes évaluations [152] est, de manière surprenante, un membre de l'espèce « Transformation » : le système `sKizzo` [23, 27] ; mais celui-ci contient bien plus que de la *skolémisation propositionnelle* et, de part sa structure complexe, il est difficile de séparer les mérites des différents composants.

L'espèce « Monolithique » est la plus représentée car elle étend des approches très étudiées pour le problème SAT. Deux sous-populations se distinguent : les systèmes, tels que `quantor`, qui s'appuient sur l'expansion bottom-up qui sont efficaces sur certaines familles d'exemples structurés non aléatoires, l'ennemi ici étant l'espace puisque cette sous-population est, pour le fragment FNC, exponentielle en espace ; les systèmes, tels que

[5] Ces symétries peuvent aussi être utilisées dynamiquement pour réduire l'espace de recherche [12]

`QUBE`, qui s'appuient sur l'expansion top-down et une pile de retour-arrière pour garantir la polynomialité en espace, l'ennemi est alors le temps.

C'est dans cette sous-population que l'inadéquation du fragment FNC prénexe est apparue le plus clairement. La mise sous forme prénexe est un processus non-déterministe qui influence grandement l'efficacité des systèmes [78] ; elle fixe un ordre total à partir d'un ordre partiel dont les heuristiques de choix du symbole propositionnel à éliminer sont esclaves (l'inversion des quantificateurs dans le lieur n'est pas libre comme pour le problème SAT) ; elle crée des dépendances entre quantificateurs non pertinentes faisant croître exponentiellement l'espace de recherche ; elle fait croître exponentiellement la taille de la QBF lors de la *linéarisation* ; elle rend l'extraction d'une connaissance du modèle difficilement exploitable du fait de la mise en relation de symboles propositionnels qui étaient dans des sous-formules indépendantes (cf. chapitre 6 pour ces deux derniers points). Pour les systèmes ayant en entrée des QBF prénexes, la solution proposée a été de construire des *arbres de quantificateurs* pour tenter de recouvrer l'arborescence initiale ou mieux en terme de *dépendances* [26, 109] ce qui est impossible dans le cas de la *linéarisation*. Obtenir l'ensemble des dépendances entre symboles propositionnels est *calculable* mais malheureusement ce problème est lui-même $PSPACE$*-complet* [191]. Si les propriétés du quantificateur existentiel dans la recherche de la validité se retrouvent dans le fragment FNC, il n'en va pas de même pour celles du quantificateur universel qui se retrouvent dans le fragment FND. Les procédures efficaces actuelles qui sont basées sur l'élimination de quantificateurs par expansion top-down possèdent en interne une structure FNC/FND ou FNN ; elles intègrent l'apprentissage de lemme, qui s'il n'est pas restreint, rend le système exponentiel en espace ce qui rejoint la nécessité pour les procédures basées sur l'expansion bottom-up dont l'ennemi est la ressource en espace [36] de gérer efficacement la redondance de clauses ou de cubes. Pour les QBF sous FNN, ou quelconque, la représentation sous forme d'arbre [136] est plus aisée à implémenter que sous la forme des *graphes acycliques orientés* tels que les *BDD* [112] ou les *AIG* [175].

Les ingrédients présents dans cette étude peuvent se marier de bien de nouvelles manières et des implantations plus modulaires permettraient de concevoir des chimères mélangeant nombre des techniques comme le fait la procédure `sKizzo`. Néanmoins ce n'est pas la direction que nous avons choisi, la chronologie le montrant : que cela soit en entrée ou en interne, le fragment FNC prénexe tend a être abandonné. Deux options possibles avant de ne plus former qu'une direction : l'abandon du prénexe en une structure de quantificateurs issue du problème initial et l'abandon du fragment FNC. Nous avons dans un premier temps opté pour la seconde : dans le chapitre 4, nous dérivons pas à pas à partir des propriétés algébriques du domaine des QBF prénexes une procédure de décision pour le problème de validité, choix motivé par l'absence de travaux sur les propriétés locales de propagation pour des QBF (prénexes) incluant la bi-implication et le ou-exclusif ; dans le chapitre 5, nous compilons les QBF prénexes en des *programmes logiques normaux* grâce, comme la procédure `sKizzo`, à la *skolémisation propositionnelle*. Le chapitre 4 est prolongé en conclusion par une intégration du langage complet des QBF en entrée et en interne.

La communauté QBF s'est longtemps focalisée sur le problème de validité des QBF

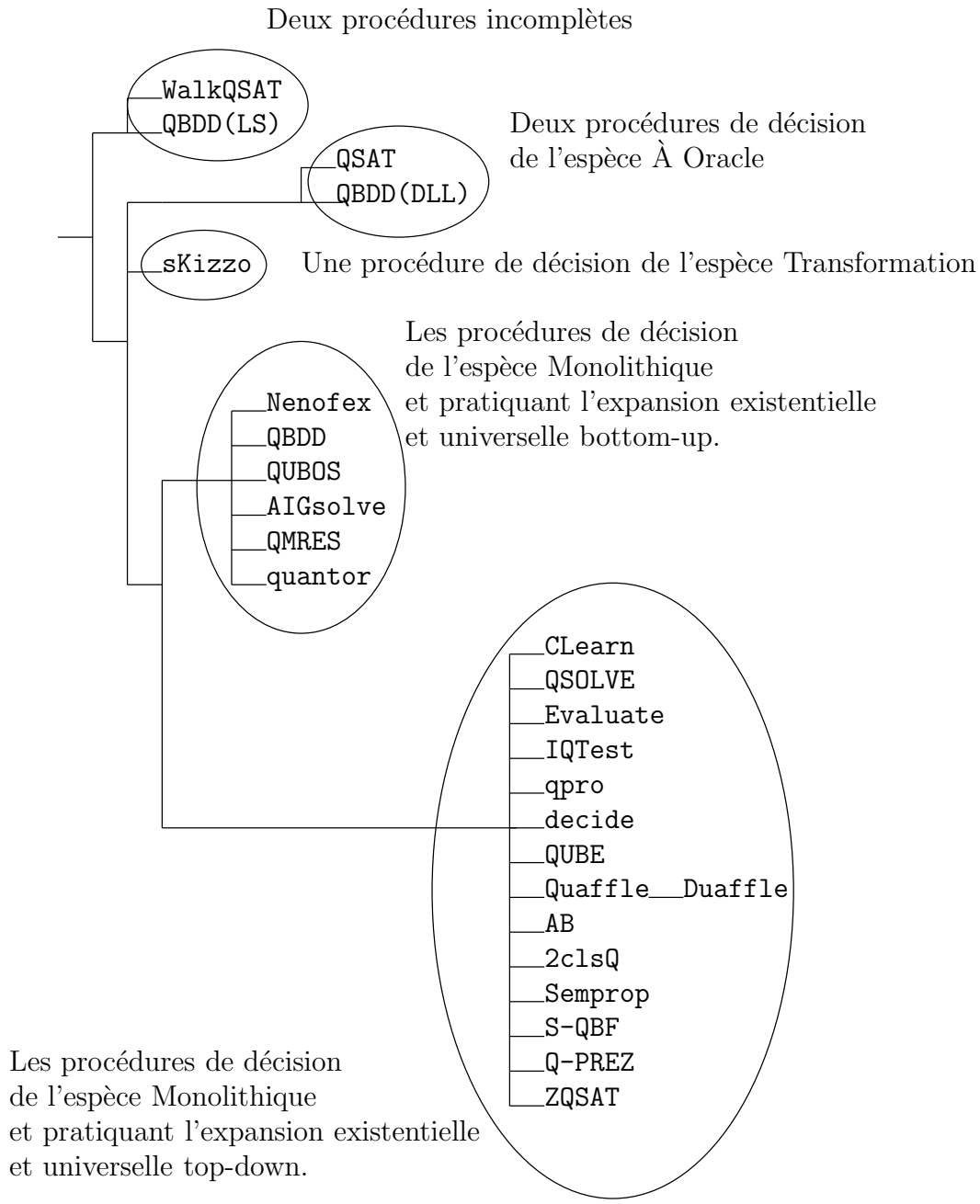


FIG. 2.1 – Une phylogénie pour les procédures portant sur la validité des QBF selon le premier gène (« Incomplet »/« Décision »), l'espèce (« À Oracle »/« Transformation »/« Monolithique ») et enfin le deuxième gène propre à l'espèce Monolithique selon que la procédure de décision utilise l'élimination des quantificateurs par expansion bottom-up ou top-down.

qui est un *problème de décision*, mais lorsqu'une solution est requise, comme dans un jeu fini à deux joueurs, c'est l'« interprétation » de la QBF (dans le sens d'un langage de programmation) qui est nécessaire, ce qui ouvre la voie à la question de la compilation des QBF (prénexes) ; c'est ce thème que nous abordons au chapitre 3 selon l'algorithmique des deux grandes sous-espèces de l'espèce Monolithique, englobant dans un même formalisme la compilation des QBF prénexes, le calcul des modèles QBF et des certificats.

Les applications et la manière de les programmer a été en grande partie la cause des recherches autour de fragments plus riches que celui des QBF prénexes FNC ; le chapitre 6 prolonge ces réflexions et établit un plaidoyé pour l'utilisation en entrée *et* en interne des QBF quelconques.

2.4 Parallélisme

À notre connaissance, il existe trois implantations de procédures de décision parallèles pour le problème de validité des QBF : `PQSOLVE` [84], `PaQube` [131] et `QMiraXT` [132]. Ces procédures sont toutes basées sur un algorithme de recherche séquentiel par élimination du quantificateur le plus externe vers le plus interne (`QSOLVE` [84] pour `PQSOLVE`, `QUBE` [103] pour `PaQube` et `PaMiraXT` [197], une procédure de décision parallèle pour le problème *SAT*, pour `QMiraXT`) et ainsi appliquent une *stratégie de partitionnement sémantique* qui choisit un symbole propositionnel du bloc de quantificateurs le plus externe et applique la sémantique des quantificateurs pour partitionner le problème et distribuer les tâches.

La procédure `PQSOLVE` est une application distribuée qui utilise les techniques de la parallélisation des programmes de jeu d'échec [84]. Elle instancie un modèle pair-à-pair : un processus inactif demande du travail à un processus choisi au hasard et en devient l'esclave pour une tâche. Chaque processus a une pile de tâches à réaliser qui est augmentée par le partitionnement sémantique de celles qui sont estimées trop complexes ; le maître envoie une de ces tâches à son esclave momentané qui a sollicité une tâche à réaliser. Un esclave peut devenir lui-même le maître d'un autre processus.

La procédure `QMiraXT` est dédiée à la prise en compte du potentiel de performance des architectures modernes multi-cœur et/ou processeurs multithreadés. En utilisant un programme utilisant des threads à mémoire partagée, les clauses apprises par conflit [228] sont partagées en les différents espaces de recherche. Il n'y a pas de processus maître mais à la place un « Master Control Object » (MCO) qui permet aux threads de communiquer via des messages asynchrones pour des évènements globaux (par exemple, si un sous-problème est valide ou non). Le MCO prend en charge aussi la stratégie de partitionnement sémantique, appelée dans ce cadre « Single Quantification Level Scheduling » (ou SQLS) et distribue les tâches.

La procédure `PaQube` est conçue selon le modèle maître/esclave où un processus est dédié au maître (ce qui ne nécessite pas de CPU dédiée) et les autres aux esclaves qui réalisent en fait la recherche. `PaQube` est un application parallèle basé sur le standard *Message Passing Interface* (MPI) [204]. Au travers de messages, les esclaves partagent certaines des clauses et cubes issus respectivement des conflits et solutions appris. Chaque esclave cumule localement toute l'expertise obtenue par l'ensemble des autres esclaves. Le

travail principal du maître est de réaliser, comme pour `QMiraXT`, la SQLS.

De par leur modèle, `PQSOLVE` et `PaQube` sont plus extensibles aux clusters et grids que `QMiraXT` mais ce dernier semble tenir plus compte de l'évolution du matériel que les deux premiers.

Chapitre 3

Les bases littérales

Il y eut un soir, il y eut un matin : troisième jour.

[207] I. Stéphan. Algorithmes d'élimination de quantificateurs pour le calcul des politiques des formules booléennes quantifiées. In *Premières Journées Francophones de Programmation par Contraintes*, 2005.

[208] I. Stéphan. Finding models for quantified Boolean formulae. In *First International Workshop on Quantification in Constraint Programming*, 2005.

[213] I. Stéphan and B. Da Mota. Base littérale et certificat pour les formules booléennes quantifiées. In *Troisièmes Journées Francophones de Programmation par Contraintes*, 2008.

[214] I. Stéphan and B. Da Mota. A unified framework for Certificate and Compilation for QBF. In *Third Indian Conference on Logic and its Applications*, 2009.

Sommaire

Il est assez courant de voir apparaître dans la littérature des articles proposant des méthodes ou des formalismes similaires simultanément ; ce fut le cas par exemple pour la méthode de résolution SLD [137, 140] qui fut décrite lors d'une même conférence par deux auteurs ; il en fut de même d'un formalisme apparu simultanément comme expression pour vérifier la correction des procédures QBF, le `sat`-certificat [27, 25] et comme langage de compilation pour les QBF (comme expression de l'ensemble des modèles) par nos soins [208].

En général, une base de connaissance est compilée à part « une bonne fois pour toute » dans un langage cible et cette nouvelle représentation de la base de connaissance est ensuite soumise à des requêtes. Si l'on considère une QBF prénexe comme un jeu à deux joueurs, une des questions primordiales pour le joueur « existentiel » est la suivante : « Que dois-je jouer pour être sûr de gagner ? ». Si la base de connaissance est compilée à part, c'est que le coût d'une seule requête est prohibitif et qu'en compilant celle-ci le coût sera moindre. Dans le cas du joueur existentiel, sa question reste $PSPACE$-complet si aucune compilation n'est effectuée. Intuitivement, pour un jeu fini à deux joueurs, il n'est pas difficile d'imaginer une représentation qui, selon les mouvements déjà joués, indiquent s'il existe « oui » ou « non » une méthode infaillible pour gagner selon que sont joués présentement les mouvements **vrai** ou **faux**. Une telle représentation est exponentielle par rapport aux nombres de symboles propositionnels encodant les mouvements mais la réponse à la question du joueur existentiel est polynomiale en temps.

Nous présentons le formalisme des *bases littérales* qui étend celui des **sat**-certificats ; nous montrons comment l'algorithme 2 de recherche *recherche_prenexe* (cf. § 1.5.1) peut être étendu pour construire des **sat**-certificats ; nous étendons cet algorithme ainsi que l'algorithme 3 d'élimination de quantificateurs *elimination_prenexe_FNC* (cf. § 1.5.2) à la compilation d'une QBF prénexe en une base littérale ; nous démontrons alors que pour les QBF compilées la question précédemment considérée du joueur existentiel est polynomiale en temps.

Dans le reste de ce chapitre, toutes les QBF sont considérées comme étant prénexes.

3.1 Base littérale

Les *formes normales disjonctive* et *conjonctive*, pour la *logique propositionnelle* comme pour les QBF, suivent « horizontalement puis verticalement » la table de vérité où chaque ligne est une *valuation* différente : pour la logique propositionnelle, la forme normale disjonctive est la disjonction des lignes, considérées comme des conjonctions de *littéraux*, des valuations qui s'évaluent à **vrai** tandis que la forme norme conjonctive est la conjonction des lignes, considérées comme des disjonctions des *complémentaires* de littéraux, des valuations qui s'évaluent à **faux** (en vertu de l'*équivalence logique* $\neg\neg \bigvee \bigwedge l \equiv \neg \bigwedge \bigvee \neg l$). Si en logique propositionnelle, cette lecture de la table de vérité est l'exact reflet de l'ensemble des *modèles*, il n'en est pas de même pour les QBF : des conjonctions de littéraux peuvent faire partie de la forme normale disjonctive sans pour autant que ce modèle (propositionnel) pour la matrice soit dans un quelconque modèle QBF.

Au propositionnel, en considérant donc que le simple fait de suivre les règles du jeu

permet au joueur existentiel de gagner, le formalisme évoqué dans l'introduction à ce chapitre est facilement exhibé et obtenu à partir de la forme normale disjonctive et grâce à l'équivalence $((\neg x \wedge A) \vee (x \wedge B)) \equiv ((A \vee B) \wedge (\neg x \vee B) \wedge (x \vee A))$ appliquée du dernier mouvement de la partie jusqu'au premier : pour que le mouvement x puisse mener à la victoire pour une série de mouvements qui précèdent x représentée par la valuation propositionnelle v, il est nécessaire que $I^*(B)(v) = \mathbf{vrai}$ et pour que le mouvement $\neg x$ puisse mener à la victoire, il est nécessaire que $I^*(A)(v) = \mathbf{vrai}$.

Exemple 27 (base littérale propositionnelle) *En continuant l'exemple 1, la formule propositionnelle*

$$\mu = \neg\neg\neg((e \rightarrow d) \rightarrow \neg(f \rightarrow a))$$

à partir de la forme normale disjonctive de μ :

$$\begin{array}{rl} fnd_\mu = & (a \wedge d \wedge e \wedge f) \vee (a \wedge d \wedge e \wedge \neg f) \vee (a \wedge d \wedge \neg e \wedge f) \vee \\ & (a \wedge d \wedge \neg e \wedge \neg f) \vee (a \wedge \neg d \wedge \neg e \wedge f) \vee (a \wedge \neg d \wedge \neg e \wedge \neg f) \vee \\ & (\neg a \wedge d \wedge e \wedge \neg f) \vee (\neg a \wedge d \wedge \neg e \wedge \neg f) \vee (\neg a \wedge \neg d \wedge \neg e \wedge \neg f) \end{array}$$

peut être décomposée selon la séquence de symboles propositionnels $a; d; e; f$ en la formule :

$$\mu \equiv \overbrace{(\neg a \vee \top) \wedge (a \vee \top)}^{1} \wedge \overbrace{(\neg d \vee \top) \wedge (d \vee \top)}^{2} \wedge \underbrace{(\neg e \vee d)}_{3} \wedge \underbrace{(e \vee \top)}_{4} \wedge \underbrace{(\neg f \vee X^\neg)}_{5} \wedge \underbrace{(f \vee X)}_{6}$$

avec

$$X = (a \wedge d \wedge e) \vee (a \wedge d \wedge \neg e) \vee (a \wedge \neg d \wedge \neg e) \vee (\neg a \wedge d \wedge e) \vee (\neg a \wedge d \wedge \neg e) \vee (\neg a \wedge \neg d \wedge \neg e)$$

et $X^\neg = (a \wedge d \wedge e) \vee (a \wedge d \wedge \neg e) \vee (a \wedge \neg d \wedge \neg e)$ ainsi que

$$\begin{array}{ll} & (X \vee X^\neg) \\ \equiv & (\neg e \wedge ((a \wedge d) \vee (a \wedge \neg d) \vee (\neg a \wedge d) \vee (\neg a \wedge \neg d))) \vee (e \wedge ((a \wedge d) \vee (\neg a \wedge d))) \\ \equiv & ((\neg e \wedge \top) \vee (e \wedge d)) \\ \equiv & ((\neg e \vee d) \wedge (e \vee \top)) \end{array}$$

Cette décomposition est intéressante car elle exhibe les propriétés suivantes :

1. *pour toute valeur de vérité de a, la formule propositionnelle admet (au moins) un modèle ;*
2. *pour toute valeur de vérité de d, la formule propositionnelle admet (au moins) un modèle ;*
3. *si e est valué à* **vrai**, *la formule propositionnelle admet un modèle si et seulement si d est valué à* **vrai** *;*
4. *si e est valué à* **faux**, *la formule propositionnelle admet (au moins) un modèle ;*

5. *si e est valué à* **vrai** *et d est valué à* **vrai**, *ou bien si e est valué à* **faux** *alors si f est valué à* **vrai** *la formule propositionnelle admet un modèle si et seulement si la formule propositionnelle $X^\neg$ est évaluée à* **vrai**.
6. *si e est valué à* **vrai** *et d est valué à* **vrai**, *ou bien si e est valué à* **faux** *alors si f est valué à* **faux** *la formule propositionnelle admet (au moins) un modèle si et seulement si la formule propositionnelle X est évaluée à* **vrai**.

La base littérale s'inscrit dans une lecture « verticale puis horizontale » de la table de vérité en accord avec l'ordre du lieur, ce qui est plus proche et plus adapté à la sémantique des QBF, qui ne permet en général pas la permutation des quantificateurs, et son interprétation en terme de jeu à deux joueurs.

Définition 34 (base littérale) *Une* base littérale *est une paire $\langle Q \mid G\rangle$ constituée*
- *soit $Q = \varepsilon$ et $G = \top$ ou $G = \bot$;*
- *soit d'un lieur $Q = q_1x_1 \dots q_nx_n$, $n > 0$, et d'une séquence de paires de formules $G = (P_1, N_1); \dots ; (P_n, N_n)$ telle que les formules P_k et N_k, appelées* gardes, *sont uniquement constituées sur l'ensemble de symboles propositionnels $\{x_1, \dots, x_{k-1}\}$ (ou $\top$ ou $\bot$ lorsque $k = 1$).*

Nous notons $\mathcal{BL}_Q$ l'ensemble des bases littérales pour un lieur Q, $\mathcal{BL} = \bigcup_Q \mathcal{BL}_Q$ le langage des bases littérales, grds la fonction telle que $grds(\langle Q \mid G\rangle) = G$, lieur la fonction telle que $lieur(\langle Q \mid G\rangle) = Q$ et grd la fonction qui associe à un symbole propositionnel sa garde dans une base littérale.

Par la définition précédente :
- si $Q = \varepsilon$ alors $\mathcal{BL}_\epsilon = \{\langle \varepsilon \mid \top\rangle, \langle \varepsilon \mid \bot\rangle\}$;
- si $Q = qx$ alors $\mathcal{BL}_{qx} = \{\langle qx \mid (\top, \top)\rangle, \langle qx \mid (\top, \bot)\rangle,\ \langle qx \mid (\bot, \top)\rangle, \langle qx \mid (\bot, \bot)\rangle\}$

Si le nombre de symboles propositionnels du lieur Q est n alors la taille de $\mathcal{BL}_Q$ est $\underbrace{2^{2^{\cdots^2}}}_{n+1}$.

Exemple 28 (base littérale) *En continuant l'exemple 27,*

$$bl = \langle \forall a \exists d \forall e \exists f \mid (\top, \top); (\top, \top); (d, \top); (X^\neg, X)\rangle$$

et

$$bl' = \langle \forall a \exists d \forall e \exists f \mid (\top, \top); (\top, \bot); (\top, \top); (X^\neg, X)\rangle$$

sont des bases littérales.

Nous interprétons le langage des bases littérales comme une représentation d'un sous-ensemble du langage des QBF.

Définition 35 (interprétation d'une base littérale) *La* fonction d'interprétation *pour les bases littérales est une fonction de l'ensemble des bases littérales $\mathcal{BL}$ dans celui des QBF, notée $(.)^*$ et définie ainsi :*

– *si* $bl = \langle \varepsilon \mid M \rangle$ *alors* $bl^* = M$;
– *si* $bl = \langle q_1 x_1 \dots q_n x_n \mid (P_1, N_1); \dots; (P_n, N_n) \rangle, n > 0$, *alors*

$$bl^* = q_1 x_1 \dots q_n x_n \bigwedge_{1 \leq i \leq n} ((\neg x_i \vee P_i) \wedge (x_i \vee N_i))$$

Exemple 29 (interpretation base littérale) *En continuant l'exemple 28.*

$$bl^* = \forall a \exists d \forall e \exists f \\ (\neg a \vee \top) \wedge (a \vee \top) \wedge (\neg d \vee \top) \wedge (d \vee \top) \wedge (\neg e \vee d) \wedge (e \vee \top) \wedge (\neg f \vee X^{\neg}) \wedge (f \vee X)$$

et

$$bl'^* = \forall a \exists d \forall e \exists f \\ (\neg a \vee \top) \wedge (a \vee \top) \wedge (\neg d \vee \top) \wedge (d \vee \bot) \wedge (\neg e \vee \top) \wedge (e \vee \top) \wedge (\neg f \vee X^{\neg}) \wedge (f \vee X)$$

avec $bl^* \cong bl'^* \cong \forall a \exists d \forall e \exists f \mu$.

Si X est un sous-ensemble de $\mathcal{BL}$ alors X^* est une notation pour $\{bl^* | bl \in X\}$. Nous notons *non_valide* les bases littérales dont l'interprétation est non valide et nous étendons la définition précédente par $non\_valide^* = \bot$. Dans la mesure où la base littérale est d'interprétation valide, les gardes associées aux symboles propositionnels universellement quantifiés peuvent être remplacées par $(\top, \top)$.

L'interprétation de la base littérale fait clairement apparaître la lecture tout d'abord « verticale » de l'ordre des symboles propositionnels puis « horizontale » des gardes (P_x, N_x) qui sont telles que, quelque soit la valuation construite jusqu'au coup portant sur le symbole propositionnel existentiellement quantifié x, si $v \not\models P_x$ alors il n'y a pas de victoire avec le mouvement $[x \leftarrow \top]$ et si $v \not\models N_x$ alors il n'y a pas de victoire avec le mouvement $[x \leftarrow \bot]$; si la victoire est toujours possible, P_x est une représentation de la combinaison des conditions *nécessaires* pour que le mouvement $[x \leftarrow \top]$ puisse mener à une victoire et N_x est une représentation de la combinaison des conditions *nécessaires* pour que le mouvement $[x \leftarrow \bot]$ puisse mener à la victoire, ce qui représente la lecture « horizontale » de la table de vérité.

Le théorème suivant établit que pour toute QBF il existe une base littérale telle que son interprétation a exactement les mêmes modèles que la QBF.

Théorème 2 (complétude de $\mathcal{BL}$) *Soit QM une QBF alors il existe une base littérale $bl \in \mathcal{BL}_Q$ telle que $bl^* \cong QM$.*

Par ce théorème, le fragment des interprétations des bases littérales peut être considéré comme une forme normale pour les QBF et le langage des bases littérales peut être considéré comme un langage cible pour la compilation des QBF.

Lorsqu'une QBF est considérée comme un jeu fini à deux joueurs, la validité de la QBF assure au joueur « existentiel » la victoire s'il suit les mouvements obtenus du modèle de la QBF. Nous sommes particulièrement intéressés par la question suivante : puisque jusqu'à présent nous avons suivi une séquence de fonctions booléennes du modèle de la QBF, puis-je changer d'avis pour le prochain mouvement ? Nous appelons ce problème celui du « choix du prochain mouvement ».

Définition 36 (le problème du choix du prochain mouvement pour un sous-ensemble X de QBF)

- Instance : *Une formule $q_1x_1 \dots q_nx_nM$ d'un sous-ensemble X de* **QBF**, *une substitution $[x_1 \leftarrow C_1] \dots [x_i \leftarrow C_i]$ obtenue à partir d'un modèle pour la QBF $q_1x_1 \dots q_nx_nM$ avec $q_i = \exists$ et $C_1, \dots, C_i \in \{\top, \bot\}$.*
- Question : *Existe-t-il un modèle pour la QBF $q_{i+1} \dots q_nx_n[x_1 \leftarrow C_1] \dots [x_{i-1} \leftarrow C_{i-1}][x_i \leftarrow \overline{C_i}](M)$?*

Clairement, le problème du choix du prochain mouvement demeure PSPACE-complet si nous considérons $X =$ **QBF**.

Nous introduisons une propriété appelée « optimalité » pour les bases littérales pour mettre en exergue un fragment du langage **QBF** pour lequel le problème du choix du prochain mouvement est polynomial en temps.

Définition 37 (optimalité d'une base littérale) *Soit une base littérale bl telle que $bl = (\langle q_1x_1 \dots q_nx_n \mid (P_1, N_1); \dots; (P_n, N_n)\rangle$ et $bl^* = q_1x_1 \dots q_nx_nM$. La base littérale bl est optimale si la propriété suivante est vérifiée. Pour tout i, $1 \leq i \leq n$, soit $[x_1 \leftarrow C_1] \dots [x_{i-1} \leftarrow C_{i-1}]$ une substitution telle que pour tout k, $1 \leq k < i$ si $C_k = \top$ alors $[x_1 \leftarrow C_1] \dots [x_{k-1} \leftarrow C_{k-1}](P_k) \equiv \top$ sinon $C_k = \bot$ et $[x_1 \leftarrow C_1] \dots [x_{k-1} \leftarrow C_{k-1}](N_k) \equiv \top$.*

Alors

$$[x_1 \leftarrow C_1] \dots [x_{i-1} \leftarrow C_{i-1}](P_i) \equiv \top$$
si et seulement s'il existe un modèle pour
$$q_{i+1}x_{i+1} \dots q_nx_n[x_1 \leftarrow C_1] \dots [x_{i-1} \leftarrow C_{i-1}][x_i \leftarrow \top](M)$$

et

$$[x_1 \leftarrow C_1] \dots [x_{i-1} \leftarrow C_{i-1}](N_i) \equiv \top$$
si et seulement s'il existe un modèle pour
$$q_{i+1}x_{i+1} \dots q_nx_n[x_1 \leftarrow C_1] \dots [x_{i-1} \leftarrow C_{i-1}][x_i \leftarrow \bot](M).$$

Nous notons **BLO** *l'ensemble des bases littérales optimales.*

Exemple 30 (optimalite) *En continuant l'exemple 29, la base littérale*

$$bl = \langle \forall a \exists d \forall e \exists f \mid (P_a = \top, \top); (\top, N_d = \top); (d, \top); (a, \top)\rangle$$

n'est pas optimale puisque $P_a \equiv \top$ et $[a \leftarrow \top](N_d) \equiv \top$ mais la QBF $\forall e \exists f[a \leftarrow \top][d \leftarrow \bot](((\neg e \vee d) \wedge (\neg f \vee a)))$ n'admet pas de modèle ($[a \leftarrow \top][d \leftarrow \bot](((\neg e \vee d) \wedge (\neg f \vee a))) \equiv \neg e$) tandis que la base littérale bl' l'est.

$$bl'^* = \forall a \exists d \forall e \exists f$$
$$\overbrace{(\neg a \vee \top) \wedge (a \vee \top)}^{1} \wedge \overbrace{(\neg d \vee \top)}^{2} \wedge \overbrace{(d \vee \bot)}^{3} \wedge \overbrace{(\neg e \vee \top) \wedge (e \vee \top)}^{4} \wedge \overbrace{(\neg f \vee X^{\top})}^{5} \wedge \overbrace{(f \vee X)}^{6}$$

or $bl'^* \cong \forall a \exists d \forall e \exists f \mu$, donc

2. *& 3. les QBF* $\forall e \exists f[a \leftarrow \top][d \leftarrow \top](\mu)$ *et* $\forall e \exists f[a \leftarrow \bot][d \leftarrow \top](\mu)$ *admettent des modèles et les QBF* $\forall e \exists f[a \leftarrow \top][d \leftarrow \bot](\mu)$ *et* $\forall e \exists f[a \leftarrow \bot][d \leftarrow \bot](\mu)$ *n'en admettent pas donc la QBF admet un modèle si et seulement si* $\hat{d} = \{(\mathbf{vrai} \mapsto \mathbf{vrai}), (\mathbf{faux} \mapsto \mathbf{vrai})\}$ *;*
5. *la QBF* $[a \leftarrow C_a][d \leftarrow \top][e \leftarrow C_e][f \leftarrow \top](\mu)$ *admet un modèle si et seulement si* $[a \leftarrow C_a][d \leftarrow \top][e \leftarrow C_e](X^{\neg}) \equiv \top$ *;*
6. *la QBF* $[a \leftarrow C_a][d \leftarrow \top][e \leftarrow C_e][f \leftarrow \bot](\mu)$ *admet un modèle si et seulement si* $[a \leftarrow C_a][d \leftarrow \top][e \leftarrow C_e](X) \equiv \top$*.*

La principale propriété des bases optimales est que le problème du choix du prochain mouvement est polynomial en temps et non plus PSPACE-complet.

Théorème 3 (choix du prochain mouvement) *Le problème du choix du mouvement pour* **BLO*** *est polynomial en temps.*

Si une QBF modélisant un jeu fini à deux joueurs est compilée préalablement en une base littérale optimale, le calcul d'une séquence de mouvements menant à la victoire pour le joueur « existentiel » est polynomial en temps. Une base littérale optimale est alors vue comme un arbre de décision dynamique dont les branches se calculent au fur et à mesure. Cette propriété d'optimalité d'une base littérale est liée à la propriété de « minimalité » d'une QBF qui exprime que la matrice d'une QBF contient exactement les modèles propositionnels nécessaires aux modèles QBF.

Définition 38 (minimalité d'une QBF) *Une QBF est minimale si tous les modèles (propositionnels) de la matrice sont au moins dans un modèle de la QBF.*

Exemple 31 (minimalité) *En continuant l'exemple 30, la base littérale* bl' *est optimale mais ce n'est pas la seule dont l'interprétation soit équivalente à* ξ *: la base littérale*

$$bl'' = \langle \forall a \exists d \forall e \exists f \mid (\top, \top); (\top, \bot); (\top, \top); (a, \top) \rangle$$

est aussi optimale et $bl''^* \cong \xi$*. La table de vérité des matrices des QBF* bl'^* *et* bl''^**, respectivement* μ' *et* μ''*, montre que les QBF* bl'^* *et* bl''^* *sont minimales.*

$v(a)$	$v(d)$	$v(e)$	$v(f)$	$I^*(\mu')(v) = I^*(\mu'')(v)$
vrai	**vrai**	**vrai**	**vrai**	**vrai**
vrai	**vrai**	**vrai**	**faux**	**vrai**
vrai	**vrai**	**faux**	**vrai**	**vrai**
vrai	**vrai**	**faux**	**faux**	**vrai**
vrai	**faux**	**vrai**	**vrai**	**faux**
vrai	**faux**	**vrai**	**faux**	**faux**
vrai	**faux**	**faux**	**vrai**	**faux**
vrai	**faux**	**faux**	**faux**	**faux**
faux	**vrai**	**vrai**	**vrai**	**faux**
faux	**vrai**	**vrai**	**faux**	**vrai**
faux	**vrai**	**faux**	**vrai**	**faux**
faux	**vrai**	**faux**	**faux**	**vrai**
faux	**faux**	**vrai**	**vrai**	**faux**
faux	**faux**	**vrai**	**faux**	**faux**
faux	**faux**	**faux**	**vrai**	**faux**
faux	**faux**	**faux**	**faux**	**faux**

Il apparaît que la QBF $\forall a \exists d \forall e \exists f \neg\neg\neg((e \rightarrow d) \rightarrow \neg(f \rightarrow a))$ *n'est pas minimale puisque, par exemple, la valuation* v *définie par* $v(a) =$ **vrai**, $v(d) =$ **faux**, $v(e) =$ **vrai** *et* $v(f) =$ **vrai** *est un modèle (propositionnel) de la matrice de la QBF sans être dans un quelconque modèle QBF.*

L'interprétation d'une base littérale optimale ne retient que les modèles propositionnels nécessaires et est donc minimale (cf. Démonstrations, lemme 31), mais la réciproque est fausse : il existe des bases littérales qui ne sont pas optimales mais dont l'interprétation est tout de même minimale.

Exemple 32 (réciproque fausse) *La base littérale* $(\langle \exists a \forall d \mid (\top, \top); (a, a) \rangle$ *n'est pas optimale (car* $\forall d \bot$ *n'est pas valide) mais son interprétation est minimale.*

La définition 35 de la fonction d'interprétation d'une base littérale et les définitions 23 et 24 des relations d'équivalence respectant, respectivement, les modèles propositionnels et les modèles QBF, suggèrent plusieurs relations d'équivalences pour les bases littérales ; nous en choisissons une qui préserve l'optimalité des bases littérales.

Définition 39 (relation $\approx$) *Soient* bl *et* bl' *deux bases littérales.* $bl \approx bl'$ *si* $bl^* \cong bl'^*$ *et pour toutes les paires* $(P_i, N_i) \in grds(bl)$ *et* $(P'_i, N'_i) \in grds(bl')$ *correspondant à un symbole existentiellement quantifié des lieurs des bases littérales* bl *et* bl', $P_i \equiv P'_i$ *et* $N_i \equiv N'_i$.

3.2 Algorithme de recherche et sat-certificat

Un **sat**-certificat (cf. § 1.5.3) pour une QBF peut être aisément étendu à une base littérale de la manière suivante : à la séquence du **sat**-certificat, pour chaque symbole

propositionnel quantifié universellement, est ajouté un couple $(\top, \top)$; cette nouvelle séquence est empaquetée dans une paire avec le lieur. Ainsi, l'interprétation de la base littérale obtenue n'a qu'un seul modèle qui est un modèle de la QBF.

Nous nous intéressons au problème suivant : comment étendre une procédure de décision basée sur un algorithme de recherche pour calculer directement un **sat**-certificat plutôt que *a posteriori* par analyse d'une trace. Pour ce faire nous définissons un opérateur pour les bases littérales qui permet à partir des **sat**-certificats pour les deux sous-problèmes d'une QBF de construire un **sat**-certificat.

Définition 40 *L'opérateur* $\circ_x : \mathcal{BL}_Q \times \mathcal{BL}_Q \to \mathcal{BL}_{\forall xQ}$ *est défini ainsi :*

$$\begin{array}{ll} & \langle Q \mid (P_1, N_1); \ldots; (P_n, N_n)\rangle \circ_x \langle Q \mid (P'_1, N'_1); \ldots; (P'_n, N'_n)\rangle \\ = \langle & \forall x Q | (\top, \top); \\ & (((\neg x \vee P_1) \wedge (x \vee P'_1)), ((\neg x \vee N_1) \wedge (x \vee N'_1))); \ldots; \\ & (((\neg x \vee P_n) \wedge (x \vee P'_n)), ((\neg x \vee N_n) \wedge (x \vee N'_n)))\rangle \end{array}$$

Selon cette définition, si x est valué à **vrai** (resp. **faux**) alors pour tout i, $1 \leq i \leq n$, $((\neg x \vee P_i) \wedge (x \vee P'_i)) \equiv P_i$ (resp. P'_i) et $((\neg x \vee N_i) \wedge (x \vee N'_i)) \equiv N_i$ (resp. N'_i). Si $(Q, (P_1, N_1); \ldots; (P_n, N_n))$ et $(Q, (P'_1, N'_1); \ldots; (P'_n, N'_n))$ sont des **sat**-certificats et $Q = q_1 x_1 \ldots q_n x_n$ avec $q_i = \forall$ alors clairement $((\neg x \vee P_i) \wedge (x \vee P'_i)) \equiv \top \equiv ((\neg x \vee N_i) \wedge (x \vee N'_i))$.

Exemple 33 *En continuant l'exemple 27. Soient deux bases littérales*

$$bl_a = \langle \exists d \forall e \exists f \mid (\top, \bot); (\top, \top); (\top, \bot)\rangle$$

et

$$bl_{\neg a} = \langle \exists d \forall e \exists f \mid (\top, \bot); (\top, \top); (\bot, \top)\rangle$$

alors

$$\begin{array}{ll} & (bl_a \circ_a bl_{\neg a}) = \langle \forall a \exists d \forall e \exists f | (\top, \top); \\ & (((\neg a \vee \top) \wedge (a \vee \top)), ((\neg a \vee \bot) \wedge (a \vee \bot))); \\ & (((\neg a \vee \top) \wedge (a \vee \top)), ((\neg a \vee \top) \wedge (a \vee \top))); \\ & (((\neg a \vee \top) \wedge (a \vee \bot)), ((\neg a \vee \bot) \wedge (a \vee \top)))\rangle \\ \approx & \langle \forall a \exists d \forall e \exists f \mid (\top, \top); (\top, \bot); (\top, \top); (a, \neg a)\rangle \end{array}$$

La QBF bl^*_a *a pour unique modèle la politique* $d; \{(e \mapsto f), (\neg e \mapsto f)\}$*, la QBF* $bl^*_{\neg a}$ *a pour unique modèle la politique* $d; \{(e \mapsto \neg f), (\neg e \mapsto \neg f)\}$ *et la QBF* $(bl_a \circ_a bl_{\neg a})^*$ *a pour unique modèle la politique*

$$\{(a \mapsto d; \{(e \mapsto f), (\neg e \mapsto f)\}), (\neg a \mapsto d; \{(e \mapsto \neg f), (\neg e \mapsto \neg f)\})\}.$$

L'opérateur $\circ_x$ compose deux **sat**-certificats en un nouveau **sat**-certificat (cf. Démonstrations lemme 32) : si $bl_\top$ est un **sat**-certificat pour $Q[x \leftarrow \top](M)$ et $bl_\bot$ est **sat**-certificat pour $Q[x \leftarrow \bot](M)$ alors $(bl_\top \circ_x bl_\bot)$ est un **sat**-certificat pour $\forall x Q M$.

Nous présentons l'algorithme de $certificat\_par\_recherche$ qui calcule un **sat**-certificat pour une QBF. L'algorithme $certificat\_par\_recherche$ teste en premier lieu si le lieur est réduit à un unique quantificateur associé à un symbole propositionnel. Dans ce cas,

Algorithme 4 *certificat_par_recherche*

Entrée: Q : le lieur d'une QBF
Entrée: M : la matrice d'une QBF
Sortie: un **sat**-certificat ou *non_valide*

si $Q = qx$ **alors**
  si $q = \exists$ **alors**
    selon M **faire**
      cas $\top$: **retourner** $\langle \exists x \mid (\top, \bot) \rangle$
      cas $\bot$: **retourner** *non_valide*
      cas x : **retourner** $\langle \exists x \mid (\top, \bot) \rangle$
      cas $\neg x$: **retourner** $\langle \exists x \mid (\bot, \top) \rangle$
    fin selon
  sinon
    si $M \equiv \top$ **alors retourner** $\langle \forall x \mid (\top, \top) \rangle$ **sinon retourner** *non_valide* **fin si**
  fin si
sinon
  $Q = qxQ'$
  $bl^{\top} := certificat\_par\_recherche(Q', [x \leftarrow \top](M))$
  si $bl^{\top} =$ *non_valide* **alors**
    si $q = \exists$ **alors**
      $bl^{\bot} := certificat\_par\_recherche(Q', [x \leftarrow \bot](M))$
      si $bl^{\bot} =$ *non_valide* **alors retourner** *non_valide*
      sinon retourner $\langle Q \mid (\bot, \top)\ ;\ grds(bl^{\bot}) \rangle$ **fin si**
    sinon
      retourner *non_valide*
    fin si
  sinon
    si $q = \exists$ **alors**
      retourner $\langle Q \mid (\top, \bot)\ ;\ grds(bl^{\top}) \rangle$
    sinon
      $bl^{\bot} := certificat\_par\_recherche(Q', [x \leftarrow \bot](M))$
      si $bl^{\bot} =$ *non_valide* **alors**
        retourner *non_valide*
      sinon
        retourner $(bl^{\top} \circ_x bl^{\bot})$
      fin si
    fin si
  fin si
fin si

si c'est un quantificateur existentiel quatre cas sont possibles, correspondant dans l'ordre de l'algorithme à : $\exists x \top \equiv \exists x x$, $\exists x \bot \equiv \bot$, $\exists x x \cong \exists x((\neg x \vee \top) \wedge (x \vee \bot))$ et $\exists x \neg x \cong \exists x((\neg x \vee \bot) \wedge (x \vee \top))$. Si le quantificateur est universel alors si $M \equiv \top$ alors $\forall x M \equiv \top$

sinon $\forall xx \equiv \forall x\neg x \equiv \forall x\bot \equiv \bot$. S'il y a plus d'un quantificateur, puisque l'algorithme est un algorithme de recherche, le quantificateur le plus externe est considéré en premier. Si ce quantificateur est existentiel alors si un des deux appels récursifs pour la substitution par $\top$ (resp. par $\bot$) sur le symbole propositionnel x retourne un résultat différent de *non_valide* alors le **sat**-certificat $\langle Q \mid (\top, \bot); grds(bl^+)\rangle$ (resp. $\langle Q \mid (\bot, \top); grds(bl^-)\rangle$) est retourné ; cela exprime que x doit être **vrai** (resp. **faux**) pour avoir au moins un modèle. Si le quantificateur est universel alors si au moins un des appels récursifs pour les substitutions par $\top$ ou $\bot$ pour le symbole propositionnel x retourne *non_valide* alors *non_valide* est retourné sinon les fonctions booléennes des deux **sat**-certificats doivent être composés, pour intégrer le nouvel argument x, par $(bl^+ \circ_x bl^-)$ avant de retourner ce nouveau **sat**-certificat.

Théorème 4 (Correction de *certificat_par_recherche***)** *Soit QM une QBF. $certificat\_par\_recherche(Q, M)$ retourne un **sat**-certificat pour QM si la QBF est valide et non_valide sinon.*

3.3 Compilation d'une QBF en une base littérale

Par le théorème 2 qui établit la complétude du langage des bases littérales, l'ensemble des bases littérales, $\mathcal{BL}$, peut être considéré comme un langage cible pour la compilation des QBF. « Compiler » est alors « résoudre » : calculer une expression dont la caractéristique principale doit être l'accès aux solutions dans une complexité moindre que la formule non compilée.

Les algorithmes *recherche_prenexe* et *elimination_prenexe_FNC* sont deux candidats à l'extension pour un compilateur de QBF vers les bases littérales.

3.3.1 Algorithme de recherche et compilation

Nous nous intéressons ici au problème suivant : comment étendre l'algorithme 2 de recherche, *recherche_prenexe*, en un compilateur de QBF en bases littérales. Pour ce faire nous définissons un opérateur sur les bases littérales qui compile une QBF par la composition du résultat de la compilation de ses deux sous-problèmes.

Définition 41 *Soit $Q = q_1x_1 \ldots q_nx_n$ un lieur et $bl, bl' \in \mathcal{BL}_Q$. L'opérateur $\otimes : \mathcal{BL}_Q \times \mathcal{BL}_Q \rightarrow \mathcal{BL}_Q$ est défini comme suit :*

Si $Q = \varepsilon$, $bl = \langle \varepsilon \mid G\rangle$ et $bl' = \langle \varepsilon \mid G'\rangle$ alors $(bl \otimes bl') = \langle \varepsilon \mid (G \vee G')\rangle$ sinon

$$\begin{array}{rl} & \langle Q \mid (P_1, N_1); \ldots; (P_n, N_n)\rangle \otimes \langle Q \mid (P'_1, N'_1); \ldots; (P'_n, N'_n)\rangle \\ = & \langle Q \mid ((P_1 \vee P'_1), (N_1 \vee N'_1)); \\ & \quad (\mathcal{P}_2 \wedge (P'_2 \vee \mathcal{X}) \wedge (P_2 \vee \mathcal{X}'),\ \mathcal{N}_2 \wedge (N'_2 \vee \mathcal{X}) \wedge (N_2 \vee \mathcal{X}')); \ldots; \\ & \quad (\mathcal{P}_n \wedge (P'_n \vee \mathcal{X}) \wedge (P_n \vee \mathcal{X}'),\ \mathcal{N}_n \wedge (N'_n \vee \mathcal{X}) \wedge (N_n \vee \mathcal{X}'))\rangle \end{array}$$

avec $\mathcal{X} = ((\neg x_1 \vee P_1) \wedge (x_1 \vee N_1)), \mathcal{X}' = ((\neg x_1 \vee P'_1) \wedge (x_1 \vee N'_1))$

et $Q' = q_2x_2 \dots q_nx_n$

$$\langle Q' \mid (\mathcal{P}_2, \mathcal{N}_2); \dots; (\mathcal{P}_n, \mathcal{N}_n)\rangle = \langle Q' \mid (P_2, N_2); \dots; (P_n, N_n)\rangle \otimes \langle Q' \mid (P'_2, N'_2); \dots; (P'_n, N'_n)\rangle$$

Exemple 34 *En continuant l'exemple 27. Soient* $Q = \forall a \exists d \forall e \exists f$ *et les bases littérales*

$$bl_a = \langle Q \mid (\top, \bot); (\top, \bot); ((a \wedge d), (a \wedge d)); ((a \wedge d), (a \wedge d))\rangle$$

et $bl_{\neg a} = \langle Q \mid (\bot, \top); (\top, \bot); ((a \wedge \neg d), (a \wedge \neg d)); (\bot, \top)\rangle$.
Alors $(bl_a \otimes bl_{\neg a}) \approx \langle Q \mid (\top, \top); (\top, \bot); (d, d); ((a \wedge d), d)\rangle$.

Si n est le nombre de symboles propositionnels quantifiés du lieur et k est le nombre maximum de littéraux, de $\top$ et de $\bot$ des gardes participant à la compilation de deux bases littérales alors le nombre maximum de littéraux, de $\top$ et de $\bot$ de la base littérale résultat est borné par un polynôme[1] en $6kn^2$.

Cet opérateur est la contrepartie de la disjonction pour les QBF (Q un lieur et $bl, bl' \in \mathcal{BL}_Q$) : si $bl^* = QM$ et $bl'^* = QM'$ alors $(bl \otimes bl')^* = QM_\otimes$ avec $M_\otimes \equiv (M \vee M')$.

Nous présentons l'algorithme de recherche *compilation_par_recherche* qui compile une QBF en une base littérale optimale. Cet algorithme teste en premier lieu si le lieur est réduit à un unique quantificateur associé à son symbole propositionnel. Si c'est le cas et si $M \equiv \top$, contrairement à l'algorithme $certificat\_par\_recherche$, $(\top, \top)$ est retourner (puisque $\exists x \top \cong \exists x((\neg x \vee \top) \wedge (x \vee \top)))$ pour conserver les deux possibilités de modèle. S'il y a plus d'un quantificateur, puisque l'algorithme est un algorithme de recherche, le quantificateur le plus externe est considéré en premier. Selon la sémantique des QBF, s'il n'y a pas de modèle pour un (resp. les deux) appels récursifs alors il n'y a pas de modèle pour la QBF si le quantificateur est universel (resp. existentiel) ; si il y a un modèle pour les deux appels récursifs, et cela pour les deux quantificateurs, $\langle Q \mid (\top, \bot)\ ;\ grds(bl^+)\rangle \otimes \langle Q \mid (\bot, \top)\ ;\ grds(bl^-)\rangle$ est retourné.

Théorème 5 (correction de l'algorithme *compilation_par_recherche***)** *Soit* QM *une QBF.* *compilation_par_recherche*(Q, M) *retourne une base littérale* bl *telle que* $bl^* \cong QM$ *si* QM *est valide et retourne non_valide sinon.*

Exemple 35 (Compilation par recherche) *En continuant l'exemple 34. La base littérale*

$$\begin{aligned} bl_r &= compilation\_par\_recherche(\forall a \exists d \forall e \exists f, \neg\neg\neg((e \rightarrow d) \rightarrow \neg(f \rightarrow a))) \\ &\approx \langle \forall a \exists d \forall e \exists f \mid (\top, \top); (\top, \bot); (d, d); ((a \wedge d), d)\rangle \end{aligned}$$

est telle que $bl_r^* \cong \forall a \exists d \forall e \exists f \neg\neg\neg((e \rightarrow d) \rightarrow \neg(f \rightarrow a))$.

La base littérale générée par l'algorithme *compilation_par_recherche* est optimale.

Théorème 6 (optimalité de l'algorithme *compilation_par_recherche***)** *Soit* QM *une QBF valide alors* *compilation_par_recherche*(Q, M) *est une base littérale optimale.*

Algorithme 5 *compilation_par_recherche*

Entrée: Q : le lieur d'une QBF
Entrée: M : la matrice d'une QBF
Sortie: soit une base littérale soit *non_valide*

si $Q = qx$ **alors**
 si $q = \exists$ **alors**
  selon M **faire**
   cas $\top$: **retourner** $\langle \exists x \mid (\top, \top) \rangle$
   cas $\bot$: **retourner** *non_valide*
   cas x : **retourner** $\langle \exists x \mid (\top, \bot) \rangle$
   cas $\neg x$: **retourner** $\langle \exists x \mid (\bot, \top) \rangle$
  fin selon
 sinon
  si $M = \top$ **alors retourner** $\langle \forall x \mid (\top, \top) \rangle$ **sinon retourner** *non_valide* **fin si**
 fin si
sinon
 $Q = qxQ'$
 $bl^{\top} := compilation\_par\_recherche(Q', [x \leftarrow \top](M))$
 $bl^{\bot} := compilation\_par\_recherche(Q', [x \leftarrow \bot](M))$
 si $q = \exists$ **alors**
  si $bl^{\top} = non\_valide$ **et** $bl^{\bot} = non\_valide$ **alors retourner** *non_valide* **fin si**
  si $bl^{\top} = non\_valide$ **alors retourner** $\langle Q \mid (\bot, \top); grds(bl^{\bot}) \rangle$ **fin si**
  si $bl^{\bot} = non\_valide$ **alors retourner** $\langle Q \mid (\top, \bot); grds(bl^{\top}) \rangle$ **fin si**
  retourner $\langle Q \mid (\top, \bot); grds(bl^{\top}) \rangle \otimes \langle Q \mid (\bot, \top); grds(bl^{\bot}) \rangle$
 sinon
  si $bl^{\top} = non\_valide$ **ou** $bl^{\bot} = non\_valide$ **alors**
   retourner *non_valide*
  sinon
   retourner $\langle Q \mid (\top, \bot); grds(bl^{\top}) \rangle \otimes \langle Q \mid (\bot, \top); grds(bl^{\bot}) \rangle$
  fin si
 fin si
fin si

3.3.2 Algorithme d'élimination de quantificateurs et compilation

Nous étendons l'algorithme 3, *elimination_prenexe_FNC*, d'élimination de quantificateurs en un compilateur de QBF en bases littérales. Cette extension est immédiate grâce aux équivalences (cf. § 1.5.2) :

$$Q\exists x((x \vee M_+) \wedge (\neg x \vee M_-) \wedge M') \equiv Q((M_- \vee M_+) \wedge M')$$

et

$$Q\forall x((x \vee M_+) \wedge (\neg x \vee M_-) \wedge M') \equiv Q((M_- \wedge M_+) \wedge M')$$

[1] pour être exact : $2(3k+2)n^2 - 2(k+2)n$ calculé par induction sur la structure de l'opérateur.

Algorithme 6 *compilation_par_elimination*

Entrée: Q : le lieur d'une QBF sous FNC
Entrée: M : la matrice d'une QBF sous FNC
Entrée: $\langle Q' \mid G \rangle$: une base littérale
Sortie: soit une base littérale soit *non_valide*
  si $Q = \varepsilon$ **alors**
   si $M \equiv \top$ **alors retourner** $\langle Q' \mid G \rangle$ **sinon retourner** *non_valide* **fin si**
  sinon
   $Q = Q''qx$
   $M = ((\neg x \vee P) \wedge (x \vee N) \wedge M')$
   si $q = \exists$ **alors**
    $M_{rec} := (fnc((P \vee N)) \wedge M')$
    $bl_{rec} := \langle Q' \exists x \mid G; (P, N) \rangle$
    retourner $compilation\_par\_elimination(Q'', M_{rec}, bl_{rec})$
   sinon
    $M_{rec} := ((P \wedge N) \wedge M')$
    $bl_{rec} := \langle Q' \forall x \mid G; (\top, \top) \rangle$
    retourner $compilation\_par\_elimination(Q'', M_{rec}, bl_{rec})$
   fin si
  fin si

qui permettent de faire apparaître les gardes. De part la validité de la QBF compilée et l'interprétation d'une base littérale, il est valide d'associer aux symboles propositionnels universellement quantifiés des gardes $(\top, \top)$. Puisqu'un algorithme d'élimination de quantificateurs est un algorithme qui opère du quantificateur le plus interne au quantificateur le plus externe, les gardes sont introduites dans la base littérale à la fin de la séquence.

Théorème 7 (correction de l'algorithme *compilation_par_elimination***)** *Soit QM une QBF. $compilation\_par\_elimination(Q, M, \langle \varepsilon \mid \top \rangle)$ retourne une base littérale bl telle que $bl^* \cong QM$ si QM est valide et retourne non_valide sinon.*

Exemple 36 (compilation par élimination) *En continuant l'exemple 27 (fnc_μ FNC de $\neg\neg\neg((e \rightarrow d) \rightarrow \neg(f \rightarrow a))$). La base littérale*

$$\begin{aligned} bl_e &= compilation\_par\_elimination(\forall a \exists d \forall e \exists f, fnc_\mu, \langle \varepsilon \mid \top \rangle) \\ &\approx \quad \langle \forall a \exists d \forall e \exists f \mid (\top, \top); (\top, \bot); (\top, \top); ((a \wedge (d \vee \neg e)), (d \vee \neg e)) \rangle \end{aligned}$$

est telle que $bl_e^* \cong \forall a \exists d \forall e \exists f \neg\neg\neg((e \rightarrow d) \rightarrow \neg(f \rightarrow a))$.

La base littérale générée par l'algorithme *compilation_par_elimination* est optimale.

Théorème 8 (optimalité de l'algorithme *compilation_par_elimination***)** *Soit QM une QBF valide alors $compilation\_par\_elimination(Q, M, \langle \varepsilon \mid \top \rangle)$ est une base littérale optimale.*

3.3.3 Comparaison des deux algorithmes de compilation

Les deux algorithmes de compilation n'ont pas en entrée le même fragment de langage mais ce n'est pas un obstacle rédhibitoire pour l'algorithme par élimination de quantificateurs : il suffit d'être en mesure de transformer la matrice de la QBF en une matrice équivalente sous la forme $((\neg x \vee P) \wedge (x \vee N) \wedge M')$, P, N et M' ne contenant pas le symbole propositionnel x associé au quantificateur le plus interne de la QBF [207]. Les deux algorithmes de compilation ne construisent évidemment pas les mêmes bases qui ne sont pas non plus équivalentes mais elles sont toutes les deux optimales.

Exemple 37 (comparaison) *En continuant les exemples 35 et 36. Les bases littérales bl_r et bl_e issues respectivement de la compilation par recherche et de la compilation par élimination ne sont pas équivalentes. En propageant que le symbole propositionnel existentiellement quantifié d ne peut être valué qu'à* **vrai** *(puisque $N_d = \bot$ dans bl_r), nous obtenons la base littérale*

$$\langle \forall a \exists d \forall e \exists f \mid (\top, \top); (\top, \bot); (\top, \top); (a, \top) \rangle$$

qui avait déjà été évoquée dans l'exemple 31 sur la minimalité, cette dernière étant particulièrement compacte.

3.4 Commentaires

Nous avons présenté dans ce chapitre un formalisme « au dessus » des QBF qui permet d'unifier deux problématiques : le calcul d'un certificat pour un algorithme de recherche via un **sat**-certificat, qui n'est autre qu'une représentation d'un modèle, et la compilation d'une QBF prénexe.

Les bases littérales ont donné lieu à deux implantations. La première, qui est en *Prolog*, reprend les définitions dans un esprit de vérification et d'appui aux exemples et démonstrations. Elle est aussi au cœur du système de génération de code du chapitre 4. Le code Prolog fait appel à une librairie de résolution de contraintes booléennes pour calculer par énumération les modèles propositionnels pour obtenir des gardes sous FNC et réduire la taille de celles-ci ; il peut être obtenu à l'adresse
`http://www.info.univ-angers.fr/pub/stephan/Research/QBF/LITERAL_BASES/literal_bases.html`.
La seconde implantation est une extension de l'algorithme 2 de recherche, *recherche_prenexe*, sur la base d'une implantation en C/C++ de l'algorithme décrit au chapitre 4 et réalisée par Benoit Da Mota. Cette implantation fait appel à une librairie de gestion des *BDD*, la librairie CUDD [205] ; la gestion de la pile de retour-arrière étant effectuée par recopie du BDD hors des primitives de la librairie, l'implantation n'est pas satisfaisante. Nous y remédions, en collaboration avec Vincent Barichard, en procédant à la refonte du calcul des **sat**-certificats et de la compilation au sein de l'architecture ouverte de propagation de contraintes GeCode [198]. Nous représentons la base littérale comme ensemble de contraintes générées par la recherche et simplifiées par propagation.

Chapitre 4

Procédure de décision par propagation

Il y eut un soir, il y eut un matin : quatrième jour.

[210] I. Stéphan. Propagation logique pour les formules booléennes quantifiées. In *Deuxièmes Journées Francophones de Programmation par Contraintes*, 2006.

[209] I. Stéphan. Boolean Propagation Based on Literals for Quantified Boolean Formulae. In *Proceedings of the 17th European Conference on Artificial Intelligence (ECAI'06)*, 2006.

[211] I. Stéphan. Une (presque) génération automatique d'un compilateur de tables de vérité vers un solveur pour formules booléennes quantifiées prénexes. In *Deuxièmes Journées Francophones de Programmation par Contraintes*, 2007.

Sommaire

Le fragment syntaxique complet le plus exploité dans le cadre de l'implantation des procédures de décision pour le problème de validité des QBF est le fragment des formules sous FNC (cf. § 1.3.3). Les raisons de cette position dominante sont claires [227] : (i) la plus grande part des procédures de décision pour le problème SAT prennent en entrée des *formules propositionnelles* sous FNC ; (ii) les structures de données et les algorithmes efficaces pour ces formules sont bien connus ; (iii) historiquement, les premiers algorithmes pour le problème de validité des QBF, tels que la *Q-résolution* [120] et la procédure de décision `Evaluate` [46], ont été pensées pour les formules propositionnelles sous FNC ; (iv) comme la grande majorité des procédures de décision et des exemples de référence sont pour le fragment FNC, les nouveaux venus, pour se comparer, n'ont guère d'autre choix que de venir aussi sur ce fragment. Cette analyse nous la faisons nôtre et ne nous soumettons pas à la conclusion.

Dans bon nombre des applications des QBF, la génération de celles-ci, considérée comme un programme, est en partie automatique, répondant à des arguments de *complétude* (et *correction*) de la sémantique des QBF vis-à-vis des formalismes dans lesquels sont exprimées les applications. Autant pour la compilation vers les QBF, le fragment hégémonique des formules sous FNC prénexes peut se discuter dans son principe, puisqu'il n'y aura pas à comprendre une QBF ainsi générée mais seulement, pour un spécialiste, à maîtriser sa génération, autant pour une programmation en QBF, le fragment FNC prénexe est abscons. Faut il alors considéré les QBF uniquement comme un langage vers lequel un autre formalisme est compilé en des QBF sous FNC prénexes dont seule la sémantique serait nécessaire et cela pour une efficacité maximale ? Pourquoi pas, si ce n'est que ce fragment apparaît de moins en moins comme étant le plus susceptible de remplir une telle mission : si le fragment FNC est bien adapté à la structure du problème SAT et des propriétés de la quantification existentielle, il ne rend pas compte des propriétés de la quantification universelle pour laquelle le fragment FND est mieux adapté [136, 227, 220, 190], de plus, la mise sous FNC introduit, dans le meilleur des cas, un ensemble de nouveaux symboles propositionnels existentiellement quantifiés dont le statut de symbole aux propriétés fonctionnelles n'est pas transmis à la procédure de décision.

Pour traduire par « renommage de formules » une QBF quelconque en une QBF sous FNC, trois transformations entrent en jeu (cf. § 1.3.3) : la linéarisation et la mise sous forme prénexe que nous aborderons dans le chapitre 6 et la mise sous FNC (propositionnelle) de la matrice. Dans ce chapitre, nous prenons le parti de considérer des QBF prénexes dont la matrice est une formule propositionnelle quelconque. L'objectif est de *ne pas* pratiquer la linéarisation qui induit, dans le pire des cas, une croissance exponentielle de la formule et de l'espace de recherche, et au contraire, de tirer pleinement bénéfice des propriétés algébriques des connecteurs binaires et quantificateurs. Si les connecteurs dits « secondaires », qui sont la bi-implication et le ou-exclusif, et qui sont éliminés par la linéarisation, ne sont pas considérés, le problème est réduit à traiter des QBF sous FNC contenant trois littéraux par clause.

La démarche qui prévaut à cette étude est d'étendre le système syntaxique S_{QBF} du chapitre 1 qui est orienté « validité » vers une analyse systématique des propriétés de la combinaison des connecteurs et des quantificateurs pour obtenir un algorithme traitant de manière symétrique la validité et la non-validité. Cette dualité est l'essence même de la négation qui disparaît, en apparence, du système S_{QBF} pour réapparaître sous la forme fonctionnelle du conjugué et aboutit à passer d'une méthode « par renommage de formules » à une « décomposition par introduction de littéraux existentiellement quantifiés » par quoi nous poursuivons ce chapitre. La démarche est aussi dans la systématisation de la recherche des propriétés algébriques et de leurs conséquences, mises sous la forme de *règles*, calculées grâce aux bases littérales du chapitre 3. Les règles ainsi obtenues peuvent être compilées avec des fonctions auxiliaires en un ensemble de fonctions de propagation pour un langage de programmation cible, pour être ensuite insérées dans un algorithme permettant d'atteindre la complétude. Cette chaîne de transformations partant des définitions des connecteurs sous la forme de tables de vérité pour terminer par une procédure de décision dans un langage de programmation cible est illustrée dans la figure 4.1.

4.1 Introduction de littéraux existentiellement quantifiés

La décomposition que réalise la méthode « par renommage de formules » introduit des symboles propositionnels existentiellement quantifiés et conserve la négation comme connecteur sur lequel s'applique la décomposition. Ainsi une formule telle que $(\neg x \rightarrow x) \equiv z$ qui permet de déduire $x \equiv z$ ne peut être prise en compte par une méthode de propagation basée sur cette décomposition car elle est décomposée en $\neg x \equiv z'$ et $(z' \rightarrow x) \equiv z$.

Algorithme 7 *decomposition*$_\exists$

Entrée: Une formule propositionnelle F
Entrée: Un entier n
Sortie: Une paire composée d'un lieur et d'une formule propositionnelle
  si $F = \top$ ou $F = \bot$ ou $F = \neg x$ ou $F = x$ avec $x \in \mathcal{SP}$ **alors**
    retourner (ε, F)
  sinon
    $F = [z_n \leftarrow (x \circ y)](G)$, avec $z_n, |x|, |y| \in \mathcal{SP}$
    $(Q, D) = decomposition_\exists(G, n+1)$
    retourner $(Q \exists z_n, (((x \circ y) \leftrightarrow z_n) \wedge D))$
  fin si

L'algorithme 7, *decomposition*$_\exists$, définit la décomposition d'une QBF prénexe QM par introduction de littéraux existentiellement quantifiés. Cette décomposition étend naturellement la décomposition par introduction de symboles propositionnels existentiellement quantifiés en faisant disparaître la négation explicite au profit des complémentaires. La décomposition ne s'applique alors que sur des formules qui ont subi jusqu'à l'obtention d'un point fixe la suppression des doubles négations grâce à l'*équivalence logique* exprimant le caractère involutif de la négation : $\neg\neg F \overset{0.4}{\equiv} F$. L'introduction de tels symboles

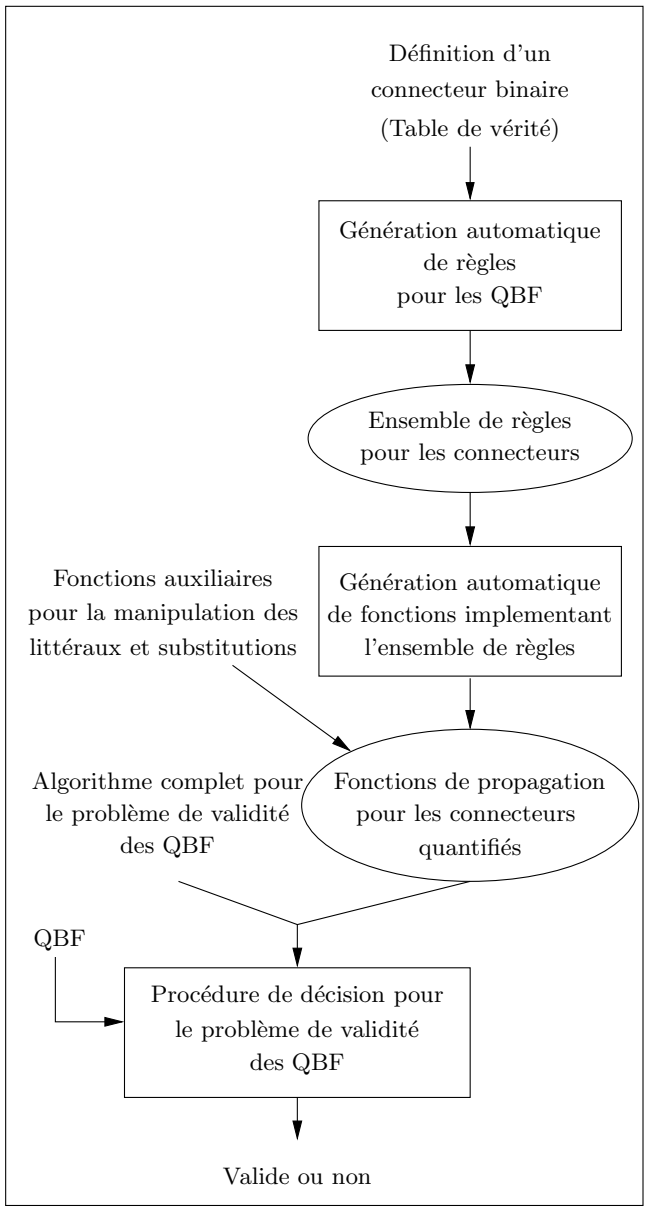


Fig. 4.1 – Chaîne de transformations de la table de vérité à un code pour un langage cible.

propositionnels existentiellement quantifiés ne modifie pas la validité de la QBF initiale : si $(Q_\exists, D) = \mathit{decomposition}_\exists(M, 1)$ alors $QM \equiv QQ_\exists D$ (cf. Démonstrations, lemme 35).

De manière duale, un algorithme par introduction de littéraux universellement quantifiés est possible et ces deux algorithmes peuvent être dérécursivés pour obtenir des algorithmes itératifs.

Exemple 38 (decomposition) *En continuant l'exemple 1. La QBF*

$$\xi = \forall a \exists d \forall e \exists f \neg\neg\neg((e \rightarrow d) \rightarrow \neg(f \rightarrow a))$$

est décomposée existentiellement, par l'appel

$$\mathit{decomposition}_\exists(\neg((e \rightarrow d) \rightarrow \neg(f \rightarrow a)), 1)$$

après avoir éliminé les double-négations, en la QBF

$$\begin{array}{ll} \xi \equiv & \forall a \exists d \forall e \exists f \exists z_3 \exists z_2 \exists z_1 ((e \rightarrow d) \leftrightarrow z_1) \wedge ((f \rightarrow a) \leftrightarrow z_2) \wedge ((z_1 \rightarrow \overline{z_2}) \leftrightarrow z_3) \wedge \overline{z_3} \\ \equiv & \forall a \exists d \forall e \exists f \exists z_3 \exists z_2 \exists z_1 ((e \rightarrow d) \leftrightarrow z_1) \wedge ((f \rightarrow a) \leftrightarrow z_2) \wedge ((z_1 \rightarrow \overline{z_2}) \leftrightarrow \bot) \end{array}$$

et universellement en la QBF

$$\begin{array}{ll} \xi \cong & \forall a \exists d \forall e \exists f \forall z_3 \forall z_2 \forall z_1 (((e \rightarrow d) \leftrightarrow z_1) \rightarrow (((f \rightarrow a) \leftrightarrow z_2) \rightarrow (((z_1 \rightarrow \overline{z_2}) \leftrightarrow z_3) \rightarrow \overline{z_3}))) \\ \cong & \forall a \exists d \forall e \exists f \forall z_3 \forall z_2 \forall z_1 (((e \rightarrow d) \leftrightarrow z_1) \wedge ((f \rightarrow a) \leftrightarrow z_2) \wedge ((z_1 \rightarrow \overline{z_2}) \leftrightarrow z_3) \rightarrow \overline{z_3}) \end{array}$$

La décomposition suggère une nouvelle forme normale utilisant conjonctivement (ou disjonctivement) la brique de base sur des littéraux $((x \circ y) \leftrightarrow z)$ que nous définissons ainsi.

Définition 42 (contraintes et formes normales de contraintes) *Une* contrainte *est une formule propositionnelle de la forme* $((x \circ y) \leftrightarrow z)$ *avec* $x \in \{\top, \bot\}$ *ou bien* $|x| \in \mathcal{SP}$, $y \in \{\top, \bot\}$ *ou bien* $|y| \in \mathcal{SP}$ *et* $z \in \{\top, \bot\}$ *ou bien* $|z| \in \mathcal{SP}$. *Une* contrainte quantifiée *est une QBF close prénexe dont la matrice est une contrainte. L'ensemble des contraintes quantifiées est notée* $\mathcal{C}$. *Une QBF close prénexe est* sous forme normale de contraintes *si c'est une QBF de la forme* $Q \bigwedge_{1 \leq i \leq n} K_i$ *avec* K_i *des contraintes.*

Le choix du terme *contrainte* n'est pas un hasard, il fait référence à la résolution de *problèmes de satisfaction de contraintes* ; nous aborderons les liens dans la dernière section de ce chapitre.

La complétude, aussi bien pour ce qui est de la validité que de la préservation des modèles, de ce fragment QBF est immédiate grâce aux algorithmes de décomposition.

Il existe des QBF prénexes sous forme normale de contraintes qui ne sont le résultat d'aucune décomposition, par exemple la QBF $\forall a \exists c \exists d((c \vee d) \leftrightarrow \neg a)$. La bi-implication a deux statuts dans la forme normale de contraintes : il est un connecteur binaire à l'instar de la conjonction ou du ou-exclusif et il est constitutif du motif de la contrainte.

4.2 Schémas et règles

La décomposition qui permet de transformer, tout en préservant les modèles propositionnels, une QBF prénexe quelconque en une QBF sous forme normale de contraintes conserve les connecteurs binaires et ne rompt pas la structure de la QBF initiale. La forme normale de contraintes nous permet d'exploiter localement, au niveau de la contrainte quantifiée les propriétés des quantificateurs puis de propager le caractère nécessaire des substitutions à l'ensemble de la QBF.

Théorème 9 *Soit une QBF sous forme normale de contraintes* $QM = Q \bigwedge_{1 \leq i \leq n} K_i$. *Si* QM *est valide alors toutes les contraintes quantifiées* $Q|_{\mathcal{SP}(K_i)} K_i$, *pour* $1 \leq i \leq n$, *sont valides.*

Il est particulièrement intéressant de remarquer que la réciproque du théorème 9 est fausse comme le démontre l'exemple suivant.

Exemple 39 *Les contraintes quantifiées* $\forall d \exists a \exists e((\neg a \wedge \neg e) \leftrightarrow d)$ *et* $\forall d \exists a \exists e((a \vee e) \leftrightarrow d)$ *sont valides mais la QBF sous forme normale de contraintes* $\forall d \exists a \exists e(((\neg a \wedge \neg e) \leftrightarrow d) \wedge ((a \vee e) \leftrightarrow d))$ *ne l'est pas.*

Le théorème 9 offre implicitement un algorithme de propagation des substitutions nécessaires à la validité : si une seule des contraintes quantifiées est non-valide alors la QBF est non-valide ; si un symbole propositionnel quantifié existentiellement est contraint à être équivalent à une constante logique ou à un littéral alors ceci se propage à l'ensemble de la QBF.

Nous abordons d'abord quelques cas de contraintes quantifiées exemplaires pour ensuite en extraire des *schémas* généraux exhaustifs. Le cas le plus simple est lorsque la contrainte quantifiée entraîne la non-validité de la QBF : il en est ainsi lorsque la contrainte quantifiée ne contient que des constantes logiques ne satisfaisant pas la sémantique du connecteur logique, par exemple $((\bot \rightarrow \bot) \leftrightarrow \bot)$; il en est de même lorsqu'un symbole propositionnel existentiellement quantifié ne peut plus être valué à aucune valeur de vérité possible pour rendre la contrainte quantifiée valide, par exemple $\exists d((\top \rightarrow d) \leftrightarrow \overline{d})$; il en est aussi de même lorsqu'un symbole propositionnel universellement quantifié est forcé par la sémantique du connecteur logique à être valué à une unique valeur de vérité, par exemple $\forall e((\top \rightarrow \bot) \leftrightarrow e)$; enfin il peut y avoir des cas plus complexes qui prennent en compte un mélange des cas précédents, tel que par exemple $\exists a \exists d \forall e((a \rightarrow d) \leftrightarrow e)$. Un autre cas simple est lorsque la contrainte quantifiée ne peut être que valide car la contrainte est une *tautologie* : il en est ainsi lorsque la contrainte quantifiée ne contient que des constantes logiques satisfaisant la sémantique du connecteur logique, par exemple $((\bot \rightarrow \bot) \leftrightarrow \top)$; il en est de même lorsque les symboles propositionnels n'influencent plus la valeur de vérité de la contrainte, par exemple puisque $((\bot \rightarrow d) \leftrightarrow \top) \equiv \top$ les contraintes quantifiées $\exists d((\bot \rightarrow d) \leftrightarrow \top)$ et $\forall d((\bot \rightarrow d) \leftrightarrow \top)$ ne peuvent être que valides (ceci est à rapprocher de la *propagation « don't care »*). La propagation apparaît lorsque la validité de la contrainte quantifiée force par la sémantique du connecteur logique les symboles propositionnels (existentiellement quantifiés) à être valués soit à une constante soit à la même valeur de vérité

(ou la valeur de vérité complémentaire) qu'un autre symbole propositionnel (quantifié existentiellement ou non), par exemple $\exists d \exists e \forall a((a \rightarrow d) \leftrightarrow e)$ force d et e à la valeur de vérité **vrai**, de même $\exists a \forall e \exists d((a \rightarrow d) \leftrightarrow e)$ force a à la valeur de vérité **vrai** et d, fonctionnellement, à la valeur de vérité de e. Ces derniers exemples montrent s'il en était besoin l'importance de l'ordre des quantificateurs dans le lieur de la contrainte quantifiée. Forcer au niveau sémantique un symbole propositionnel revient au niveau syntaxique à réaliser une substitution et éliminer le quantificateur associé au symbole propositionnel substitué. Dans ces deux exemples, une fois les substitutions réalisées, les contraintes sont des tautologies ; ce n'est pas nécessairement le cas comme dans l'exemple $\exists e \forall a \exists d((a \rightarrow d) \leftrightarrow e)$ qui ne force qu'uniquement e à la valeur de vérité **vrai**. De ces schémas et de leurs conséquences, nous extrayons des règles.

Définition 43 (schémas et règles) *Un* schéma *est une séquence de symboles de la forme* $\mathcal{Q}((X \circ Y) \leftrightarrow Z)$ *avec* $X \in \{x, \top, \bot\}$, $Y \in \{y, x, \overline{x}, \top, \bot\}$, $Z \in \{z, y, \overline{y}, x, \overline{x}, \top, \bot\}$, $\mathcal{Q}$ *une permutation de (s'ils sont définis)* $q_x|x|$ *si* $X \in \{x, \overline{x}\}$, $q_y|y|$ *si* $Y \in \{y, \overline{y}\}$ *et* $q_z|z|$ *si* $Z = z$ *avec* $q_x, q_y, q_z \in \{\exists, \forall\}$. *Par abus de langage,* $((X \circ Y) \leftrightarrow Z)$ *et* $\mathcal{Q}$ *sont aussi appelés respectivement matrice et lieur. La fonction* τ *associe un schéma à une contrainte quantifiée si celle-ci est une instance de celui-là.*

Un schéma est contradictoire *si toute contrainte quantifiée, instance de ce schéma, est non-valide. Tous les autres schémas sont tels que toute contrainte quantifiée instance de ce schéma admet un modèle QBF. Un schéma est* tautologique *si pour toute contrainte quantifiée, instance de ce schéma, la matrice est une tautologie. Un schéma est* contingent *si pour toute contrainte quantifiée, instance de ce schéma, aucun des symboles propositionnels (existentiellement quantifiés) n'est forcé à être valué à une valeur vérité ou fonctionnellement à la valeur de vérité d'un autre symbole propositionnel. Un schéma induit une* règle de propagation/simplification *si pour toute contrainte quantifiée, instance de ce schéma, tous les symboles propositionnels (existentiellement quantifiés) sont forcés à être valués à des valeurs de vérité ou fonctionnellement aux valeurs de vérité d'autres symboles propositionnels. Un schéma induit une* règle de propagation *si pour toute contrainte quantifiée, instance de ce schéma, une partie des symboles propositionnels (existentiellement quantifiés) sont forcés à être valués à des valeurs de vérité ou fonctionnellement aux valeurs de vérité d'autres symboles propositionnels. Une règle associe à un schéma les couples constitués d'un symbole propositionnel forcé et de sa valeur exprimés sous la forme de substitutions.*

Exemple 40 (schémas et règles)
- *Le schéma* $\exists|x|\exists|y|\forall|z|((x \rightarrow y) \leftrightarrow z)$ *est un schéma contradictoire ;*
- *le schéma* $\exists|y|((\bot \rightarrow y) \leftrightarrow \top)$ *est un schéma tautologique ;*
- *le schéma* $\forall|z|\exists|x|\exists|y|((x \rightarrow y) \leftrightarrow z)$ *est un schéma contingent ;*
- *au schéma* $\exists|x|\forall|z|\exists|y|((x \rightarrow y) \leftrightarrow z)$ *est associé la substitution* $[x \leftarrow \top]$ $[y \leftarrow z]$ *au sein d'une règle de propagation/simplification ;*
- *au schéma* $\exists|z|\forall|x|\exists|y|((x \rightarrow y) \leftrightarrow z)$ *est associé la substitution* $[z \leftarrow \top]$ *au sein d'une règle de propagation.*

Les règles sont correctes pour l'équivalence qui préserve les modèles QBF (cf. Démonstrations, lemme 38) et l'argument de correction qui suit met en exergue que la propriété locale d'un schéma s'étend globalement à la QBF contenant conjonctivement une contrainte quantifiée suivant un tel schéma (K une contrainte, Q un lieur et F une formule propositionnelle) :

1. Si $Q|_{\mathcal{SP}(K)}K$ est une instance d'un schéma contradictoire alors $Q(K \wedge F)$ est non-valide.
2. Si $Q|_{\mathcal{SP}(K)}K$ est une instance d'un schéma tautologique alors $Q(K \wedge F) \cong QF$.
3. Si $Q|_{\mathcal{SP}(K)}K$ est une instance d'un schéma associé à une substitution instanciée sur les symboles propositionnels de K en une substitution σ au sein d'une règle de propagation/simplification alors $Q(K \wedge F) \cong Q(\Sigma \wedge \sigma(F))$ avec Σ la formule propositionnelle associée à la substitution.
4. Si $Q|_{\mathcal{SP}(K)}K$ est une instance d'un schéma associé à une substitution instanciée sur les symboles propositionnels de K en une substitution σ au sein d'une règle de propagation alors $Q(K \wedge F) \cong Q(\Sigma \wedge \sigma((K \wedge F)))$ avec Σ la formule propositionnelle associée à la substitution.

La définition 43 autorise des schémas dits « redondants » car possédant des instances communes.

Exemple 41 (Schémas redondants) *Les schémas* $\forall|y|\forall|z|((\top \circ y) \leftrightarrow z)$ *et* $\forall|x|\forall|y|((\top \circ \overline{x}) \leftrightarrow \overline{y})$ *sont redondants car la contrainte quantifiée* $\forall a \forall d((\top \circ \neg a) \leftrightarrow \neg d)$ *est une instance de ces deux schémas par les substitutions* $[y \leftarrow \neg a][z \leftarrow \neg d]$ *et* $[x \leftarrow a][y \leftarrow d]$*, respectivement.*

Il existe deux cent-huit schémas non-redondants qui sont explicités dans les tables 4.1 à 4.5 qui contiennent les schémas contradictoires, les règles de propagation/simplification, les règles de propagation, les schémas tautologiques, et les schémas contingents pour le connecteur d'implication. Par dualité et sans surprise, le nombre de schémas contradictoires représentent la moitié de la totalité des schémas non-redondants.

Chaque contrainte quantifiée est une instance d'un unique schéma non-redondant et tout schéma admet au moins une instance donc les contraintes quantifiées peuvent être partitionnées en des classes d'équivalence pour la substitution par rapport à un schéma. Nous formalisons cette remarque par la définition d'une relation d'équivalence sur les contraintes quantifiées.

Définition 44 (relation $\sim$) *Soit c et c' deux contraintes quantifiées. $c \sim c'$ s'il existe un schéma non-redondant s tel que $\tau(c) = s$ et $\tau(c') = s$.*

La relation $\sim$ est une relation d'équivalence et dans la suite nous confondons, d'un côté, la classe d'équivalence avec le schéma non-redondant dont toutes les instances sont membres et, d'un autre côté, $\mathcal{C}_{/\sim}$ avec l'ensemble des schémas non-redondants (cf. Démonstrations, lemme 39).

TAB. 4.1 – Schémas contradictoires

$((\bot\to\bot)\leftrightarrow\bot)$	$\forall\|z\|((\bot\to\bot)\leftrightarrow z)$	$((\bot\to\top)\leftrightarrow\bot)$
$\forall\|z\|((\bot\to\top)\leftrightarrow z)$	$\exists\|y\|((\bot\to y)\leftrightarrow\bot)$	$\forall\|y\|((\bot\to y)\leftrightarrow\bot)$
$\forall\|y\|((\bot\to y)\leftrightarrow y)$	$\forall\|y\|((\bot\to y)\leftrightarrow\overline{y})$	$\exists\|y\|\forall\|z\|((\bot\to y)\leftrightarrow z)$
$\forall\|y\|\forall\|z\|((\bot\to y)\leftrightarrow z)$	$\forall\|z\|\exists\|y\|((\bot\to y)\leftrightarrow z)$	$\forall\|z\|\forall\|y\|((\bot\to y)\leftrightarrow z)$
$((\top\to\bot)\leftrightarrow\top)$	$\forall\|z\|((\top\to\bot)\leftrightarrow z)$	$((\top\to\top)\leftrightarrow\bot)$
$\forall\|z\|((\top\to\top)\leftrightarrow z)$	$\forall\|y\|((\top\to y)\leftrightarrow\bot)$	$\forall\|y\|((\top\to y)\leftrightarrow\top)$
$\exists\|y\|((\top\to y)\leftrightarrow\overline{y})$	$\forall\|y\|((\top\to y)\leftrightarrow\overline{y})$	$\exists\|y\|\forall\|z\|((\top\to y)\leftrightarrow z)$
$\forall\|y\|\forall\|z\|((\top\to y)\leftrightarrow z)$	$\exists\|z\|\forall\|y\|((\top\to y)\leftrightarrow z)$	$\forall\|z\|\forall\|y\|((\top\to y)\leftrightarrow z)$
$\forall\|x\|((x\to\bot)\leftrightarrow\bot)$	$\forall\|x\|((x\to\bot)\leftrightarrow\top)$	$\exists\|x\|((x\to\bot)\leftrightarrow x)$
$\forall\|x\|((x\to\bot)\leftrightarrow x)$	$\exists\|x\|\forall\|z\|((x\to\bot)\leftrightarrow z)$	$\forall\|x\|\forall\|z\|((x\to\bot)\leftrightarrow z)$
$\exists\|z\|\forall\|x\|((x\to\bot)\leftrightarrow z)$	$\forall\|z\|\forall\|x\|((x\to\bot)\leftrightarrow z)$	$\exists\|x\|((x\to\top)\leftrightarrow\bot)$
$\forall\|x\|((x\to\top)\leftrightarrow\bot)$	$\forall\|x\|((x\to\top)\leftrightarrow x)$	$\forall\|x\|((x\to\top)\leftrightarrow\overline{x})$
$\exists\|x\|\forall\|z\|((x\to\top)\leftrightarrow z)$	$\forall\|x\|\forall\|z\|((x\to\top)\leftrightarrow z)$	$\forall\|z\|\exists\|x\|((x\to\top)\leftrightarrow z)$
$\forall\|z\|\forall\|x\|((x\to\top)\leftrightarrow z)$	$\exists\|x\|((x\to x)\leftrightarrow\bot)$	$\forall\|x\|((x\to x)\leftrightarrow\bot)$
$\forall\|x\|((x\to x)\leftrightarrow x)$	$\forall\|x\|((x\to x)\leftrightarrow\overline{x})$	$\exists\|x\|\forall\|z\|((x\to x)\leftrightarrow z)$
$\forall\|x\|\forall\|z\|((x\to x)\leftrightarrow z)$	$\forall\|z\|\exists\|x\|((x\to x)\leftrightarrow z)$	$\forall\|z\|\forall\|x\|((x\to x)\leftrightarrow z)$
$\forall\|x\|((x\to\overline{x})\leftrightarrow\bot)$	$\forall\|x\|((x\to\overline{x})\leftrightarrow\top)$	$\exists\|x\|((x\to\overline{x})\leftrightarrow x)$
$\forall\|x\|((x\to\overline{x})\leftrightarrow x)$	$\exists\|x\|\forall\|z\|((x\to\overline{x})\leftrightarrow z)$	$\forall\|x\|\forall\|z\|((x\to\overline{x})\leftrightarrow z)$
$\exists\|z\|\forall\|x\|((x\to\overline{x})\leftrightarrow z)$	$\forall\|z\|\forall\|x\|((x\to\overline{x})\leftrightarrow z)$	$\exists\|x\|\forall\|y\|((x\to y)\leftrightarrow\bot)$
$\forall\|x\|\exists\|y\|((x\to y)\leftrightarrow\bot)$	$\forall\|x\|\forall\|y\|((x\to y)\leftrightarrow\bot)$	$\exists\|y\|\forall\|x\|((x\to y)\leftrightarrow\bot)$
$\forall\|y\|\exists\|x\|((x\to y)\leftrightarrow\bot)$	$\forall\|y\|\forall\|x\|((x\to y)\leftrightarrow\bot)$	$\forall\|x\|\forall\|y\|((x\to y)\leftrightarrow\top)$
$\forall\|y\|\forall\|x\|((x\to y)\leftrightarrow\top)$	$\exists\|x\|\forall\|y\|((x\to y)\leftrightarrow x)$	$\forall\|x\|\exists\|y\|((x\to y)\leftrightarrow x)$
$\forall\|x\|\forall\|y\|((x\to y)\leftrightarrow x)$	$\exists\|y\|\forall\|x\|((x\to y)\leftrightarrow x)$	$\forall\|y\|\exists\|x\|((x\to y)\leftrightarrow x)$
$\forall\|y\|\forall\|x\|((x\to y)\leftrightarrow x)$	$\forall\|x\|\forall\|y\|((x\to y)\leftrightarrow\overline{x})$	$\forall\|y\|\forall\|x\|((x\to y)\leftrightarrow\overline{x})$
$\forall\|x\|\forall\|y\|((x\to y)\leftrightarrow y)$	$\forall\|y\|\forall\|x\|((x\to y)\leftrightarrow y)$	$\exists\|x\|\forall\|y\|((x\to y)\leftrightarrow\overline{y})$
$\forall\|x\|\exists\|y\|((x\to y)\leftrightarrow\overline{y})$	$\forall\|x\|\forall\|y\|((x\to y)\leftrightarrow\overline{y})$	$\exists\|y\|\forall\|x\|((x\to y)\leftrightarrow\overline{y})$
$\forall\|y\|\exists\|x\|((x\to y)\leftrightarrow\overline{y})$	$\forall\|y\|\forall\|x\|((x\to y)\leftrightarrow\overline{y})$	$\exists\|x\|\exists\|y\|\forall\|z\|((x\to y)\leftrightarrow z)$
$\exists\|x\|\forall\|y\|\forall\|z\|((x\to y)\leftrightarrow z)$	$\forall\|x\|\exists\|y\|\forall\|z\|((x\to y)\leftrightarrow z)$	$\forall\|x\|\forall\|y\|\forall\|z\|((x\to y)\leftrightarrow z)$
$\exists\|x\|\forall\|z\|\forall\|y\|((x\to y)\leftrightarrow z)$	$\forall\|x\|\exists\|z\|\forall\|y\|((x\to y)\leftrightarrow z)$	$\forall\|x\|\forall\|z\|\exists\|y\|((x\to y)\leftrightarrow z)$
$\forall\|x\|\forall\|z\|\forall\|y\|((x\to y)\leftrightarrow z)$	$\exists\|y\|\exists\|x\|\forall\|z\|((x\to y)\leftrightarrow z)$	$\exists\|y\|\forall\|x\|\forall\|z\|((x\to y)\leftrightarrow z)$
$\forall\|y\|\exists\|x\|\forall\|z\|((x\to y)\leftrightarrow z)$	$\forall\|y\|\forall\|x\|\forall\|z\|((x\to y)\leftrightarrow z)$	$\exists\|y\|\forall\|z\|\forall\|x\|((x\to y)\leftrightarrow z)$
$\forall\|y\|\exists\|z\|\forall\|x\|((x\to y)\leftrightarrow z)$	$\forall\|y\|\forall\|z\|\exists\|x\|((x\to y)\leftrightarrow z)$	$\forall\|y\|\forall\|z\|\forall\|x\|((x\to y)\leftrightarrow z)$
$\exists\|z\|\forall\|x\|\forall\|y\|((x\to y)\leftrightarrow z)$	$\forall\|z\|\exists\|x\|\forall\|y\|((x\to y)\leftrightarrow z)$	$\forall\|z\|\forall\|x\|\exists\|y\|((x\to y)\leftrightarrow z)$
$\forall\|z\|\forall\|x\|\forall\|y\|((x\to y)\leftrightarrow z)$	$\exists\|z\|\forall\|y\|\forall\|x\|((x\to y)\leftrightarrow z)$	$\forall\|z\|\exists\|y\|\forall\|x\|((x\to y)\leftrightarrow z)$
$\forall\|z\|\forall\|y\|\exists\|x\|((x\to y)\leftrightarrow z)$	$\forall\|z\|\forall\|y\|\forall\|x\|((x\to y)\leftrightarrow z)$	

TAB. 4.2 – Règles de propagation/simplification

$\exists\|z\|((\bot\rightarrow\bot)\leftrightarrow z)$	$[z \leftarrow \top]$	$\exists\|z\|((\bot\rightarrow\top)\leftrightarrow z)$	$[z \leftarrow \top]$
$\exists\|y\|((\bot\rightarrow y)\leftrightarrow y)$	$[y \leftarrow \top]$	$\exists\|y\|((\bot\rightarrow y)\leftrightarrow \overline{y})$	$[y \leftarrow \bot]$
$\exists\|y\|\exists\|z\|((\bot\rightarrow y)\leftrightarrow z)$	$[z \leftarrow \top]$	$\forall\|y\|\exists\|z\|((\bot\rightarrow y)\leftrightarrow z)$	$[z \leftarrow \top]$
$\exists\|z\|\exists\|y\|((\bot\rightarrow y)\leftrightarrow z)$	$[z \leftarrow \top]$	$\exists\|z\|\forall\|y\|((\bot\rightarrow y)\leftrightarrow z)$	$[z \leftarrow \top]$
$\exists\|z\|((\top\rightarrow\bot)\leftrightarrow z)$	$[z \leftarrow \bot]$	$\exists\|z\|((\top\rightarrow\top)\leftrightarrow z)$	$[z \leftarrow \top]$
$\exists\|y\|((\top\rightarrow y)\leftrightarrow\bot)$	$[y \leftarrow \bot]$	$\exists\|y\|((\top\rightarrow y)\leftrightarrow\top)$	$[y \leftarrow \top]$
$\exists\|y\|\exists\|z\|((\top\rightarrow y)\leftrightarrow z)$	$[z \leftarrow y]$	$\forall\|y\|\exists\|z\|((\top\rightarrow y)\leftrightarrow z)$	$[z \leftarrow y]$
$\exists\|z\|\exists\|y\|((\top\rightarrow y)\leftrightarrow z)$	$[y \leftarrow z]$	$\forall\|z\|\exists\|y\|((\top\rightarrow y)\leftrightarrow z)$	$[y \leftarrow z]$
$\exists\|x\|((x\rightarrow\bot)\leftrightarrow\bot)$	$[x \leftarrow \top]$	$\exists\|x\|((x\rightarrow\bot)\leftrightarrow\top)$	$[x \leftarrow \bot]$
$\exists\|x\|\exists\|z\|((x\rightarrow\bot)\leftrightarrow z)$	$[z \leftarrow \overline{x}]$	$\forall\|x\|\exists\|z\|((x\rightarrow\bot)\leftrightarrow z)$	$[z \leftarrow \overline{x}]$
$\exists\|z\|\exists\|x\|((x\rightarrow\bot)\leftrightarrow z)$	$[x \leftarrow \overline{z}]$	$\forall\|z\|\exists\|x\|((x\rightarrow\bot)\leftrightarrow z)$	$[x \leftarrow \overline{z}]$
$\exists\|x\|((x\rightarrow\top)\leftrightarrow x)$	$[x \leftarrow \top]$	$\exists\|x\|((x\rightarrow\top)\leftrightarrow\overline{x})$	$[x \leftarrow \bot]$
$\exists\|x\|\exists\|z\|((x\rightarrow\top)\leftrightarrow z)$	$[z \leftarrow \top]$	$\forall\|x\|\exists\|z\|((x\rightarrow\top)\leftrightarrow z)$	$[z \leftarrow \top]$
$\exists\|z\|\exists\|x\|((x\rightarrow\top)\leftrightarrow z)$	$[z \leftarrow \top]$	$\exists\|z\|\forall\|x\|((x\rightarrow\top)\leftrightarrow z)$	$[z \leftarrow \top]$
$\exists\|x\|((x\rightarrow x)\leftrightarrow x)$	$[x \leftarrow \top]$	$\exists\|x\|((x\rightarrow x)\leftrightarrow\overline{x})$	$[x \leftarrow \bot]$
$\exists\|x\|\exists\|z\|((x\rightarrow x)\leftrightarrow z)$	$[z \leftarrow \top]$	$\forall\|x\|\exists\|z\|((x\rightarrow x)\leftrightarrow z)$	$[z \leftarrow \top]$
$\exists\|z\|\exists\|x\|((x\rightarrow x)\leftrightarrow z)$	$[z \leftarrow \top]$	$\exists\|z\|\forall\|x\|((x\rightarrow x)\leftrightarrow z)$	$[z \leftarrow \top]$
$\exists\|x\|((x\rightarrow\overline{x})\leftrightarrow\bot)$	$[x \leftarrow \top]$	$\exists\|x\|((x\rightarrow\overline{x})\leftrightarrow\top)$	$[x \leftarrow \bot]$
$\exists\|x\|\exists\|z\|((x\rightarrow\overline{x})\leftrightarrow z)$	$[z \leftarrow \overline{x}]$	$\forall\|x\|\exists\|z\|((x\rightarrow\overline{x})\leftrightarrow z)$	$[z \leftarrow \overline{x}]$
$\exists\|z\|\exists\|x\|((x\rightarrow\overline{x})\leftrightarrow z)$	$[x \leftarrow \overline{z}]$	$\forall\|z\|\exists\|x\|((x\rightarrow\overline{x})\leftrightarrow z)$	$[x \leftarrow \overline{z}]$
$\exists\|x\|\exists\|y\|((x\rightarrow y)\leftrightarrow\bot)$	$[x \leftarrow \top][y \leftarrow \bot]$	$\exists\|y\|\exists\|x\|((x\rightarrow y)\leftrightarrow\bot)$	$[y \leftarrow \bot][x \leftarrow \top]$
$\exists\|x\|\forall\|y\|((x\rightarrow y)\leftrightarrow\top)$	$[x \leftarrow \bot]$	$\exists\|y\|\forall\|x\|((x\rightarrow y)\leftrightarrow\top)$	$[y \leftarrow \top]$
$\exists\|x\|\exists\|y\|((x\rightarrow y)\leftrightarrow x)$	$[x \leftarrow \top][y \leftarrow \top]$	$\exists\|y\|\exists\|x\|((x\rightarrow y)\leftrightarrow x)$	$[y \leftarrow \top][x \leftarrow \top]$
$\exists\|x\|\forall\|y\|((x\rightarrow y)\leftrightarrow\overline{x})$	$[x \leftarrow \bot]$	$\exists\|y\|\forall\|x\|((x\rightarrow y)\leftrightarrow\overline{x})$	$[y \leftarrow \bot]$
$\exists\|x\|\forall\|y\|((x\rightarrow y)\leftrightarrow y)$	$[x \leftarrow \top]$	$\exists\|y\|\forall\|x\|((x\rightarrow y)\leftrightarrow y)$	$[y \leftarrow \top]$
$\exists\|x\|\exists\|y\|((x\rightarrow y)\leftrightarrow\overline{y})$	$[x \leftarrow \bot][y \leftarrow \bot]$	$\exists\|y\|\exists\|x\|((x\rightarrow y)\leftrightarrow\overline{y})$	$[y \leftarrow \bot][x \leftarrow \bot]$
$\exists\|x\|\exists\|z\|\forall\|y\|((x\rightarrow y)\leftrightarrow z)$	$[x \leftarrow \bot][z \leftarrow \top]$	$\exists\|x\|\forall\|z\|\exists\|y\|((x\rightarrow y)\leftrightarrow z)$	$[x \leftarrow \top][y \leftarrow z]$
$\exists\|y\|\exists\|z\|\forall\|x\|((x\rightarrow y)\leftrightarrow z)$	$[y \leftarrow \top][z \leftarrow \top]$	$\exists\|y\|\forall\|z\|\exists\|x\|((x\rightarrow y)\leftrightarrow z)$	$[y \leftarrow \bot][x \leftarrow \overline{z}]$
$\exists\|z\|\exists\|x\|\forall\|y\|((x\rightarrow y)\leftrightarrow z)$	$[z \leftarrow \top][x \leftarrow \bot]$	$\exists\|z\|\exists\|y\|\forall\|x\|((x\rightarrow y)\leftrightarrow z)$	$[z \leftarrow \top][y \leftarrow \top]$

TAB. 4.3 – Règles de propagation

$\exists\|z\|\forall\|x\|\exists\|y\|((x\rightarrow y)\leftrightarrow z)$	$[z \leftarrow \top]$
$\exists\|z\|\forall\|y\|\exists\|x\|((x\rightarrow y)\leftrightarrow z)$	$[z \leftarrow \top]$

TAB. 4.4 – Schémas contingents

$\exists\|x\|\exists\|y\|((x\rightarrow y)\leftrightarrow\top)$	$\forall\|x\|\exists\|y\|((x\rightarrow y)\leftrightarrow\top)$	$\exists\|y\|\exists\|x\|((x\rightarrow y)\leftrightarrow\top)$
$\forall\|y\|\exists\|x\|((x\rightarrow y)\leftrightarrow\top)$	$\exists\|x\|\exists\|y\|((x\rightarrow y)\leftrightarrow\overline{x})$	$\forall\|x\|\exists\|y\|((x\rightarrow y)\leftrightarrow\overline{x})$
$\exists\|y\|\exists\|x\|((x\rightarrow y)\leftrightarrow\overline{x})$	$\forall\|y\|\exists\|x\|((x\rightarrow y)\leftrightarrow\overline{x})$	$\exists\|x\|\exists\|y\|((x\rightarrow y)\leftrightarrow y)$
$\forall\|x\|\exists\|y\|((x\rightarrow y)\leftrightarrow y)$	$\exists\|y\|\exists\|x\|((x\rightarrow y)\leftrightarrow y)$	$\forall\|y\|\exists\|x\|((x\rightarrow y)\leftrightarrow y)$
$\exists\|x\|\exists\|y\|\exists\|z\|((x\rightarrow y)\leftrightarrow z)$	$\exists\|x\|\forall\|y\|\exists\|z\|((x\rightarrow y)\leftrightarrow z)$	$\forall\|x\|\exists\|y\|\exists\|z\|((x\rightarrow y)\leftrightarrow z)$
$\forall\|x\|\forall\|y\|\exists\|z\|((x\rightarrow y)\leftrightarrow z)$	$\exists\|x\|\exists\|z\|\exists\|y\|((x\rightarrow y)\leftrightarrow z)$	$\forall\|x\|\exists\|z\|\exists\|y\|((x\rightarrow y)\leftrightarrow z)$
$\exists\|y\|\exists\|x\|\exists\|z\|((x\rightarrow y)\leftrightarrow z)$	$\exists\|y\|\forall\|x\|\exists\|z\|((x\rightarrow y)\leftrightarrow z)$	$\forall\|y\|\exists\|x\|\exists\|z\|((x\rightarrow y)\leftrightarrow z)$
$\forall\|y\|\forall\|x\|\exists\|z\|((x\rightarrow y)\leftrightarrow z)$	$\exists\|y\|\exists\|z\|\exists\|x\|((x\rightarrow y)\leftrightarrow z)$	$\forall\|y\|\exists\|z\|\exists\|x\|((x\rightarrow y)\leftrightarrow z)$
$\exists\|z\|\exists\|x\|\exists\|y\|((x\rightarrow y)\leftrightarrow z)$	$\forall\|z\|\exists\|x\|\exists\|y\|((x\rightarrow y)\leftrightarrow z)$	$\exists\|z\|\exists\|y\|\exists\|x\|((x\rightarrow y)\leftrightarrow z)$
$\forall\|z\|\exists\|y\|\exists\|x\|((x\rightarrow y)\leftrightarrow z)$		

TAB. 4.5 – Schémas tautologiques

$((\bot\rightarrow\bot)\leftrightarrow\top)$	$((\bot\rightarrow\top)\leftrightarrow\top)$	$\exists\|y\|((\bot\rightarrow y)\leftrightarrow\top)$
$\forall\|y\|((\bot\rightarrow y)\leftrightarrow\top)$	$((\top\rightarrow\bot)\leftrightarrow\bot)$	$((\top\rightarrow\top)\leftrightarrow\top)$
$\exists\|y\|((\top\rightarrow y)\leftrightarrow y)$	$\forall\|y\|((\top\rightarrow y)\leftrightarrow y)$	$\exists\|x\|((x\rightarrow\bot)\leftrightarrow\overline{x})$
$\forall\|x\|((x\rightarrow\bot)\leftrightarrow\overline{x})$	$\exists\|x\|((x\rightarrow\top)\leftrightarrow\top)$	$\forall\|x\|((x\rightarrow\top)\leftrightarrow\top)$
$\exists\|x\|((x\rightarrow x)\leftrightarrow\top)$	$\forall\|x\|((x\rightarrow x)\leftrightarrow\top)$	$\exists\|x\|((x\rightarrow\overline{x})\leftrightarrow\overline{x})$
$\forall\|x\|((x\rightarrow\overline{x})\leftrightarrow\overline{x})$		

4.3 Propagation

Nous construisons l'algorithme 8, *propagation*, qui prend en entrée une QBF sous forme normale de contraintes, qui applique l'ensemble des règles de la section précedente jusqu'à obtention d'un point fixe, qui signifie qu'il n'y a plus que des instances de schémas contingents, et qui retourne une forme normale de contraintes. Si un point fixe n'est pas atteint, c'est qu'il demeure soit une instance d'un schéma contradictoire, dans ce cas la QBF est non valide, le calcul du point fixe est interrompu et *non_valide* est retourné, soit une instance d'un schéma tautologique, dans ce cas la contrainte est purement et simplement supprimée, soit une instance d'un schéma de propagation/simplification, dans ce cas la contrainte est supprimée et la substitution propagée aux autres contraintes, soit une instance d'un schéma de propagation, dans ce cas la substitution est propagée aux autres contraintes. Si le point fixe est atteint (sans contrainte quantifiée instance d'un schéma contradictoire), celui-ci est l'ensemble des contraintes restantes quantifiées par le lieur original restreint aux quantificateurs des symboles propositionnels présents dans la matrice.

Théorème 10 *Soit F une QBF sous forme normale de contraintes alors propagation(F) termine et propagation$(F) \equiv F$.*

La spécification d'un algorithme étendant l'algorithme 8 à l'équivalence qui préserve les modèles QBF est un peu plus technique à décrire puisque les arguments de correction des règles de propagation/simplification et de propagation ne sont pas *clos* pour la forme normale de contraintes (puisque rajoutant la formule associée à la substitution pour préserver les modèles QBF) tandis que sa restriction à l'équivalence qui préserve les modèles propositionnels l'est (la formule associée à la substitution étant supprimée).

Exemple 42 *En continuant l'exemple 38. La propagation tracée en figure 4.2 est pour la QBF sous forme normale de contraintes*

$$\forall a \exists d \forall e \exists f \exists z_3 \exists z_2 \exists z_1 ((e \rightarrow d) \leftrightarrow z_1) \wedge ((f \rightarrow a) \leftrightarrow z_2) \wedge ((z_1 \rightarrow \overline{z_2}) \leftrightarrow \bot)$$

À gauche de la ligne qui figure le temps sont inscrites les contraintes quantifiées ainsi que les substitutions tandis qu'à droite sont inscrits les schémas (et, le cas échéant, les règles) ainsi que leurs références dans les tables (données par le couple (colonne,ligne)). Donc, d'après le théorème 10

$$\xi \equiv \forall a \exists d \forall e \exists f \exists z_3 \exists z_2 \exists z_1 ((e \rightarrow d) \leftrightarrow z_1) \wedge ((f \rightarrow a) \leftrightarrow z_2) \wedge ((z_1 \rightarrow \overline{z_2}) \leftrightarrow \bot)$$

donc la QBF ξ est valide. Lors de la propagation, la substitution $[d \leftarrow \top]$ indique le calcul de la fonction booléenne $\hat{d} = \{(\mathbf{vrai} \mapsto \mathbf{vrai}), (\mathbf{faux} \mapsto \mathbf{vrai})\}$ comme faisant partie de tout modèle.

Algorithme 8 *propagation*

Entrée: Une QBF sous forme normale de contraintes $Q \bigwedge_{i \in I} K_i$
Sortie: Une QBF sous forme normale de contraintes

tant que Il existe une contrainte K_i, $i \in I$, telle que $Q|_{\mathcal{SP}(K_i)} K_i$ ne soit pas instance d'un schéma contingent **faire**
si $Q|_{\mathcal{SP}(K_i)} K_i$ est une instance d'un schéma contradictoire **alors**
retourner *non_valide*
fin si
si $Q|_{\mathcal{SP}(K_i)} K_i$ est une instance d'un schéma tautologique **alors**
$I := I \setminus \{i\}$
sinon si $Q|_{\mathcal{SP}(K_i)} K_i$ est instance d'un schéma associé à une substitution instanciée sur les symboles propositionnels de K_i en une substitution σ au sein d'une règle de propagation/simplification **alors**
$I := I \setminus \{i\}$
tant que $j \in I$ **faire**
$K_j := \sigma(K_j)$
fin tant que
sinon
$Q|_{\mathcal{SP}(K_i)} K_i$ est une instance d'un schema associé à une substitution instanciée sur les symboles propositionnels de K_i en une substitution σ au sein d'une règle de propagation
tant que $j \in I$ **faire**
$K_j := \sigma(K_j)$
fin tant que
fin si
fin tant que
retourner $Q|_{\mathcal{SP}(\cup_{i \in I} K_i)} \bigwedge_{i \in I} K_i$

4.4 Génération

Il est assez délicat d'obtenir les schémas non-redondants et les règles sans le faire algorithmiquement, nous avons donc, dans l'esprit de la figure 4.1, conçu des automates et programmes en *Prolog* pour les générer systématiquement et les compiler dans un langage de programmation cible. L'automate de la figure 4.3 génère les matrices possibles des schémas non-redondants auxquelles doivent être rajoutés les lieurs.

Pour un connecteur binaire donné, c'est sa sémantique qui permet de calculer si un schéma est contradictoire, tautologique, contingent ou s'il est associé soit à une règle de propagation/simplification soit à une règle de propagation. La base littérale (cf. 3) est ici un outil puissant pour faire apparaître si les contraintes quantifiées instances des schémas sont non-valides ou si les symboles propositionnels ont des valeurs de vérité forcées pour conserver la validité. Nous définissons une « contrainte quantifiée représentante d'un schéma » comme étant ce même schéma dont les littéraux ont été substitués par des symboles propositionnels et les complémentaires par des symboles propositionnels précédés

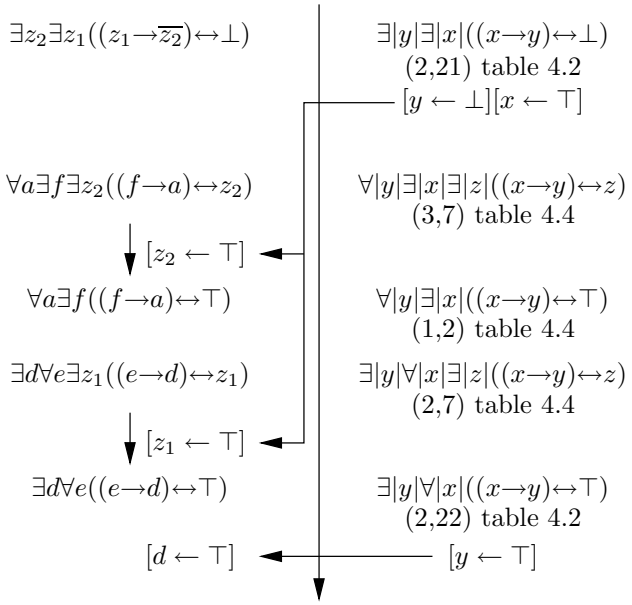


FIG. 4.2 – Propagation de l'exemple 42

$\Omega = \{\top, \bot, x, \overline{x}, y, \overline{y}, z\}$

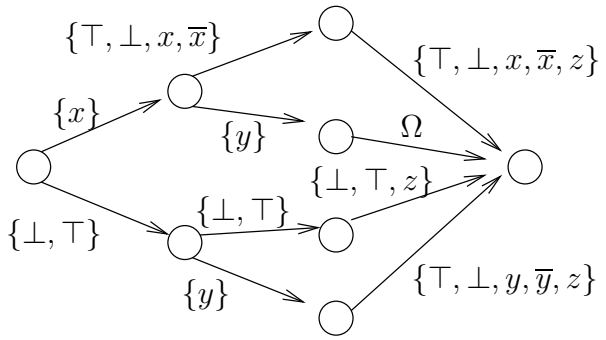


FIG. 4.3 – Automate de génération des schémas non-redondants

d'une négation. Les gardes forçant les symboles propositionnels existentiellement quantifiés d'une base littérale optimale et les substitutions des règles de propagation(/simplification) vérifient alors les équivalences suivantes (cf. Démonstrations, lemme 40) :

- le schéma est contradictoire si et seulement si la base littérale est *non_valide* ;
- le schéma $\mathcal{QE}$ est tautologique si et seulement si la base littérale optimale bl est équivalente à la base littérale $\langle Q \mid (\top, \top)^n \rangle$ avec n le nombre de quantificateurs du lieur ;
- le schéma induit une règle de propagation (/simplification) si et seulement si toutes les substitutions sont telles que :
  - $[x \leftarrow \top]$ si et seulement si $grd(x, bl) = (\top, \bot)$;
  - $[x \leftarrow \bot]$ si et seulement si $grd(x, bl) = (\bot, \top)$;
  - $[x \leftarrow \neg y]$ si et seulement si $grd(x, bl) = (\neg y, y)$;
  - $[x \leftarrow y]$ si et seulement si $grd(x, bl) = (y, \neg y)$.

($\mathcal{QE}$ un schéma, sa contrainte quantifiée représentante R et une base littérale optimale bl telle que $bl^* \cong R$).

Seuls les symboles propositionnels existentiellement quantifiés sont substitués et un littéral x est toujours substitué par un littéral y tel que $y \prec x$ dans le but de préserver la correction (quoique cela ne soit pas nécessaire dans le cas de deux littéraux dans la même classe d'équivalence induite par le lieur). Un schéma entraînant une substitution $[x \leftarrow y]$ avec $x \prec y$, x symbole propositionnel existentiellement quantifié et y symbole propositionnel universellement quantifié ne peut apparaître que dans un schéma contradictoire, ce qui correspond dans le cas de la *résolution* pour la *logique des prédicats* à échouer au *test d'occurrence* dans l'*unification*.

Exemple 43 *Le schéma* $\forall|z|\exists|x|((x \rightarrow \overline{x}) \leftrightarrow z)$ *entraîne une substitution* $[x \leftarrow \overline{z}]$ *tandis que le schéma* $\exists|x|\forall|z|((x \rightarrow \overline{x}) \leftrightarrow z)$ *est contradictoire.*

La table 4.6 (resp. table 4.7) met en relation les bases littérales des contraintes quantifiées associées et les schémas menant à des règles de propagation/simplification (resp. propagation) pour l'implication.

Exemple 44 *Schémas et bases littérales Au schéma* $\forall|x|\forall|y|((x \rightarrow y) \leftrightarrow \overline{x})$ *est associée la contrainte quantifiée représentante* $\forall x \forall y((x \rightarrow y) \leftrightarrow \neg x)$.

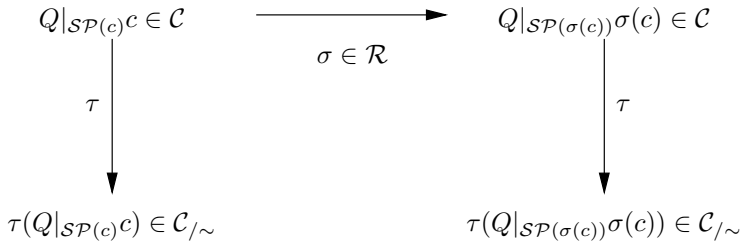
- *La contrainte quantifiée* $\exists x \exists y \forall z((x \rightarrow y) \leftrightarrow z)$ *est non-valide donc le schéma* $\exists|x|\exists|y|\forall|z|((x \rightarrow y) \leftrightarrow z)$ *est un schéma contradictoire ;*
- *la contrainte quantifiée* $\exists y((\bot \rightarrow y) \leftrightarrow \top)$ *a pour base littérale optimale réduite* $\langle \exists y \mid (\top, \top) \rangle$ *donc le schéma* $\exists|y|((\bot \rightarrow y) \leftrightarrow \top)$ *est un schéma tautologique ;*
- *la contrainte quantifiée* $\forall z \exists x \exists y((x \rightarrow y) \leftrightarrow z)$ *a pour base littérale optimale réduite* $\langle \forall z \exists x \exists y \mid (\top, \top); (\top, z); (z, (x \leftrightarrow \neg z)) \rangle$ *donc le schéma* $\forall|z|\exists|x|\exists|y|((x \rightarrow y) \leftrightarrow z)$ *est un schéma contingent ;*
- *la contrainte quantifiée* $\exists x \forall z \exists y((x \rightarrow y) \leftrightarrow z)$ *a pour base littérale optimale réduite* $\langle \exists x \forall z \exists y \mid (\top, \bot); (\top, \top); (z, \neg z) \rangle$ *donc au schéma* $\exists|x|\forall|z|\exists|y|((x \rightarrow y) \leftrightarrow z)$ *est associé la substitution* $[x \leftarrow \top][y \leftarrow z]$ *au sein d'une règle de propagation/simplification ;*

– *la contrainte quantifiée* $\exists z \forall x \exists y((x \to y) \leftrightarrow z)$ *a pour base littérale optimale réduite* $\langle \exists z \forall x \exists y \mid (\top, \bot); (\top, \top); (\top, \neg x) \rangle$ *donc au schéma* $\exists|z|\forall|x|\exists|y|((x \to y) \leftrightarrow z)$ *est associé la substitution* $[z \leftarrow \top]$ *au sein d'une règle de propagation.*

Nous avons implanté la chaîne entière de transformations qui va de la table de vérité pour un connecteur binaire jusqu'à un code pour un langage cible. Cette démarche est représentée dans la figure 4.1. C'est un programme Prolog basé sur un algorithme de calcul des bases littérales qui génère les schémas contradictoires, les schémas tautologiques et les règles de propagation(/simplification) pour l'implication, présentés ci-dessus, la conjonction, la disjonction, la bi-implication et le ou-exclusif, non présentés. Ce programme implante l'automate de la figure 4.3 qui calcule tous les schémas possibles ; de chacun en est dérivé immédiatement une contrainte quantifiée dont la base littérale optimale réduite est calculée ; de celle-ci est déduite s'il y a lieu une règle. Pour chaque connecteur binaire est donc généré à partir de sa table de vérité du code dans un langage cible.

4.5 Propagation dans l'espace des schémas

L'algorithme 8 nécessite, à chaque fois qu'une substitution est effectuée et pour les contraintes quantifiées dont au moins un symbole propositionnel a été modifié, de recalculer les schémas associés à celles-ci selon le diagramme ci-dessous.



Si l'algorithme du calcul d'un schéma (une série de tests) est élémentaire, étant reproduit un très grand nombre de fois, il en devient très coûteux. Mais connaissant la substitution, le calcul du nouveau schéma pour la contrainte quantifiée ayant subie celle-ci peut être décomposé en deux phases dont l'une consiste à calculer et l'autre à appliquer une abstraction de la substitution, une *substitution sur schéma*, dans l'espace quotient de la relation d'équivalence $\sim$. Si le calcul de la substitution sur schéma reste dynamique, l'application de celle-ci peut être construite une bonne fois pour toutes et compilée à part car elle représente le comportement des schémas par rapport aux substitutions de schéma qui est fixe. La définition de la substitution sur schéma découle des trois possibilités de substitutions pour un littéral existentiellement quantifié qui sont soit par une constante propositionnelle, soit par un littéral déjà présent dans la contrainte, soit par un littéral non encore présent dans la contrainte.

TAB. 4.6 – Bases littérales optimales pour les règles de propagation/simplification

$\exists\|z\|((\bot\rightarrow\bot)\leftrightarrow z)$	$(\top,\bot)$	$\exists\|z\|((\bot\rightarrow\top)\leftrightarrow z)$	$(\top,\bot)$
$\exists\|y\|((\bot\rightarrow y)\leftrightarrow y)$	$(\top,\bot)$	$\exists\|y\|((\bot\rightarrow y)\leftrightarrow\overline{y})$	$(\bot,\top)$
$\exists\|y\|\exists\|z\|((\bot\rightarrow y)\leftrightarrow z)$	$(\top,\top);(\top,\bot)$	$\forall\|y\|\exists\|z\|((\bot\rightarrow y)\leftrightarrow z)$	$(\top,\top);(\top,\bot)$
$\exists\|z\|\exists\|y\|((\bot\rightarrow y)\leftrightarrow z)$	$(\top,\bot);(\top,\top)$	$\exists\|z\|\forall\|y\|((\bot\rightarrow y)\leftrightarrow z)$	$(\top,\bot);(\top,\top)$
$\exists\|z\|((\top\rightarrow\bot)\leftrightarrow z)$	$(\bot,\top)$	$\exists\|z\|((\top\rightarrow\top)\leftrightarrow z)$	$(\top,\bot)$
$\exists\|y\|((\top\rightarrow y)\leftrightarrow\bot)$	$(\bot,\top)$	$\exists\|y\|((\top\rightarrow y)\leftrightarrow\top)$	$(\top,\bot)$
$\exists\|y\|\exists\|z\|((\top\rightarrow y)\leftrightarrow z)$	$(\top,\top);(y,\neg y)$	$\forall\|y\|\exists\|z\|((\top\rightarrow y)\leftrightarrow z)$	$(\top,\top);(y,\neg y)$
$\exists\|z\|\exists\|y\|((\top\rightarrow y)\leftrightarrow z)$	$(\top,\top);(z,\neg z)$	$\forall\|z\|\exists\|y\|((\top\rightarrow y)\leftrightarrow z)$	$(\top,\top);(z,\neg z)$
$\exists\|x\|((x\rightarrow\bot)\leftrightarrow\bot)$	$(\top,\bot)$	$\exists\|x\|((x\rightarrow\bot)\leftrightarrow\top)$	$(\bot,\top)$
$\exists\|x\|\exists\|z\|((x\rightarrow\bot)\leftrightarrow z)$	$(\top,\top);(\neg x,x)$	$\forall\|x\|\exists\|z\|((x\rightarrow\bot)\leftrightarrow z)$	$(\top,\top);(\neg x,x)$
$\exists\|z\|\exists\|x\|((x\rightarrow\bot)\leftrightarrow z)$	$(\top,\top);(\neg z,z)$	$\forall\|z\|\exists\|x\|((x\rightarrow\bot)\leftrightarrow z)$	$(\top,\top);(\neg z,z)$
$\exists\|x\|((x\rightarrow\top)\leftrightarrow x)$	$(\top,\bot)$	$\exists\|x\|((x\rightarrow\top)\leftrightarrow\overline{x})$	$(\bot,\top)$
$\exists\|x\|\exists\|z\|((x\rightarrow\top)\leftrightarrow z)$	$(\top,\top);(\top,\bot)$	$\forall\|x\|\exists\|z\|((x\rightarrow\top)\leftrightarrow z)$	$(\top,\top);(\top,\bot)$
$\exists\|z\|\exists\|x\|((x\rightarrow\top)\leftrightarrow z)$	$(\top,\bot);(\top,\top)$	$\exists\|z\|\forall\|x\|((x\rightarrow\top)\leftrightarrow z)$	$(\top,\bot);(\top,\top)$
$\exists\|x\|((x\rightarrow x)\leftrightarrow x)$	$(\top,\bot)$	$\exists\|x\|((x\rightarrow x)\leftrightarrow\overline{x})$	$(\bot,\top)$
$\exists\|x\|\exists\|z\|((x\rightarrow x)\leftrightarrow z)$	$(\top,\top);(\top,\bot)$	$\forall\|x\|\exists\|z\|((x\rightarrow x)\leftrightarrow z)$	$(\top,\top);(\top,\bot)$
$\exists\|z\|\exists\|x\|((x\rightarrow x)\leftrightarrow z)$	$(\top,\bot);(\top,\top)$	$\exists\|z\|\forall\|x\|((x\rightarrow x)\leftrightarrow z)$	$(\top,\bot);(\top,\top)$
$\exists\|x\|((x\rightarrow\overline{x})\leftrightarrow\bot)$	$(\top,\bot)$	$\exists\|x\|((x\rightarrow\overline{x})\leftrightarrow\top)$	$(\bot,\top)$
$\exists\|x\|\exists\|z\|((x\rightarrow\overline{x})\leftrightarrow z)$	$(\top,\top);(\neg x,x)$	$\forall\|x\|\exists\|z\|((x\rightarrow\overline{x})\leftrightarrow z)$	$(\top,\top);(\neg x,x)$
$\exists\|z\|\exists\|x\|((x\rightarrow\overline{x})\leftrightarrow z)$	$(\top,\top);(\neg z,z)$	$\forall\|z\|\exists\|x\|((x\rightarrow\overline{x})\leftrightarrow z)$	$(\top,\top);(\neg z,z)$
$\exists\|x\|\exists\|y\|((x\rightarrow y)\leftrightarrow\bot)$	$(\top,\bot);(\bot,\top)$	$\exists\|y\|\exists\|x\|((x\rightarrow y)\leftrightarrow\bot)$	$(\bot,\top);(\top,\bot)$
$\exists\|x\|\forall\|y\|((x\rightarrow y)\leftrightarrow\top)$	$(\bot,\top);(\top,\top)$	$\exists\|y\|\forall\|x\|((x\rightarrow y)\leftrightarrow\top)$	$(\top,\bot);(\top,\top)$
$\exists\|x\|\exists\|y\|((x\rightarrow y)\leftrightarrow x)$	$(\top,\bot);(\top,\bot)$	$\exists\|y\|\exists\|x\|((x\rightarrow y)\leftrightarrow x)$	$(\top,\bot);(\top,\bot)$
$\exists\|x\|\forall\|y\|((x\rightarrow y)\leftrightarrow\overline{x})$	$(\bot,\top);(\top,\top)$	$\exists\|y\|\forall\|x\|((x\rightarrow y)\leftrightarrow\overline{x})$	$(\bot,\top);(\top,\top)$
$\exists\|x\|\forall\|y\|((x\rightarrow y)\leftrightarrow y)$	$(\top,\bot);(\top,\top)$	$\exists\|y\|\forall\|x\|((x\rightarrow y)\leftrightarrow y)$	$(\top,\bot);(\top,\top)$
$\exists\|x\|\exists\|y\|((x\rightarrow y)\leftrightarrow\overline{y})$	$(\bot,\top);(\bot,\top)$	$\exists\|y\|\exists\|x\|((x\rightarrow y)\leftrightarrow\overline{y})$	$(\bot,\top);(\bot,\top)$
$\exists\|x\|\exists\|z\|\forall\|y\|((x\rightarrow y)\leftrightarrow z)$	$(\bot,\top);(\top,\bot);(\top,\top)$	$\exists\|x\|\forall\|z\|\exists\|y\|((x\rightarrow y)\leftrightarrow z)$	$(\top,\bot);(\top,\top);(z,\neg z)$
$\exists\|y\|\exists\|z\|\forall\|x\|((x\rightarrow y)\leftrightarrow z)$	$(\top,\bot);(\top,\bot);(\top,\top)$	$\exists\|y\|\forall\|z\|\exists\|x\|((x\rightarrow y)\leftrightarrow z)$	$(\bot,\top);(\top,\top);(\neg z,z)$
$\exists\|z\|\exists\|x\|\forall\|y\|((x\rightarrow y)\leftrightarrow z)$	$(\top,\bot);(\bot,\top);(\top,\top)$	$\exists\|z\|\exists\|y\|\forall\|x\|((x\rightarrow y)\leftrightarrow z)$	$(\top,\bot);(\top,\bot);(\top,\top)$

TAB. 4.7 – Bases littérales optimales pour les règles de propagation

$\exists\|z\|\forall\|x\|\exists\|y\|((x\rightarrow y)\leftrightarrow z)$	$(\top,\bot);(\top,\top);(\top,\neg x)$
$\exists\|z\|\forall\|y\|\exists\|x\|((x\rightarrow y)\leftrightarrow z)$	$(\top,\bot);(\top,\top);(y,\top)$

Définition 45 (substitution sur schéma) *L'ensemble des substitutions dans l'ensemble des schémas $\mathcal{C}_{/\sim}$ est*

$$\begin{array}{l}\mathcal{R}_{\mathcal{C}_{/\sim}} = \\ \quad \{[o \leftarrow \top], [o \leftarrow \bot] \mid o \in \{x,y,z\}\} \cup \\ \quad \{[o \leftarrow \overline{o'}], [o \leftarrow o'] \mid o \in \{x,y,z\}, o' \in \{x,y,z\} \setminus \{o\}\} \cup \\ \quad \{[o \leftarrow l] \mid \ o \in \{x,y,z\}, \\ \qquad\qquad l \in \{\exists <, < \exists, \exists <<, < \exists <, << \exists, \forall <, < \forall, \forall <<, < \forall <, << \forall\}\}\end{array}$$

Dans la définition précédente, le symbole « $<$ » marque la position dans le lieur d'un littéral qui n'a pas été substitué ; le symbole $\exists$ signifie que le littéral existentiellement quantifié a été substitué par un autre littéral existentiellement quantifié ; le symbole $\forall$ signifie que le littéral existentiellement quantifié a été substitué par un littéral universellement quantifié.

Les substitutions sur schéma $[o \leftarrow \exists <]$, $[o \leftarrow< \exists]$, $[o \leftarrow \forall <]$ et $[o \leftarrow< \forall]$ ne peuvent être appliquées que sur des schémas contenant deux quantificateurs ; les autres substitutions sur schéma $[o \leftarrow \exists <<]$, $[o \leftarrow< \exists <]$, $[o \leftarrow<< \exists]$, $[o \leftarrow \forall <<]$, $[o \leftarrow< \forall <]$ et $[o \leftarrow<< \forall]$ ne peuvent être appliquées que sur des schémas contenant trois quantificateurs.

Exemple 45 *Les vingt-quatre substitutions sur schéma possibles ($\circ$ un connecteur binaire) sur le schéma $\exists|x|\exists|y|\exists|z|((x \circ y) \leftrightarrow z)$.*

$\tilde{\sigma} \in \mathcal{R}_{\mathcal{C}_{/\sim}}$	$\tilde{\sigma}(\exists\|x\|\exists\|y\|\exists\|z\|((x\circ y)\leftrightarrow z))$	$\tilde{\sigma} \in \mathcal{R}_{\mathcal{C}_{/\sim}}$	$\tilde{\sigma}(\exists\|x\|\exists\|y\|\exists\|z\|((x\circ y)\leftrightarrow z))$
$[x \leftarrow \top]$	$\exists\|y\|\exists\|z\|((\top\circ y)\leftrightarrow z)$	$[x \leftarrow \bot]$	$\exists\|y\|\exists\|z\|((\bot\circ y)\leftrightarrow z)$
$[x \leftarrow \exists <<]$	$\exists\|x\|\exists\|y\|\exists\|z\|((x\circ y)\leftrightarrow z)$	$[x \leftarrow \forall <<]$	$\forall\|x\|\exists\|y\|\exists\|z\|((x\circ y)\leftrightarrow z)$
$[y \leftarrow \top]$	$\exists\|x\|\exists\|z\|((x\circ \top)\leftrightarrow z)$	$[y \leftarrow \bot]$	$\exists\|x\|\exists\|z\|((x\circ \bot)\leftrightarrow z)$
$[y \leftarrow \exists <<]$	$\exists\|y\|\exists\|x\|\exists\|z\|((x\circ y)\leftrightarrow z)$	$[y \leftarrow \forall <<]$	$\forall\|y\|\exists\|x\|\exists\|z\|((x\circ y)\leftrightarrow z)$
$[y \leftarrow< \exists <]$	$\exists\|x\|\exists\|y\|\exists\|z\|((x\circ y)\leftrightarrow z)$	$[y \leftarrow< \forall <]$	$\exists\|x\|\forall\|y\|\exists\|z\|((x\circ y)\leftrightarrow z)$
$[y \leftarrow x]$	$\exists\|x\|\exists\|z\|((x\circ x)\leftrightarrow z)$	$[y \leftarrow \overline{x}]$	$\exists\|x\|\exists\|z\|((x\circ \overline{x})\leftrightarrow z)$
$[z \leftarrow \top]$	$\exists\|x\|\exists\|y\|((x\circ y)\leftrightarrow \top)$	$[z \leftarrow \bot]$	$\exists\|x\|\exists\|y\|((x\circ y)\leftrightarrow \bot)$
$[z \leftarrow \exists <<]$	$\exists\|z\|\exists\|x\|\exists\|y\|((x\circ y)\leftrightarrow z)$	$[z \leftarrow \forall <<]$	$\forall\|z\|\exists\|x\|\exists\|y\|((x\circ y)\leftrightarrow z)$
$[z \leftarrow< \exists <]$	$\exists\|x\|\exists\|z\|\exists\|y\|((x\circ y)\leftrightarrow z)$	$[z \leftarrow< \forall <]$	$\exists\|x\|\forall\|z\|\exists\|y\|((x\circ y)\leftrightarrow z)$
$[z \leftarrow<< \exists]$	$\exists\|x\|\exists\|y\|\exists\|z\|((x\circ y)\leftrightarrow z)$	$[z \leftarrow<< \forall]$	$\exists\|x\|\exists\|y\|\forall\|z\|((x\circ y)\leftrightarrow z)$
$[z \leftarrow x]$	$\exists\|x\|\exists\|y\|((x\circ y)\leftrightarrow x)$	$[z \leftarrow \overline{x}]$	$\exists\|x\|\exists\|y\|((x\circ y)\leftrightarrow \overline{x})$
$[z \leftarrow y]$	$\exists\|x\|\exists\|y\|((x\circ y)\leftrightarrow y)$	$[z \leftarrow \overline{y}]$	$\exists\|x\|\exists\|y\|((x\circ y)\leftrightarrow \overline{y})$

Nous définissons sans l'expliciter algorithmiquement (car elle est fastidieuse mais élémentaire) la fonction π qui permet à partir d'une substitution, selon le contexte de la QBF, c'est-à-dire la contrainte et le lieur, de calculer la substitution sur schéma correspondante.

Définition 46 (fonction de calcul de la substitution sur schéma) *La fonction de calcul de la substitution sur schéma π calcule à partir d'une contrainte c et d'un lieur Q une fonction $\pi(c,Q)$ qui à partir d'une substitution σ calcule la substitution sur schéma $\tilde{\sigma} = \pi(c,Q)(\sigma)$ telle que $\tilde{\sigma}(\tau(Q|_{\mathcal{SP}(c)}c)) = \tau(Q|_{\mathcal{SP}(\sigma(c))}\sigma(c))$.*

Exemple 46 (substitution sur schéma) *Soit* $c = ((a{\rightarrow}d){\leftrightarrow}e)$ *une contrainte d'une QBF sous forme normale de contraintes de lieur* $Q = \ldots\forall l\ldots\exists a\ldots\forall d\ldots\exists e\ldots$ *alors* $\tau(\exists a\forall d\exists ec) = \exists|x|\forall|y|\exists|z|((x{\rightarrow}y){\leftrightarrow}z)$ *; prenons la substitution* $\sigma = [e \leftarrow l]$ *alors la substitution sur schéma correspondante est*

$$\pi(c,Q)([e \leftarrow l])(\exists|x|\forall|y|\exists|z|((x{\rightarrow}y){\leftrightarrow}z)) = [z \leftarrow \forall <<]$$

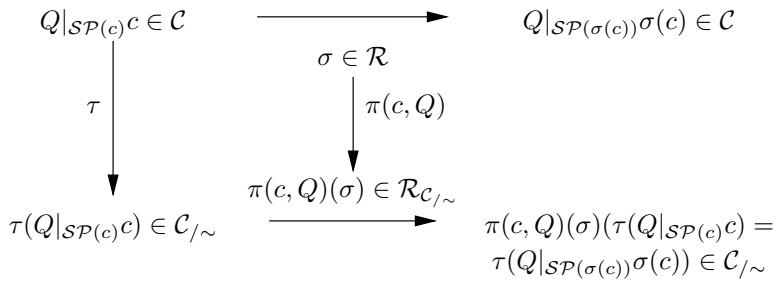
puisque le résultat de l'application de la substitution σ *sur la contrainte* c *est* $\sigma(c) = ((a{\rightarrow}d){\leftrightarrow}l)$ *et que la contrainte quantifiée* $\forall l\exists a\forall d\sigma(c)$ *est de schéma* $\forall|z|\exists|x|\forall|y|((x{\rightarrow}y){\leftrightarrow}z)$ *(le « ∀ » de « ∀ << » correspond au « ∀z », le premier « < » correspondant à « ∃x » et le second à « ∀y »).*

Maintenant, modifions le lieur en $Q' = \ldots\exists a\ldots\forall l\ldots\forall d\ldots\exists e\ldots$ *ce qui ne change pas* $\tau(\exists a\forall d\exists ec)$ *(mais* $a \prec l$ *au lieu de* $l \prec a$*) ; conservons aussi la substitution* $\sigma = [e \leftarrow l]$ *alors la substitution sur schéma correspondant à la substitution* σ *est maintenant*

$$\pi(c,Q')([e \leftarrow l])(\exists|x|\forall|y|\exists|z|((x{\rightarrow}y){\leftrightarrow}z)) = [z \leftarrow< \forall <]$$

puisque la contrainte quantifiée $\exists a\forall l\forall d\sigma(c)$ *est de schéma* $\exists|x|\forall|z|\forall|y|((x{\rightarrow}y){\leftrightarrow}z)$*.*

Le diagramme précédent, grâce à la fonction de calcul de la substitution sur schéma devient :



Comme pour la génération des schémas, celle de la fonction de calcul de la substitution sur schéma est automatique grâce à un programme Prolog et cette fonction est compilée en un langage cible.

Exemple 47 *En continuant l'exemple 42. La propagation tracée en figure 4.4 est pour la QBF sous forme normale de contraintes*

$$Q((e{\rightarrow}d){\leftrightarrow}z_1){\wedge}((f{\rightarrow}a){\leftrightarrow}z_2){\wedge}((z_1{\rightarrow}\overline{z_2}){\leftrightarrow}\bot)$$

avec $Q = \forall a\exists d\forall e\exists f\exists z_3\exists z_2\exists z_1$*. À gauche de la ligne qui figure le temps sont inscrites les contraintes quantifiées ainsi que les substitutions tandis que à droite sont inscrits les schémas (et, le cas échéant, les règles) ainsi que leurs références dans les tables (données par le couple (colonne,ligne)).*

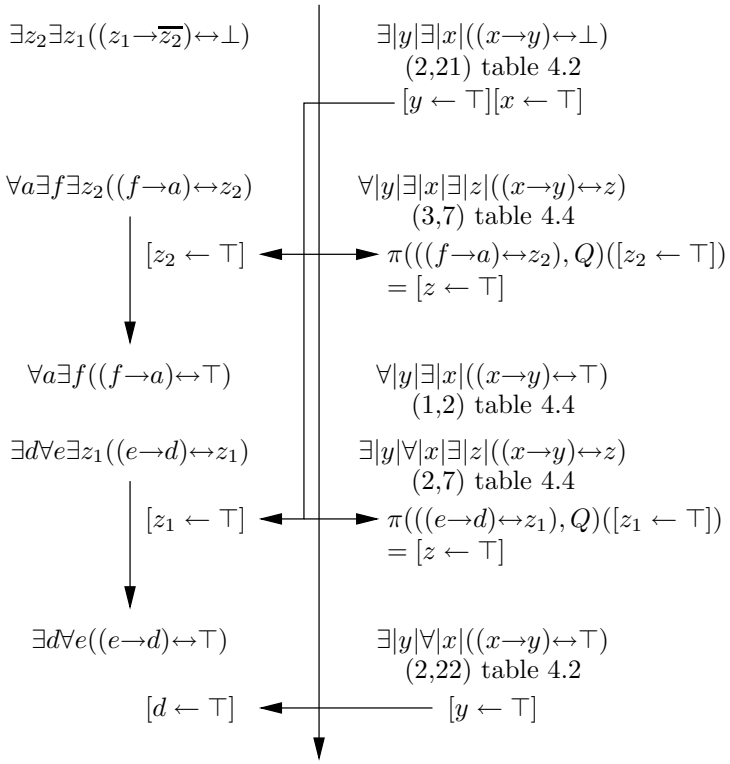


FIG. 4.4 – Propagation de l'exemple 47

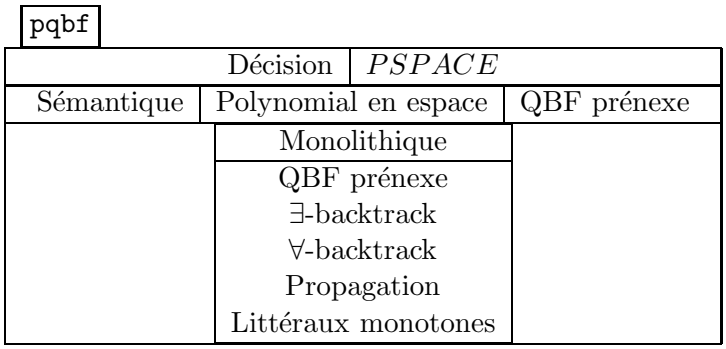
4.6 Algorithme complet

L'algorithme 8 de propagation n'est pas suffisant pour décider si une QBF est valide ou non, pour s'en convaincre il suffit d'écrire une QBF sous forme normale de contraintes ne contenant que des instances de schémas contingents (elle ne sera pas nécessairement pour autant non-valide) ; l'algorithme de propagation doit donc être inséré dans un algorithme qui lui apporte la *complétude*. Nous avons exploré deux voies.

La première est la compilation directe de l'ensemble des schémas et des règles, sans utiliser la fonction de calcul des substitutions sur schéma, dans un formalisme déclaratif qui fournit la gestion de la propagation et les mécanismes de parcours de l'espace de recherche tel que le formalisme CHR. L'expérience a été menée avec l'implantation CHR au sein de l'interprète/compilateur Prolog de Sicstus. Des prédicats Prolog auxiliaires manipulent les quantificateurs et les littéraux : un prédicat réalise l'ordre $\prec$ sur le lieur ; des prédicats testent si le littéral est existentiellement ou universellement quantifié ; un prédicat teste si deux littéraux sont conjugués. Une règle CHR est constituée d'une « tête » qui associe le nom de la contrainte et ses arguments, d'un « cou » qui identifie soit une règle de propagation/simplification ou soit une règle de propagation et d'un corps constitué d'une garde et d'actions. La traduction des schémas et règles est directe et intuitive en CHR (voir l'article sur CHR pour plus de détails). La principale difficulté est de modifier CHR pour prendre en compte les littéraux. Cette implantation a pour principal intérêt de valider l'approche mais le temps d'exécution est exécrable sans doute par le calcul incessant des gardes. Le code CHR généré pour les cinq connecteurs binaires classiques peut être obtenu à l'adresse : `http://www.info.univ-angers.fr/pub/stephan/Research/QBF/PROPAGATION/propagation.html`.

La seconde voie explorée est l'implantation de l'algorithme 2, *recherche_prenexe*, et l'insertion, avant l'appel récursif pour éliminer un quantificateur, de l'algorithme de propagation. Le langage de programmation cible est le C++, choisi pour ses performances et nos connaissances. Le résultat est la procédure de décision `pqbf` qui implante la version utilisant la fonction de calcul des substitutions sur schéma. Cette implantation a participé à la compétition QBF'08 dans la catégorie non-prénexe non-FNC[1]. Elle a terminé première sur les tests non aléatoires et deuxième sur les tests aléatoires sur trois compétiteurs (!) bien que les tests lui soient particulièrement défavorables puisque ayant subi la linéarisation et ne contenant alors ni bi-implication ni ou-exclusif (rendant notre approche peu pertinente).

[1]Notre implantation n'est pas non-prénexe mais les tests de référence étant uniquement sur le fragment FNN, la mise sous forme prénexe est réalisable « à la volée ».



4.7 Commentaires

Le choix du vocabulaire de ce chapitre n'est pas dû au hasard : il est directement inspiré des *problèmes de satisfaction de contraintes*, et en particulier, de l'approche basée sur les règles [7] dont le langage CHR [85] permet l'implantation. Notre système peut-être vue comme une extension, même au niveau propositionnel, des classiques ensembles de règles booléennes [6] mais en plus puissant dans sa capacité à propager. La raison en est la même que pour l'*algorithme de Stålmarck* : l'utilisation des littéraux et la construction des classes d'équivalence. De même, notre système peut-être vu comme une extension des règles de propagation proposées dans [40] mais, une fois encore, en plus puissant puisque l'utilisation des littéraux et la construction des classes d'équivalence dépasse la propriété de *consistance d'arc quantifiée*.

Le système `pqbf` qui implante la procédure de décision par propagation utilise un mécanisme de point fixe similaire à celui de l'*algorithme AC3* présent dans le domaine de la résolution de problèmes de satisfaction de contraintes ; nous avons peu tiré parti de la proximité avec ce domaine. Nous y remédions, en collaboration avec Vincent Barichard, en procédant à la refonte de la procédure `pqbf`, au sein de l'architecture ouverte de propagation de contraintes GeCode [198]. Ce choix a été fait pour deux raisons : pouvoir distribuer un code que la communauté peut s'approprier, modifier et dont elle peut vérifier les performances ; analyser les différentes composantes de l'approche pour en mieux comprendre ce qui est réellement efficace et ce qui peut alors s'étendre soit aux procédures de décision pour le problème *SAT* mais non-FNC [138, 116, 139, 222] soit à la résolution de *problèmes de satisfaction de contraintes quantifiées*. Ce chapitre a été consacré à la propagation booléenne pour des QBF *prénexes* non-FNC mais par dualité nous avons refondu le système `pqbf` en une procédure de décision pour les QBF *non-prénexes* non-FNC.

Chapitre 5

Des QBF vers les ASP

Il y eut un soir, il y eut un matin : cinquième jour.

[60] B. Da Mota, I. Stéphan, and P. Nicolas. From (Quantified) Boolean Formulas to Answer Set Programming. In *Proceedings of the 7th International Answer Set Programming Workshop (ASP'07)*, 2007.

[61] B. Da Mota, I. Stéphan, and P. Nicolas. Des formules booléennes (quantifiées) à la programmation par ensembles réponses. In *Reconnaissance des Formes et Intelligence Artificielle*, 2008.

[215] I. Stéphan, B. Da Mota, and P. Nicolas. From (Quantified) Boolean Formulas to Answer Set Programming. *Journal of logic and computation*, 19(4) :565–590, 2009.

Il peut paraître assez étrange de vouloir compiler une QBF qui est susceptible de représenter un problème *complet* pour la classe de complexité PSPACE en un *programme logique normal* qui ne pourra au mieux représenter qu'un problème complet pour la classe de complexité NP, à moins d'obtenir un programme de taille exponentielle par rapport à la QBF (ce qui peut apparaître *a priori* sans intérêt mais qui ne l'est pas) ou de réussir à démontrer que la *hiérarchie polynomiale* s'effondre sur elle-même (ce qui apparaît peu probable). C'est pourtant bien cette optique que nous avons choisi de suivre. Notre étude est la conséquence d'un travail préparatoire qui part du double constat que, d'une part, la plus grande partie des travaux de résolution complète pour SAT s'appuie sur le fragment en forme normale négative, voire en forme normale conjonctive et que peu de travaux s'intéressent au *langage de la logique propositionnelle* dans son entier [138, 116, 139, 222] et que, d'autre part, des expériences de traduction pour le fragment FNC de SAT [167] et des cas particuliers [113] vers les *programmes logiques normaux* ont déjà donné lieu à des travaux. De part la relation entre le problème de validité des QBF et la hiérarchie polynomiale, il est naturel de considérer le langage des QBF comme un langage cible admissible pour la compilation des langages logiques non monotones. C'est le parti qui a été choisi dans le système `QUIP`. Au delà d'un but purement théorique de donner une traduction polynomiale pour toute *formule propositionnelle* vers un programme logique normal, l'intérêt est double : offrir un outil permettant de calculer les *modèles* de la formule propositionnelle en utilisant la puissance des outils développer pour la programmation logique normale et proposer un nouvel ensemble de tests pour comparer les différentes implémentations entre-elles. Notre étude se place dans le cadre de la *programmation par ensembles réponses* (ASP) selon la *sémantique des modèles stables*. Nous nous appesantissons sur la traduction de SAT vers ASP car la traduction de QBF vers ASP l'étend.

Le langage choisi pour la traduction n'autorise pas l'utilisation de la négation logique mais « seulement » d'une *négation par défaut* dans le langage cible. La technique classique est d'introduire pour chaque symbole propositionnel, deux *atomes* : l'un pour représenter le *littéral positif* et l'autre le littéral négatif ; deux *règles*, par exemple pour le symbole propositionnel x, $(x \leftarrow not\ \overline{x}.)$ et $(\overline{x} \leftarrow not\ x.)$ sont ajoutées à la traduction pour assurer le postulat de bivalence qui exprime qu'un symbole propositionnel est interprété soit à **vrai**, soit à **faux** mais pas aux deux. Le reste de la traduction est par induction sur la structure de la formule propositionnelle : un symbole propositionnel est traduit en un atome de même symbole ; pour chaque instance de connecteur à l'occurrence o de la formule sont associés un nouvel atome s_o *intermédiaire* capturant le résultat de la formule ainsi qu'une règle pour la négation et la conjonction et deux règles pour les autres connecteurs binaires (en supposant que les nouveaux atomes s', s_g et s_d soient associés aux sous-formules α', α_g et α_d) : à l'instance $\alpha_o = \neg\alpha'$ à l'occurrence o est associée la règle $(s_o \leftarrow not\ s'.)$; à l'instance $\alpha_o = (\alpha_g \wedge \alpha_d)$ à l'occurrence o est associée la règle $(s_o \leftarrow s_g, s_d.)$; à l'instance $\alpha_o = (\alpha_g \vee \alpha_d)$ à l'occurrence o sont associées les règles $(s_o \leftarrow s_g.)$ et $(s_o \leftarrow s_d.)$; à l'instance $\alpha_o = (\alpha_g \rightarrow \alpha_d)$ à l'occurrence o sont associées les règles $(s_o \leftarrow not\ s_g.)$ et $(s_o \leftarrow s_d.)$; à l'instance $\alpha_o = (\alpha_g \leftrightarrow \alpha_d)$ à l'occurrence o sont associées les règles $(s_o \leftarrow s_g, s_d.)$ et $(s_o \leftarrow not\ s_g, not\ s_d.)$; à l'instance $\alpha_o = (\alpha_g \oplus \alpha_d)$ à l'occurrence o sont associées les règles $(s_o \leftarrow s_g, not\ s_d.)$ et $(s_o \leftarrow not\ s_g, s_d.)$. Les règles introduites pour la négation, la conjonction et la disjonction sont obtenues de

la *sémantique des connecteurs* tandis que les règles introduites pour l'implication, la bi-implication et le ou-exclusif sont obtenues des *équivalences logiques* (prenant ici valeur de définition) de ces connecteurs par rapport aux trois précédemment cités respectivement par $(F \rightarrow G) \equiv (\neg F \vee G)$, $(F \leftrightarrow G) \equiv ((F \wedge G) \vee (\neg F \wedge \neg G))$ et $(F \oplus G) \equiv ((F \wedge \neg G) \vee (\neg F \wedge G))$. Les atomes intermédiaires introduits par la traduction sont similaires d'un point de vue logique aux symboles propositionnels introduits dans une formule propositionnelle pour en capturer les sous formules [223, 74, 177] comme dans la mise sous forme normale conjonctive par « renommage de formules » (cf. § 1.3.3) et la décomposition par introduction de littéraux au chapitre 4 mais ils n'en ont pas le statut dans le programme logique : la version niée n'apparaît pas. De part ce fait, un des deux atomes issus d'un symbole propositionnel de la formule apparaîtra donc nécessairement dans un *modèle stable* signifiant que l'interprétation du littéral est égale à **vrai**, tandis qu'un atome intermédiaire apparaîtra dans le modèle stable si la sous formule qui lui est associée est interprétée à **vrai** tandis qu'il en sera absent si elle est interprétée à **faux**. Une *contrainte* ($\leftarrow$ *not* s_ϵ) est rajoutée au programme pour l'atome associé à la formule entière (rendant sa présence obligatoire dans le modèle stable) pour exprimer que la formule doit être *satisfiable*. La *correction* de la méthode garantit que de tout modèle stable du programme généré peut être extrait un modèle propositionnel pour la formule en oubliant les atomes intermédiaires et la *complétude* garantit que tout modèle propositionnel de la formule peut être étendu en un (unique) modèle stable pour le programme.

L'extension de la traduction au langage des QBF prénexes est réalisée grâce à la *skolémisation* : chaque symbole propositionnel existentiellement quantifié est éliminé au profit d'un *symbole de fonction* ; la restriction de l'interprétation aux *booléens* traduit donc tout symbole propositionnel existentiellement quantifié en une fonction des booléens dans les booléens, en un prédicat. C'est donc un programme logique normal au premier ordre qui devient le langage cible. Un tel programme n'est qu'une version compacte des instanciations possibles des règles sur l'ensemble des *termes* possibles construits sur les symboles de fonction et *symboles de constante* du programme. Dans le cadre de la restriction à l'interprétation aux booléens, l'ensemble des termes possibles se réduit donc à $\{1, 0\}$, le symbole 1 (resp. 0) étant interprété à **vrai** (resp. **faux**). Chaque symbole propositionnel existentiellement quantifié est traduit en un atome dont les arguments sont des *variables logiques* représentant les symboles propositionnels universellement quantifiés (le cas propositionnel n'en est donc qu'un cas dégénéré avec des prédicats sans argument). Les règles obtenues pour la traduction au niveau propositionnel sont simplement « élevées » au premier ordre par l'ajout des arguments. Tous les symboles intermédiaires introduits lors de la traduction propositionnelle sont donc aussi « élevés ». La traduction du quantificateur universel est obtenue directement par sa sémantique exprimée sous la forme d'une conjonction : tout symbole propositionnel universellement quantifié x précédé de n symboles propositionnels universellement quantifiés est traduite en une règle

$$s_n(U_1, \ldots, U_n) \leftarrow s_{n+1}(U_1, \ldots, U_n, 1), s_{n+1}(U_1, \ldots, U_n, 0).$$

et toute instance d'un symbole propositionnel universellement quantifié à l'occurrence o, par exemple x précédé de n symboles propositionnels universellement quantifiées, introduit une règle ($s_o(U_1, \ldots, U_p) \leftarrow U_{n+1} = 1.$), p étant le nombre total de symboles

propositionnels universellement quantifiés. Comme dans le cas propositionnel, pour un symbole propositionnel x précédé de n symboles propositionnels universellement quantifiés et pour toute combinaison $(u_1, \ldots, u_n)$ de $\{1,0\}^n$, un des deux atomes $x(u_1, \ldots, u_n)$ (resp. $nx(u_1, \ldots, u_n)$) apparaîtra donc nécessairement dans un modèle stable signifiant que la fonction booléenne, associée à ce symbole propositionnel existentiellement quantifié dans le modèle QBF, est égale à **vrai** (resp. **faux**) pour cette combinaison, tandis qu'un atome intermédiaire apparaîtra dans le modèle stable si la sous formule qui lui est associée est interprétée à **vrai** pour cette combinaison tandis qu'il en sera absent si elle est interprétée à **faux** pour cette même combinaison. Cette dernière remarque explique l'absence de toute règle établissant $U = 0$.

Exemple 48 *En continuant l'exemple 1. La QBF* $xi = \forall a \exists d \forall e \exists f \mu$ *avec*

$$\mu = \neg\neg\neg((e \rightarrow d) \rightarrow \neg(f \rightarrow a))$$

est traduite en le programme logique normal au premier ordre

$$\left\{ \begin{array}{ll}
(\leftarrow not\ \mu_\epsilon.), & \\
(\mu_\epsilon \leftarrow \mu_0(0), \mu_0(1).), & (\mu_0(U_1) \leftarrow \mu_{0^2}(U_1, 0), \mu_{0^2}(U_1, 1).), \\
(d(U_1) \leftarrow not\ nd(U_1).), & (nd(U_1) \leftarrow not\ d(U_1).), \\
(f(U_1, U_2) \leftarrow not\ nf(U_1, U_2).), & (nf(U_1, U_2) \leftarrow not\ f(U_1, U_2).), \\
(a(U_1, U_2) \leftarrow U_1 = 1.), & (e(U_1, U_2) \leftarrow U_2 = 1.) \\
(\mu_{0^2}(U_1, U_2) \leftarrow not\ \mu_{0^3}(U_1, U_2).) & (\mu_{0^3}(U_1, U_2) \leftarrow not\ \mu_{0^4}(U_1, U_2).) \\
(\mu_{0^4}(U_1, U_2) \leftarrow not\ \mu_{0^5}(U_1, U_2).) & \\
(\mu_{0^5}(U_1, U_2) \leftarrow not\ \mu_{0^6}(U_1, U_2).) & (\mu_{0^5}(U_1, U_2) \leftarrow \mu_{0^5.1}(U_1, U_2).) \\
(\mu_{0^6}(U_1, U_2) \leftarrow not\ e(U_1, U_2).) & (\mu_{0^6}(U_1, U_2) \leftarrow d(U_1, U_2).) \\
(\mu_{0^5.1}(U_1, U_2) \leftarrow not\ \mu_{0^5.1.0}(U_1, U_2).) & \\
(\mu_{0^5.1.0}(U_1, U_2) \leftarrow not\ f(U_1, U_2).) & (\mu_{0^5.1.0}(U_1, U_2) \leftarrow a(U_1, U_2).)
\end{array} \right\}$$

qui a quatre modèles stables dont le deuxième est

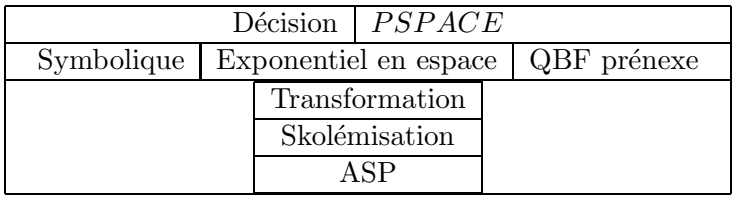
$$\left\{ \begin{array}{l}
\mu_\epsilon, \mu_0(0), \mu_0(1), \mu_{0^2}(1,1), \mu_{0^2}(0,1), \mu_{0^2}(1,0), \mu_{0^2}(0,0), , \\
\mu_{0^4}(1,1), \mu_{0^4}(0,1), \mu_{0^4}(1,0), \mu_{0^4}(0,0), \\
\mu_{0^5.1.0}(1,1), \mu_{0^5.1.0}(0,1), \mu_{0^5.1.0}(1,0), \mu_{0^5.1.0}(0,0), \\
\mu_{0^6}(1,0), \mu_{0^6}(0,0), \mu_{0^6}(1,1), \mu_{0^6}(0,1), \\
e(1,1), e(0,1), a(1,1), a(1,0), \\
d(1), d(0), f(1,1), f(1,0), nf(0,1), nf(0,0)
\end{array} \right\}$$

dans lequel est facilement identifiable le modèle QBF de l'exemple 9 dont le composante fonctionnelle est

$$\begin{array}{lll}
\{ & (d \mapsto \{(\mathbf{vrai} \mapsto \mathbf{vrai}), (\mathbf{faux} \mapsto \mathbf{vrai})\}), & \\
 & (f \mapsto & \{ \quad ((\mathbf{vrai}, \mathbf{vrai}) \mapsto \mathbf{vrai}), ((\mathbf{vrai}, \mathbf{faux}) \mapsto \mathbf{vrai}), \\
 & & \quad\ ((\mathbf{faux}, \mathbf{vrai}) \mapsto \mathbf{faux}), ((\mathbf{faux}, \mathbf{faux}) \mapsto \mathbf{faux})\})\}.
\end{array}$$

Enfin, de même que dans le cas propositionnel, la correction de la méthode garantit que de tout modèle stable du programme généré peut être extrait un modèle QBF pour la formule en oubliant les atomes intermédiaires et la complétude garantit que tout modèle QBF de la formule peut être étendu en un (unique) modèle stable pour le programme. La traduction est à nouveau polynomiale et le programme est de taille polynomiale par rapport à la QBF initiale. La classe complexité PSPACE n'a pour autant pas été ramenée à la classe de complexité NP : l'explosion combinatoire a été repoussée simplement car une phase d'instanciation du programme logique normal au premier ordre est nécessaire et les procédures décidant des programmes logiques normaux ne prennent en entrée que des programmes complètement instanciés. Le résultat formel de la traduction sous forme de programme logique normal au premier ordre suivie d'une ins_tanciation est très similaire à celui de la *skolémisation propositionnelle*. Malheureusement, le résultat opérationnel est beaucoup plus « contrasté » puisque la première approche peine à résoudre des instances faciles tandis que le logiciel `sKizzo` qui implémente la seconde est parmi les implémentations les plus efficaces pour résoudre les QBF. Cela est d'autant plus remarquable que la comparaison du résultat opérationnel pour le niveau propositionnel est beaucoup plus favorable. Une série de tests basée sur l'ensemble de problèmes QG6 [145] (représentant des problèmes de classification d'algèbres finies, problèmes aussi bien codés en formule non-FNC qu'en FNC) a été réalisée en particulier sur la mise en série de la traduction propositionnelle, de l'instanciateur GrinGo [90] et du solveur ASP CLASP[1] [89], et comparée au solveur *SAT* non-FNC SatMate [116] ainsi qu'aux solveurs *SAT* FNC zChaff [151] et MiniSat [73] (MiniSat 2.0 beta). Notre approche fait mieux que le solveur SatMate (SatMate.20.05.2006), qui est pourtant dédié au problème considéré, et zChaff(zChaff 2007.3.12.) ; seul MiniSat fait mieux que notre approche mais dans des temps qui sont comparables. Ces résultats confirment, s'il en était besoin, que le problème *SAT* et le problème du calcul d'un modèle stable appartiennent à la même classe de complexité. Une étude plus poussée dans le cas propositionnel est hors de notre propos. Dans le cas des QBF, les instanciateurs considérés ne détectant pas la structure particulière des programmes sousmis, ils ne peuvent rivaliser avec l'instanciateur dédié de `sKizzo` qui réalise un raisonnement symbolique avant la réelle phase d'instanciation pour la minimiser. Notre approche tirera sans doute les plus grands bénéfices des prochains progrès de la résolution de programmes logiques normaux directement au premier ordre [127]. L'ensemble des tests peuvent être analysés dans [215] et les traducteurs peuvent être obtenus à l'adresse : `http://forge.info.univ-angers.fr/~damota/asp/qbf.php`.

[1] N'est évoqué ici que la meilleure combinaison instanciateur/solveur. L'étude inclut, outre l'ins_tanciateur GrinGo (GrinGo 1.0.0) et le solveur CLASP (CLASP 1.0.1), l'instanciateur Lparse [219] (Lparse 1.0.17) et les solveurs noMoRe++ [2] (noMoRe++ v1.5), smodels [168] (smodels-2.32), ASSAT (ASSAT 2.0 couplé avec MiniSat) et enfin DLV [128] (July 14th, 2006) qui dispose de son propre instanciateur en interne.


Décision
$PSPACE$
Symbolique
Exponentiel en espace
QBF prénexe
Transformation
Skolémisation
ASP

Chapitre 6

Programmation en formules booléennes quantifiées

Il y eut un soir, il y eut un matin : sixième jour.

[62] B. Da Mota, I. Stéphan, and P. Nicolas. Une mise sous forme prénexe préservant les résultats intermédiaires pour les formules booléennes quantifiées. In *Journées Nationales de l'IA Fondamentale (IAF'08)*, 2008.

[63] B. Da Mota, I. Stéphan, and P. Nicolas. A new prenexing strategy for quantified boolean formulae with bi-implications. In *Proceedings of the 6th International Workshop on Constraints in Formal Verification*, 2009.

[64] B. Da Mota, I. Stéphan, and P. Nicolas. Une mise sous forme prénexe préservant les résultats intermédiaires pour les formules booléennes quantifiées. *Revue internationale I3 (Information - Interaction - Intelligence)*, 2009.

Sommaire

Quiconque a enseigné la programmation sait que le problème du programmeur débutant est d'au moins réaliser un programme sans erreur qui réponde à la question tandis que le problème du programmeur confirmé est de choisir parmi la multiplicité des possibilités qui s'offrent à lui, devant rentrer en compte, bien sûr, l'efficacité mais tout autant la lisibilité du code, sa grande réutilisabilité et le temps nécessaire au développement. Cette progression est en partie liée pour l'efficacité à la connaissance acquise sur la manière dont s'exécute le programme : la sémantique opérationnelle influence radicalement la manière de programmer (voir [185] pour de nombreux exemples). La sémantique associée au langage des QBF n'a pas la puissance du calculable en ce qu'il ne peut simuler une *machine de Turing*, il est possible néanmoins de considérer le formalisme des QBF comme un langage de programmation permettant, entre-autres, de coder tout jeu fini à deux joueurs [172, 75, 94, 3, 190]. La communauté QBF s'est très peu penchée sur le lien entre la manière de programmer et l'efficacité de la procédure de décision [24, 3, 190].

Le chapitre 2, présentant un panorama de l'univers des QBF, démontre que les procédures de décision récentes pour le problème de validité des QBF ont abandonné en interne mais seulement en partie la restriction prénexe ; en partie seulement puisque le format d'entrée reste le plus souvent prénexe (sauf pour de rares exceptions). Cette situation est paradoxale puisque, pour des questions d'efficacité, il y a un « aller-retour » des quantificateurs que nous montrons, dans le premier temps de ce chapitre, comme particulièrement préjudiciable justement pour l'efficacité des procédures mais aussi pour l'extraction des solutions.

La pratique de la programmation dans des langages « general purpose » et de nombreuses applications des QBF militent pour la conservation, dans l'ensemble des connecteurs accessibles au programmeur d'une QBF, de la bi-implication (et du ou-exclusif) qui peut être vue comme une égalité voire une affectation. La linéarisation qui élimine la bi-implication (et le ou-exclusif) est trop souvent négligée et supposée effectuée or elle a un énorme impact sur la complexité en espace de la formule si celle-ci n'est pas effectuée avec prudence. Puisque la majorité des procédures de décision pour le problème de validité des QBF de l'état de l'art prennent en entrée des formules prénexes, nous nous intéressons dans le second temps de ce chapitre, à la conversion d'une QBF non prénexe contenant des bi-implications vers le fragment prénexe.

6.1 De l'abandon du fragment prénexe

Si dans une première approche liée à la proximité du problème SAT, le fragment prénexe pour les QBF a pu apparaître comme un fragment en interne efficace pour des systèmes (ou comme langage vers lequel sont compilés d'autres formalismes), avec du recul, ce n'est plus le cas : la mise sous forme prénexe est non-déterministe et le choix de la stratégie influence grandement l'efficacité des systèmes [78], de plus, elle augmente l'espace de recherche en faisant dépendre des symboles propositionnels existentiellement quantifiés de symboles universellement quantifiés qui n'ont « en réalité » aucun lien entre-eux, d'ailleurs de nombreux systèmes ajoutent *ab initio* une phase de *miniscoping* associée à un *arbre de quantificateurs* pour recouvrer les portées perdues de ceux-ci [81, 17, 26, 109] et

le *retour-arrière intelligent* ignore certaines fausses *dépendances* qui ont pu être créées [80].

Le fragment prénexe n'est évidemment pas le bon du point de vue du programmeur en QBF : rarement un problème s'exprime directement de manière prénexe [77, 34] car la portée locale du quantificateur fait sens pour le programmeur, mais surtout dans la solution elle exprime (généralement) une dépendance. Un excellent exemple est celui de l'« omelette » qui nous est servie dans la littérature sur les « diagrammes d'influence possibilistes » [87] : « Le problème est de faire une omelette à six œufs à partir d'une omelette à cinq œufs, le nouvel œuf pouvant être soit frais, soit pourri. » Il est des événements incertains tels que « l'œuf est frais » régis par des possibilités et qui sont codées de manière universelle dans le cas de la recherche d'une stratégie pessimiste optimale et des décisions telles que « vais-je casser le sixième œuf à part pour l'inspecter ? » qui sont codées de manière existentielle. Ce qui nous intéresse dans cette exemple est que, selon que l'œuf est ou non cassé à part, l'espace des possibles n'est plus le même : s'il est cassé à part, deux possibilités s'offrent à nous : l'œuf est jugé frais ou non ; tandis que s'il n'est pas cassé à part, l'état de l'œuf restera indéterminé. La démonstration de $PSPACE$-complétude du calcul de la stratégie pessimiste optimale donne une manière de coder le problème en une QBF sous FNC prénexe, mais qu'en est-il des solutions extraites ? Certaines d'entre-elles feront dépendre des symboles propositionnels existentiellement quantifiés de symboles universellement quantifiés représentant des événements qui ne sont même pas dans le champs du possible et des symboles propositionnels existentiellement quantifiés représenteront des décisions sans sémantique qui n'auront aucune influence sur celles-ci.

L'abandon de la forme prénexe en entrée n'a que peu d'impact en interne sur l'extraction des solutions si aucun quantificateur n'apparaît dans une sous-formule d'une négation explicite ou implicite (c'est-à-dire en partie gauche d'une implication ou dans une bi-implication ou un ou-exclusif). Dans le cas contraire, c'est la définition même de solution qui est discutable : lors de la mise sous forme prénexe la polarité d'un quantificateur change à chaque « traversée » par les *équivalences logiques* $\neg(\forall x\ F) \overset{1.3}{\equiv} (\exists x\ \neg F)$ et $\neg(\exists x\ F) \overset{1.4}{\equiv} (\forall x\ \neg F)$ en conséquence, ce qui a été exprimé dans la formule comme étant un coup du joueur « existentiel » devient un coup du joueur « universel » et réciproquement. La sémantique des QBF prénexes présentée dans le chapitre 1 est alors insuffisante et doit être étendue.

6.2 De la bi-implication

L'introduction du chapitre 4 confère des arguments historiques et de proximité au problème SAT au choix du fragment prénexe FNC et ce qui précède montre qu'au moins la restriction au prénexe doit être enlevée. Ce qui a prévalu à la conception des premiers algorithmes pour les QBF était d'un côté un argument d'ordre théorique (le fragment prénexe FNC est complet) et un argument pratique (le fragment FNC pour le problème SAT est amplement implémenté), mais qu'en est-il du point de vue du programmeur en QBF ? Bien sûr, celui-ci désire le langage le plus expressif possible. Pourquoi ne pas choisir un fragment non-prénexe restreint à la négation, la conjonction, la disjonction et l'implication (cette dernière pouvant se convertir aisément en FNN ; cf. la mise sous FNN au

chapitre 1) comme le format pour la compétition QBFEVAL 2010, session « non prénexe, non FNC » [102] ? Parce que c'est se priver principalement du connecteur de bi-implication qui est de première nécessité dans la mise en équivalence de propriétés, comme par exemple pour le « bounded model checking » [16, 17] et d'un outil de programmation indispensable qui permet de capturer des résultats intermédiaires ; et c'est donc naturellement que ce connecteur est inclus dans le langage exploité au chapitre 4. Il n'en demeure pas moins que les formats d'entrée des procédures de décision du panorama de l'univers des QBF décrit au chapitre 2 excluent la bi-implication (et le ou-exclusif) et considère que la *linéarisation* (cf. § 1.3.3) a déjà eu lieu. Il y a donc dans la littérature sur les QBF un vide non pas théorique mais purement pratique entre l'expression par le programmeur en une QBF quelconque de son problème et la version soumise en entrée à une procédure de décision n'incluant pas la bi-implication dans son ensemble de connecteurs. En effet, la linéarisation est techniquement très simple : $(F \leftrightarrow G) \overset{0.3}{\equiv} ((F \rightarrow G) \wedge (G \rightarrow F))$; mais elle induit une croissance exponentielle de la taille de la QBF. Pire, elle induit aussi une croissance exponentielle du nombre de quantificateurs pour ceux qui sont nichés dans les sous-formules des bi-implications avec un changement de polarité pour une des deux occurrences des symboles propositionnels qui doit être renommée lors de la mise en forme prénexe (x' un symbole propositionnel n'apparaissant ni dans F, ni dans G alors $((\exists x\ F) \leftrightarrow G) \equiv (\exists x\ (\forall x'\ (([x \leftarrow x'](F) \rightarrow G) \wedge (G \rightarrow F)))))$. Ces nouveaux quantificateurs font exploser l'espace de recherche et le renommage n'est ni exploité par la procédure de décision qui en ignore l'existence ni n'a de sens pour le programmeur en QBF.

Nous nous sommes donc intéressé à la linéarisation dans l'optique dans étudier ses effets délétères et, dans la mesure du possible, de proposer une méthodologie de mise en forme prénexe dans le cas de quantificateurs nichés dans les sous-formules des bi-implications. En particulier, nous nous sommes focalisé sur des QBF issues de la vérification formelle d'équivalence (par une bi-implication) entre la structure d'un circuit et son comportement souhaité décrits par deux QBF à symboles propositionnels libres (les entrées et la sortie du circuit) et clôturée universellement. Dans les QBF qui représentent les deux circuits, des symboles existentiellement quantifiés capturent naturellement les connexions entre les sorties et les entrées de sous-circuits encodés en formules propositionnelles [18, 17]. L'introduction des symboles existentiellement quantifiés est nécessaire pour éviter une duplication de l'encodage des sous-circuits. La conversion d'une telle QBF sous FNC prénexe fait croître dans le pire des cas exponentiellement et le nombre de connecteurs et le nombre de quantificateurs dans la QBF FNC prénexe par rapport à la QBF initiale : les quantificateurs existentiels sont disséminés non seulement sous leur polarité d'origine mais aussi universellement et sous de nouveaux symboles, la structure de la QBF initiale est complètement perdue. La conséquence en terme de complexité en temps et en espace pour les systèmes de l'état de l'art est évidemment catastrophique. Nous avons donc proposé une méthodologie pour extraire les bi-implications qui définissent ces sous-circuits et réduire ainsi les dépendances entre symboles propositionnels pour réduire l'espace de recherche. Le principal résultat est un algorithme qui applique l'équivalence suivante (F, G et H trois QBF et x un symbole propositionnel qui représente le résultat intermédiaire F, avec

$x \notin \mathcal{SPL}(H)$, $x \notin \mathcal{SPL}(F)$)

$$(H \leftrightarrow (\exists x\ ((x \leftrightarrow F) \wedge G))) \equiv (\exists x\ ((x \leftrightarrow F) \wedge (H \leftrightarrow G)))$$

jusqu'à extraire tous les motifs possibles. Les expériences menées sur la mise en équivalence entre la structure et le comportement de l'additionneur n-bits encodée à partir d'une spécification en logique monadique [18, 17] montrent les gains indiscutables aussi bien en espace qu'en temps. Nos expériences sur un exemple théorique *ad hoc* montrent en outre le passage d'une complexité exponentielle à une complexité polynomiale en temps. Mais en général, la linéarisation de la formule induit une croissance, dans le pire des cas, exponentielle de la formule. L'ensemble des tests peuvent être obtenus à l'adresse : `http://forge.info.univ-angers.fr/~damota/qbf/`, analysés dans [64] et les algorithmes implantés en *Prolog* peuvent être obtenus à l'adresse :
`http://www.info.univ-angers.fr/pub/stephan/ri3_10.html`.

Conclusion générale et perspectives

. . .il chôma, le septième jour, après tout le travail qu'il avait fait.

Dans ce « Propos sur les formules booléennes quantifiées », j'ai décrit mes travaux personnels et en collaboration dont l'objectif est de mieux appréhender le domaine des formules booléennes quantifiées. En particulier, je me suis attaché à développer aussi bien les aspects formels (au chapitre 1 sur les sémantiques et le calcul syntaxique S_{QBF} ainsi qu'au chapitre 3 sur le nouveau formalisme des bases littérales) que calculatoires (au chapitre 4 pour la procédure de décision par propagation `pqbf` et au chapitre 5 pour la traduction des QBF en des programmes ASP) et pragmatiques (au chapitre 6 pour le choix de fragments de langage en interne et en entrée des procédures de décision) tout en replaçant mes travaux dans ceux de la communauté (au chapitre 2 qui dresse le panorama sur le problème de validité des QBF).

Ce document ne rapporte pas un travail en cours : celui sur la mise au point d'outils parallèles pour les QBF qui fait l'objet de la thèse, que je coencadre avec le Professeur Pascal Nicolas, de Benoit Da Mota qui lui appartient de présenter en détail et dont je n'esquisse ci-dessous que l'idée maîtresse [58, 59].

En ce qui concerne le panorama dressé des procédures de décision exploitant le parallélisme (cf. § 2.4), nous faisons deux remarques : ces procédures sont toutes dédiées aux QBF sous FNC prénexes et elles sont toutes basées une *stratégie de partitionnement sémantique*. Nous proposons une nouvelle approche : l'« architecture de parallélisation syntaxique ». Celle-ci n'est pas basée sur le partitionnement de l'espace de recherche selon la sémantique des quantificateurs mais selon la localité de l'information dans les QBF non prénexes (et non FNC) représentée par des sous-formules quantifiées à symboles propositionnels libres. Cette approche revient à l'idée initiale de la *hiérarchie polynomiale* en ce qu'un *oracle* est nécessaire mais non plus pour *décider* de la validité d'un sous-problème mais pour en calculer une formule propositionnelle équivalente, fonction de ses symboles propositionnels libres. L'hypothèse à valider est que la stratégie de partitionnement syntaxique exploite *a priori* la localité de l'information, évite le parcours redondant d'une partie de l'espace de recherche et se substitue ainsi à l'*apprentissage de lemme*.

Les perspectives qui s'offrent à nous sont nombreuses, elles s'articulent autour de la résolution de *problèmes de satisfaction de contraintes quantifiées* (QCSP), de l'introduction de métaheuristiques pour réduire l'espace de recherche et de l'étude des aspects liés à la représentation des connaissances.

Le problème de satisfaction de contraintes quantifiées est au *problème de satisfaction de contraintes*, ce que le problème de validité d'une QBF est au problème SAT : une extension par l'introduction de la possibilité de quantifier les variables universellement. Une nouvelle perspective de recherche est donc l'extension des résultats de ce document aux

QCSP. D'un côté technique, la refonte de la procédure de décision `pqbf` dans une architecture ouverte de propagation de contraintes permet d'envisager l'étude d'un langage combinant le domaine des booléens quantifiés à d'autres domaines comme proposé dans QCSP+ [29, 30, 32, 31]. D'un côté théorique, l'extension des travaux sur les bases littérales offriraient un cadre pour l'étude des certificats et de la compilation des QCSP.

Les métaheuristiques ont connu un réel succès pour le calcul des solutions de problèmes combinatoires de la classe $\mathcal{NP}$-complet mais les premières rares études dans le cadre des QBF ont montré leurs insuffisances. La raison en est simple : l'oracle (et dans le cadre des métaheuristiques, la procédure qui vérifie une solution potentielle) est polynomial en temps pour les problèmes de la classe $\mathcal{NP}$-complet tandis qu'il est $co-\mathcal{NP}$-complet pour les classes Σ_k^p et Π_k^p avec $k > 1$ (cf. § 1.2.2). Néanmoins, nous pensons que des métaheuristiques telles que les *colonies de fourmis* [72, 39] n'ont pas été suffisamment étudiées dans le cadre des QBF. Nous avons mené des expériences, dans le cadre d'un stage de master recherche, qui n'ont été que malheureusement peu concluantes mais c'était au début de notre intérêt pour les QBF et notre compréhension du domaine a très largement évoluée. Nous pensons aussi que des méthodes locales du style *méthode tabou* [110] peuvent être favorablement utilisée dans une procédure incomplète basée sur l'élagage de l'espace de recherche (considéré comme un *arbre sémantique*) pour les quantificateurs universels telle que l'*abstract branching*.

Le formalisme des QBF est un excellent compromis entre les aspects calculatoires, déjà largement explorés, et le pouvoir d'expression dans le cadre de la représentation de connaissances. Ce dernier a été relativement peu abordé au sein de la communauté QBF, en particulier le calcul des solutions dans le cas des QBF quelconques. Nous avons présenter dans le chapitre 1 une sémantique des QBF non-prénexes décisionnelle et une sémantique des QBF prénexes qui permet de manipuler des solutions au problème de validité des QBF prénexes ; dans le chapitre 6 sur la programmation en QBF, nous avons mis en exergue la nécessité d'une sémantique permettant de manipuler des solutions au problème de validité des QBF quelconques : celle-ci devra étendre les deux précédentes. La problématique est la suivante : dans la vision d'un jeu à deux joueurs, le quantificateur existentiel désigne-t-il toujours le même joueur ? La réponse n'est *a priori* pas évidente car dans un jeu qui inclut un « qui perd gagne » par une négation sur une partie des règles, des symboles existentiellement quantifiés de la solution extraite après une mise sous forme prénexe de la QBF passent dans le camp adverse et réciproquement. La question est similaire si ce sont les bases littérales du chapitre 3 qui sont étendues pour compiler des QBF non-prénexes. La base littérale ne différenciant pas les quantificateurs entre-eux dans sa construction linéaire, elle apparaît plus facile a étendre à une version arborescente basée sur le squelette des quantificateurs sous-jacent à la QBF.

J'ai présenté dans ce document six années de recherche dédiées à une meilleur compréhension, personnelle et pour la communauté scientifique, du domaine des formules booléennes quantifiées. Finalement, quel en a été le « fil rouge » : c'est l'équivalence, qu'elle soit présente dans le langage étudié, où elle est nommée « bi-implication », que le langage qui permet de l'étudier. Elle est tour-à-tour nouvelle pour partitionner les solutions des

formules booléennes quantifiées, cachée mais présente pour définir le calcul syntaxique, définitionnelle pour la compilation, centrale dans le motif exploité par la procédure de décision et enfin réhabilitée en tant qu'outil de programmation. J'espère que cet étude pourra, au-delà de la nécessité de faire le point sur mes connaissances et du plaisir qu'elle m'a procuré en les exprimant, servir à autrui.

Deuxième partie

Glossaire

PROP : en lien avec la *logique propositionnelle* ; PRED : en lien avec la *logique des prédicats* ; QBF : en lien avec les formules booléennes quantifiées ; Morph : définition de morphologie ; Syn : définition de syntaxe ; Sem : définition de sémantique ; Langage : définition de théorie des langages.

ε**.** cf. définition 5.

$\prec$**.** cf. définition 6.

$\equiv$**.** cf. définition 23.

$\cong$**.** cf. définition 24.

$\underline{|.|}$**.** cf. définition 7.

(.)**.** cf. définition 3.

.| cf. définition 5.

Σ_k − **QBF.** cf. définition 5.

Σ_k^p**.** cf. *hiérarchie polynomiale.*

Σ_k^p**-complet .** cf. *hiérarchie polynomiale.*

Π_k − **QBF.** cf. définition 5.

Π_k^p**.** cf. *hiérarchie polynomiale.*

Π_k^p**-complet .** cf. *hiérarchie polynomiale.*

$\bot$**. 1.** Morph**.** cf. définition 1. **2.** Sem**.** cf. définition 13.

$\top$**. 1.** Morph**.** cf. définition 1. **2.** Sem**.** cf. définition 13.

$\wedge$**. 1.** Morph**.** cf. définition 1. **2.** Sem**.** cf. définition 13.

$\vee$**. 1.** Morph**.** cf. définition 1. **2.** Sem**.** cf. définition 13.

$\leftrightarrow$**. 1.** Morph**.** cf. définition 1. **2.** Sem**.** cf. définition 13.

$\oplus$**. 1.** Morph**.** cf. définition 1. **2.** Sem**.** cf. définition 13.

$\neg$**. 1.** Morph**.** cf. définition 1. **2.** Sem**.** cf. définition 13.

$\rightarrow$**. 1.** Morph**.** cf. définition 1. **2.** Sem**.** cf. définition 13.

$\exists$**. 1.** Morph**.** cf. définition 1. **2.** Sem**.** cf. définition 13.

$\forall$**. 1.** Morph**.** cf. définition 1. **2.** Sem**.** cf. définition 13.

ϕ**.** cf. définition 17.

ν**.** cf. définition 17.

$\langle . \mid . \rangle$**.** cf définition 34.

$(.)^*$**.** cf définition 35.

$\models$**.** cf. *sémantique de la logique propositionnelle.*

$\not\models$**.** cf. *sémantique de la logique propositionnelle.*

$[. \leftarrow .](.)$**.** cf. définition 9.

$[. := .]$**.** cf. définition 12.

$(. \circ_x .)$**.** cf. définition 40.

$\otimes$**.** cf. définition 41.

$\approx$**.** cf. définition 39.

$\sim$**.** cf. définition 44.

`2clsQ`**.** [193] `2clsQ` est une *procédure de décision* pour le *problème de validité* des QBF *prénexes* dont la *matrice* est sous *FNC* basée sur un `QDLL` dont la *propagation de clauses unitaires* est augmentée par de l'*hyper résolution binaire* pour la génération de *clauses unitaires*. La procédure intègre aussi une forme binaire de *raisonnement sur les équivalences.*

2clsQ

Décision	*PSPACE*	
Sémantique	Polynomial en espace	FNC prénexe
	Monolithique	
	FNC prénexe ∃-learning ∀-learning Hyper résolution binaire Réduction universelle	

2QBF. 1. Une *2QBF* est une QBF *close prénexe* dont la *matrice* est sous *FNC* et dont chaque *clause* contient au plus deux *littéraux*. Le fragment syntaxique des 2QBF est linéaire en temps, ce que met en exergue l'algorithme suivant [4] qui réduit le problème de validité pour les 2QBF à un problème de calcul des composantes fortement connexes dans un graphe (les sous-ensembles de sommets tels qu'il existe un chemin d'un sommet à tout autre) qui est lui-même linéaire en temps [221]. Chacune des clauses $(x \vee y)$ de la 2QBF est considérée comme une paire d'arcs $(\overline{x}, y)$ et $(\overline{y}, x)$ dans un graphe dont l'ensemble des sommets est formé de l'ensemble des littéraux obtenus à partir des *symboles propositionnels* de la QBF (les clauses ne contenant qu'un seul littéral l forment des pairs d'arcs $(\overline{l}, l)$ et $(l, \overline{l})$). La 2QBF est valide si aucune des trois conditions suivantes n'est vérifiée : (i) un sommet associé à un symbole propositionnel *existentiellement quantifié* appartient à la même composante fortement connexe que son complémentaire ; (ii) un sommet associé à un symbole propositionnel *universellement quantifié* appartient à la même composante fortement connexe qu'un sommet associé à un symbole propositionnel existentiellement *quantifié plus externe* ; (iii) il existe un chemin entre deux sommets associés à des symboles propositionnels universellement quantifiés. **2.** Dans [183, 115] les *2QBF* sont des QBF prénexes de lieur ∀∃.

AB. [28] AB est une *procédure de décision* pour le *problème de validité* des QBF *prénexes* dont la *matrice* est sous *FNC* basée sur un QDLL qui implémente l'*abstract branching*.

AB

Décision	*PSPACE*	
Sémantique	Exponentiel en espace	FNC prénexe
	Monolithique	
	∃-backtrack Abstract Branching Simplification ⊤ et ⊥	

abstract branching. [28] L'*abstract branching* est une technique par élagage de l'espace de recherche (qui est considéré comme un *arbre sémantique*) pour les quantificateurs universels grâce aux *scénarii* : à partir d'un scénario, d'autres sont calculés

en conservant uniquement la partie existentielle ; ces nouveaux scénarii sont ôtés de l'espace de recherche pour les quantificateurs universels. Une procédure de décision basée sur l'*abstract branching* prouve la *validité* d'une QBF close lorsque l'espace de recherche pour les quantificateurs universels a été totalement élagué et prouve la non-validité lorsque l'espace de recherche pour les quantificateurs existentiels a été totalement épuisé. La représentation explicite de l'espace de recherche pour les quantificateurs universels est dans le pire des cas exponentielle.

ACNF. cf. *apprentissage de lemme*.

AIG. cf. *`AIGsolve`*.

`AIGsolve`. [175] `AIGsolve` est une *procédure de décision* pour le *problème de validité* des QBF *prénexes* dont la *matrice* est sous *FNC*. Le système `AIGsolve` applique sur la QBF sous FNC prénexe une série de pré-traitements : *propagation de clauses unitaires*, *réduction universelle*, *propagation de littéraux monotones*, *raisonnement sur équivalence*. La QBF ainsi réduite est soumise à la recherche du motif $(y \leftrightarrow f(x_1, \ldots, x_n))$ de définition fonctionnel sous FNC : si les paramètres de la fonction sont en accord avec le lieur (c'est-à-dire pour tout x_i, $x_i \prec y$) alors

$$Q \exists y Q' ((y \leftrightarrow f(x_1, \ldots, x_n)) \wedge g(y, x_1, \ldots, x_n)) \equiv Q \exists y Q' g(f(x_1, \ldots, x_n), x_1, \ldots, x_n)$$

et $Q \forall y Q' ((y \leftrightarrow f(x_1, \ldots, x_n)) \wedge g(y, x_1, \ldots, x_n))$ est non-valide (sinon il ne se passe rien). Une fois les définitions exploitées, la formule est transformée en un *AIG* (c'est-à-dire And-Inverter-Graph, *graphe acyclique orienté* ne contenant que des conjonctions et des négations). L'*élimination de quantificateurs* est réalisée par une *expansion bottom-up* qui rend la procédure de décision exponentielle en espace. Le système `AIGsolve` intègre un *arbre de quantificateurs* obtenu, à l'instar de *`sKizzo`*, par *miniscoping*.

`AIGsolve`

Décision	$PSPACE$	
Symbolique	Exponentiel en espace	FNC prénexe
	Monolithique	
	DAG $\exists$-expansion$\uparrow$ $\forall$-expansion$\uparrow$	

algorithme AC3. cf. *problèmes de satisfaction de contraintes*.

algorithme de Stålmarck. [202] L'« algorithme de Stålmarck » est une procédure de décision pour le problème $TAUT$ de complexité $co - \mathcal{NP}$-complet et donc correspond au fragment des QBF prénexes avec uniquement des symboles propositionnels universellement quantifiés. L'algorithme de Stålmarck tente de prouver que la matrice de la QBF prénexe ne peut pas être équivalente à $\bot$. Cet algorithme utilise d'abord la *décomposition* par introduction de *littéraux* (existentiellement quantifiés) et ensuite applique un ensemble de *règles*. Cet algorithme considère les négations $\neg x$ codées en $(x \rightarrow \bot)$ (ce qui peut être calculé grâce aux *équivalences logiques* (0.1), (0.20) et (0.15)) et ne couvre que l'ensemble des formules construites sur $\rightarrow$ et $\bot$. Ainsi, l'ensemble initial des *schémas* et règles de Stålmarck sont pour l'implication ;

nous les transformons (cf. tables ci-dessous) pour les comparer aux règles de la propagation. Dans la table 4.2, les règles sont identifiées par le couple (colonne,ligne) : la règle r_1 couvre les règles (1,21) et (2,21) mais aussi (1,6), (1,9) et (1,18) ; la règle r_2 couvre les règles (1,13) et (1,14) mais aussi (1,12), (2,1), (2,5) et (2,12) ; la règle r_3 couvre les règles (1,3) et (1,4) mais aussi (1,1), (1,2), (2,1) et (2,2) ; la règle r_4 couvre les règles (1,7) et (1,8) mais aussi (1,5) et (2,6) ; la règle r_5 couvre les règles (1,10) et (1,11) mais aussi (1,9) et (2,9) ; la règle r_6 couvre (imparfaitement) les règles (1,23) et (2,23) mais aussi (1,15) et (1,12) ; la règle r_7 couvre les règles (1,16) et (1,17) mais aussi (1,15). Les règles (1,19) (ou (1,20) par permutatibilié des quantificateurs existentiels), (2,15), (2,18) et (1,26) (ou (2,26) par permutatibilié des quantificateurs existentiels) ne sont couvertes par aucune règle de l'algorithme de Stålmarck et peuvent y être ajoutées.

Les règles de l'algorithme de Stålmarck		
	Schémas	Substitutions
r_1	$((x \rightarrow y) \leftrightarrow \bot)$	$[x \leftarrow \top][y \leftarrow \bot]$
r_2	$((x \rightarrow \top) \leftrightarrow z)$	$[z \leftarrow \top]$
r_3	$((\bot \rightarrow y) \leftrightarrow z)$	$[z \leftarrow \top]$
r_4	$((\top \rightarrow y) \leftrightarrow z)$	$[z \leftarrow y]$
r_5	$((x \rightarrow \bot) \leftrightarrow z)$	$[z \leftarrow \overline{x}]$
r_6	$((x \rightarrow y) \leftrightarrow x)$	$[x \leftarrow \top]$
r_7	$((x \rightarrow x) \leftrightarrow z)$	$[z \leftarrow \top]$

La méthode de Stålmarck inclut aussi des schémas contradictoires (appelés « triplets terminaux »), ils sont aussi exprimés en fonction de l'implication et donnés ci-dessous pour comparaison. Les schémas contradictoires couvrent plus de cas que les schémas terminaux de Stålmarck et les suivants peuvent être ajoutés : $((x \rightarrow x) \leftrightarrow \bot)$, $((\top \rightarrow y) \leftrightarrow \overline{y})$, $((x \rightarrow \overline{x}) \leftrightarrow \overline{x})$ et $((x \rightarrow \bot) \leftrightarrow x)$.

Triplets terminaux de l'algorithme de Stålmarck
$((\top \rightarrow \bot) \leftrightarrow \top)$
$((\bot \rightarrow y) \leftrightarrow \bot)$
$((x \rightarrow \top) \leftrightarrow \bot)$

L'algorithme de Stålmarck prouve qu'une formule propositionnelle est une tautologie en prouvant qu'il est impossible de la *falsifiée* ; nous ne pouvons faire de même pour une QBF dans le cas général car il existe des QBF qui n'ont pas de *modèle QBF* mais dont la matrice a au moins un *modèle* propositionnel (par exemple, la QBF $\forall y \forall z \exists x((x \rightarrow y) \leftrightarrow z)$).

alphabet. cf. *machine de Turing.*

alphabet des QBF. cf. définition 1.

Answer Set Programming. cf. *programmation par ensembles réponses.*

apprentissage de lemme. L'*apprentissage de lemme* (en. *learning*) en QBF est une technique issue de la résolution du problème *SAT* sous *FNC* [144] (et cela, avant du domaine des *problèmes de satisfaction de contraintes* [22, 21]). L'apprentissage

de lemme a été introduit, dans le domaine des QBF, quasiment simultanément pour les *procédures de décision* `Quaffle` [230], `Semprop` [130] et `QUBE` [104] dans le cas de l'échec et du quantificateur existentiel sous la forme de « conflict learning » [228], « caching lemmas » [130], « nogoods learning » [104, 108] et dans le cas du succès et du quantificateur universel (« solution learning ») sous la forme de « satisfiability-directed learning» [230], « caching models » [130], « goods learning » [104, 108]. Dans le cas de l'échec, c'est une *clause* qui est apprise à partir des symboles propositionnels existentiellement quantifiés qui ont mené à l'échec. Dans le cas du succès, c'est un *cube* qui est appris à partir des symboles propositionnels universellement quantifiés qui ont mené à ce succès. L'information capturée par ces clauses et cubes était déjà présente mais en intention dans la formule initiale et est donc exprimée sous forme de lemme pour éviter de recalculer sous une forme ou une autre à nouveau cette information ; ce principe est à rapprocher de la *règle de la coupure* dans le *calcul des séquents*. L'apprentissage de lemme s'intègre dans la modification du *retour-arrière* (chronologique) comme étant la technique de *retour-arrière intelligent* par excellence et donc dans des procédures basées sur `QDLL`. Dans le cas où la procédure de décision est sur le fragment des QBF *prénexes* dont la *matrice* est sous *FNC*, l'intégration des cubes sort du modèle ; celui-ci a donc été modifié par la manipulation, en interne, de formules sous *forme normale conjonctive augmentée* (ou *FNCA*, en. *augmented conjunctive normal form* ou *ACNF*) qui augmente la FNC par l'ajout (conjonctivement ou disjonctivement) d'« une » (c'est-à-dire un ensemble de) disjonction(s) de cubes (initialement vide). La génération des clauses (resp. de cubes) est assurée par l'application de la *Q-résolution* (resp. du dual de la Q-résolution) à partir des littéraux de la clause (resp. du cube) qui a été réduite en la *clause vide* (resp. le *cube vide*) et des littéraux des clauses (resp. cubes) qui ont été réduites à des *clauses unitaires* (resp. *cubes unitaires*) [228]. La génération des cubes est assurée de manière parfaitement duale [230]. Les trois formes d'apprentissage de lemme introduites quasiment simultanément ne permettent pas de capturer tous les lemmes possibles, dans `CLearn` [97] sont introduites deux techniques d'apprentissage de lemme (l'une locale et l'autre globale par persistance des lemmes) basée sur l'introduction de symboles propositionnels intermédiaires qui capturent plus finement les *dépendances*. Le fragment des formules QBF sous FNC montre ici clairement ses limites pour les QBF : s'il est bien adapté à la résolution du problème *SAT* (qui ne traite que des symboles propositionnels existentiellement quantifiés [8]), sa dissymétrie par rapport au traitement des conjonctions et disjonctions a amené à s'intéresser à l'apprentissage de lemme pour une combinaison FNC/FND (CCDNF) dans `IQTest` [227].

`AQME`. [180, 182] `AQME` pour « Adaptive QBF Multi-Engine » est une *procédure de décision* pour le *problème de validité* des QBF *prénexes* dont la *matrice* est sous *FNC* qui fait appel à différentes procédures pour le même problème en choisissant la procédure considérée comme la mieux adaptée par une *heuristique*.

arbre de déduction. **1.** QBF. cf. définition 28. **2.** PROP. cf. *calcul des séquents*.

arbre de preuve. **1.** QBF. cf. définition 30. **2.** PROP. cf. *calcul des séquents*.

arbre de quantificateurs. cf. *dépendance*.

arbre sémantique. [130] Un *arbre sémantique* (en. *semantic tree*) est un arbre binaire dans lequel chaque paire d'arcs est étiquetée avec un *littéral* et son *complémentaire*. Tout nœud, de nœud-père associé à une *formule propositionnelle* F (ou une QBF $(qx\ F)$) et d'arc entrant étiqueté par le littéral x (resp. le littéral $\neg x$), est associé à la formule $[x \leftarrow \top](F)$ (resp. $[x \leftarrow \bot](F)$).

assignation. cf. *sémantique de la logique des prédicats.*

ASP. cf. *programmation par ensembles réponses.*

atome. Si p est un *symbole de prédicat* sans argument alors p est un *atome* ; si p est un symbole de prédicat avec n arguments, $n > 0$, et $t_1, \ldots, t_n$ sont des *termes* alors $pt_1 \ldots t_n$ est un *atome.*

augmented conjunctive normal form. cf. *apprentissage de lemme.*

axiome. **1.** QBF. cf. définition 29. **2.** PROP. cf. *calcul des séquents.*

backjumping. cf. *retour-arrière.*

base littérale. cf. définition 34.

BDD. cf. *diagramme de décision binaire.*

bi-implication. cf. définition 1.

$\mathcal{BL}$. cf. définition 34.

BLO. cf. définition 37.

BOOL cf. définition 11.

booléen. cf. définition 11.

bottom-up. cf. §. 1.5.

$\mathcal{C}$. cf. définition 42.

caching lemma. cf. *apprentissage de lemme.*

caching model. cf. *apprentissage de lemme.*

calculable. cf. *machine de Turing.*

Calcul des séquents. [86] Le *calcul des séquents* est un formalisme qui permet de représenter des systèmes de déduction. Un *séquent* est une paire $(\Gamma \Rightarrow \Delta)$, avec Γ et Δ deux ensembles, possiblement vide, de formules. Γ est l'*antécédent* (ou partie gauche) et Δ le *conséquent* (ou partie droite). Une *règle d'inférence* est constituée d'un ensemble de *séquents prémisses* $\{\Gamma_1 \Rightarrow \Delta_1, \ldots, \Gamma_n \Rightarrow \Delta_n\}$ et d'un *séquent conclusion* $(\Gamma \Rightarrow \Delta)$:

$$\frac{\Gamma_1 \Rightarrow \Delta_1 \ldots \Gamma_n \Rightarrow \Delta_n}{\Gamma \Rightarrow \Delta}$$

La *formule principale* est la formule sur laquelle s'applique une instance d'une règle d'inférence. Si cette formule apparaît dans le conséquent alors cette règle

d'inférence est une *règle d'élimination ou d'introduction à droite* sinon c'est une *règle d'élimination ou d'introduction à gauche*. Un système d'inférence est constitué d'un ensemble de règles d'inférence et d'un ensemble de séquents appelés *axiomes*. Un séquent $\Rightarrow$ est un *séquent contre-exemple* s'il ne contient plus aucun *connecteur logique* et s'il n'est pas un axiome. A partir d'un séquent *racine* se développe un *arbre de déduction* qui est un *arbre de preuve* si toutes les *feuilles* de l'arbre sont des axiomes et un arbre contre-exemple si toutes les feuilles sont des axiomes ou des séquents contre-exemples avec au moins un séquent contre-exemple. **Sem PRED.** Un séquent $(A, \ldots, A_n \Rightarrow B_1, \ldots, B_m)$ est *falsifiable* s'il existe une *structure* $\mathcal{S} = (I, \mathcal{D})$ et une *assignation* α telle que

$$I^*((\bigwedge_{1\leq i\leq n} A_i \rightarrow \bigvee_{1\leq i\leq m} B_i))(\alpha) = \textbf{faux}.$$

Un séquent $(A, \ldots, A_n \Rightarrow B_1, \ldots, B_m)$ est *valide* si

$$\models (\bigwedge_{1\leq i\leq n} A_i \rightarrow \bigvee_{1\leq i\leq m} B_i).$$

Voici ci-dessous le *système de séquents* $\mathbb{G}$ de Gentzen qui est *correct* et *complet* vis-à-vis de la *sémantique de la logique des prédicats*.

$$\frac{A, B, \Gamma \Rightarrow \Delta}{(A \wedge B), \Gamma \Rightarrow \Delta}(\wedge g) \qquad \frac{\Gamma \Rightarrow A, \Delta \quad \Gamma \Rightarrow B, \Delta}{\Gamma \Rightarrow (A \wedge B), \Delta}(\mathbb{G} \wedge d)$$

$$\frac{A, \Gamma \Rightarrow \Delta \quad B, \Gamma \Rightarrow \Delta}{(A \vee B), \Gamma \Rightarrow \Delta}(\mathbb{G} \vee g) \qquad \frac{\Gamma \Rightarrow A, B, \Delta}{\Gamma \Rightarrow (A \vee B), \Delta}(\mathbb{G} \vee d)$$

$$\frac{\Gamma \Rightarrow A, \Delta \quad B, \Gamma \Rightarrow \Delta}{(A \rightarrow B), \Gamma \Rightarrow \Delta}(\mathbb{G} \rightarrow g) \qquad \frac{A, \Gamma \Rightarrow B, \Delta}{\Gamma \Rightarrow (A \rightarrow B), \Delta}(\mathbb{G} \rightarrow d)$$

$$\frac{\Gamma \Rightarrow A, \Delta}{\neg A, \Gamma \Rightarrow \Delta}(\mathbb{G} \neg g) \qquad \frac{A, \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \neg A, \Delta}(\mathbb{G} \neg d)$$

$$\frac{A, B, \Gamma \Rightarrow \Delta \quad \Gamma \Rightarrow A, B, \Delta}{(A \leftrightarrow B), \Gamma \Rightarrow \Delta}(\mathbb{G} \leftrightarrow g) \qquad \frac{A, \Gamma \Rightarrow B, \Delta \quad B, \Gamma \Rightarrow A, \Delta}{\Gamma \Rightarrow (A \leftrightarrow B), \Delta}(\mathbb{G} \leftrightarrow d)$$

$$\frac{A[x \leftarrow y], \Gamma \Rightarrow \Delta}{(\exists x\ A), \Gamma \Rightarrow \Delta}(\mathbb{G} \exists g) \qquad \frac{\Gamma \Rightarrow A[x \leftarrow t], (\exists x\ A), \Delta}{\Gamma \Rightarrow (\exists x\ A), \Delta}(\mathbb{G} \exists d)$$

$$\frac{(\forall x\ A), A[x \leftarrow t], \Gamma \Rightarrow \Delta}{(\forall x\ A), \Gamma \Rightarrow \Delta}(\mathbb{G} \forall g) \qquad \frac{\Gamma \Rightarrow A[x \leftarrow y], \Delta}{\Gamma \Rightarrow (\forall x\ A), \Delta}(\mathbb{G} \forall d)$$

Dans les règles pour les quantificateurs, la variable x est une variable quelconque et y est une variable telle que $y \notin \mathcal{VL}(A) \setminus \{x\}$. Le *terme t est libre pour la variable x dans*

la formule A (c'est-à-dire que les variables libres du terme ne sont pas quantifiées une fois la substitution réalisée). Dans les règles $\mathbb{G}\exists g$ et $\mathbb{G}\forall d$, la variable y n'apparaît pas libre dans le séquent conclusion.
L'ensemble des axiomes du système $\mathbb{G}$ est défini comme étant l'ensemble des séquents $(\Gamma \Rightarrow \Delta)$ tels que Γ et Δ ont des formules en commun (un séquent qui contient $\top$ dans sa partie droite ou $\bot$ dans sa partie gauche est aussi un axiome).
La *règle de la coupure* étend les règles du système $\mathbb{G}$ par la construction et l'utilisation de lemmes ($\Gamma, \Omega, \Delta, \Lambda$ quatre ensembles de formules et L une formule) :

$$\frac{\Omega \Rightarrow L, \Lambda \qquad \Gamma, L \Rightarrow \Delta}{\Omega, \Gamma \Rightarrow \Lambda, \Delta}$$

Le lemme L est isolé et démontré une unique fois sur une partie Ω des hypothèses ; le lemme est alors utilisable à loisir dans la preuve de Δ. Tout théorème (c'est-à-dire séquent valide) démontré grâce à la règle de la coupure peut l'être sans celle-ci ; le coût par rapport au nombre d'applications des règles peut être alors exponentiel. Puisque ce sont des ensembles de formules qui sont manipulés, de cette règle peuvent s'extraire deux restrictions plus intuitives :

$$\frac{\Omega \Rightarrow L \qquad \Gamma, L \Rightarrow \Delta}{\Omega, \Gamma \Rightarrow \Delta} \qquad \frac{\Gamma \Rightarrow L \qquad \Gamma, L \Rightarrow \Delta}{\Gamma \Rightarrow \Delta}$$

La première illustre, l'activité de recherche dans un domaine formel : pour établir un nouveau théorème, il n'est pas nécessaire de tout redémontrer et la preuve de L est considérée comme acquise ; la seconde est plus proche de l'*apprentissage de lemme* : un lemme est calculé pour épargner une partie du parcours de l'espace de recherche.

CCDNF. cf. `IQTest`.

certificat. cf. § 1.5.3.

CHR. [85] Le langage CHR (pour « Constraints Handling Rules ») est un langage de règles de réécriture destiné à l'implantation de solveurs de contraintes dans un langage hôte. Le langage CHR permet à l'utilisateur de définir déclarativement des contraintes, le langage hôte réalisant le reste des calculs. Le langage CHR est un langage de règles gardées à gardes multiples dont la syntaxe (simplifiée) d'une règle est la suivante :

– une règle de simplification est de la forme :

$$H_1, \ldots, H_i <=> G_1, \ldots, G_j | B_1, \ldots, B_k$$

– ou bien une règle de propagation est de la forme :

$$H_1, \ldots, H_i ==> G_1, \ldots, G_j | B_1, \ldots, B_k$$

avec $i > 0$, $j \geq 0$, $k \geq 0$, $H_1, \ldots, H_i$ une suite non vide de contraintes CHR, $G_1, \ldots, G_j$ une suite de contraintes gérées par l'hôte et $B_1, \ldots, B_k$ une suite de contraintes. La sémantique déclarative est la suivante :

– pour une règle de simplification, c'est l'équivalence

$$\forall X((G_1 \wedge \ldots \wedge G_j) \rightarrow ((H_1 \wedge \ldots \wedge H_i) \leftrightarrow (\exists Z\ (B_1 \wedge \ldots \wedge B_k))))$$

– pour une règle de propagation, c'est l'implication

$$\forall X((G_1 \wedge \ldots \wedge G_j) \rightarrow ((H_1 \wedge \ldots \wedge H_i) \rightarrow (\exists Z\ (B_1 \wedge \ldots \wedge B_k))))$$

Plus informellement, la règle de simplification remplace, si les gardes $G_1 \wedge \ldots \wedge G_j$ sont vérifiées, les contraintes de la tête $H_1, \ldots, H_i$ par celles du corps $B_1, \ldots, B_k$ dans l'ensemble des contraintes tandis que la règle de propagation ajoute les contraintes du corps tout en conservant celles de la tête. Les règles sont appliquées jusqu'à ce qu'un point fixe soit atteint.
Le formalisme est illustré ci-dessous sur les *règles* du chapitre 4 avec pour langage hôte *Prolog* ; le *prédicat* CHR pour les règles de l'implication est `'qbf>'`.
Un *schéma* contradictoire d'instance $Q((x \rightarrow y) \leftrightarrow z)$ est tel que $Q((x \rightarrow y) \leftrightarrow z) \equiv \bot$; une règle CHR de simplification en est déduite :

```
'qbf>'(LX, LY, LZ) <=> condition(LX,LY,LZ) | fail.
```

Le prédicat `fail` du langage hôte échoue systématiquement ; le prédicat `condition` assure la garde sur le lieur et les égalités possibles des *littéraux* x, y et z entre-eux ou avec les constantes propositionnelles qui sont codées `0` pour $\bot$ et `1` pour $\top$. Par exemple, le schéma $\forall x \forall y((x \rightarrow y) \leftrightarrow \overline{y})$ est contradictoire, la règle CHR en est déduite :

```
'qbf>'(LX, LY, LZ) <=>
  'u?'(LX), 'u?'(LY), (LX < LY), lit_opp(LY,LZ) | fail.
```

Le prédicat `<` réalise l'ordre « $\prec$ » du lieur ; le prédicat `'e?'` (resp. `'u?'`) est tel que `'e?'(L)` (resp. `'u?'(L)`) est vrai si le littéral `L` est quantifié existentiellement (resp. universellement) ; le prédicat `lit_opp` est tel que `lit_opp(L,L_)` est vrai si le littéral `L` est le *complémentaire* du littéral `L_`.
Un schéma tautologique d'instance $Q((x \rightarrow y) \leftrightarrow z)$ est tel que $(Q((x \rightarrow y) \leftrightarrow z) \leftrightarrow \top)$; une règle CHR de simplification en est déduite :

```
'qbf>'(LX, LY, LZ) <=>  condition(LX,LY,LZ) | true.
```

Le prédicat `true` s'efface avec succès systématiquement.
Les règles de propagation/simplification s'implantent en des règles CHR de simplification. Par exemple, la règle pour le schéma $\exists|y|\forall|z|\exists|x|((x \rightarrow y) \leftrightarrow \overline{x})$ et la propagation $[y \leftarrow \bot][x \leftarrow \overline{z}]$ s'implante en la règle CHR de simplification :

```
'qbf>'(LX, LY, LZ) <=>
  'e?'(LX), 'e?'(LY), 'u?'(LZ), (LY < LZ), (LZ < LX) |
  LY <- 0, LY <- ~(LZ).
```

Le prédicat `'<-'` réalise la substitution sur les littéraux.
Les règles de propagation s'implantent en des règles CHR de propagation. Par exemple, la règle de propagation pour le schéma $\exists|z|\forall|x|\exists|y|((x \rightarrow y) \leftrightarrow z)$ s'implante en la règle CHR de propagation :

```
'qbf>'(LX, LY, LZ) ==>
  'u?'(LX), 'e?'(LY), 'e?'(LZ), (LZ < LX), (LX < LY) | LZ <- 1.
```

Les littéraux sont obtenus grâce à des variables Prolog attribuées [114] qui permettent d'attacher des attributs aux variables. Ces derniers sont des informations (par exemple un domaine) ou des comportements et sont particulièrement utiles pour implanter des systèmes pour des *problèmes de satisfaction de contraintes*. L'élimination des quantificateurs (de la *logique des prédicats* implicitement existentiels) entre deux phases de propagation (c'est-à-dire deux calculs de point fixe) s'effectue dans le langage hôte Prolog par des prédicats définis par l'utilisateur qui énumèrent les valeurs pour les variables associées aux quantificateurs (via le mécanisme de *retour-arrière* de Prolog).

`CirQit.` [112] `CirQit` est une *procédure de décision* pour le *problème de validité* des QBF *prénexes* dont la *matrice* est quelconque (sur le fragment $\{\wedge, \vee, \neg\}$) basée sur `QDLL` par extension d'une procédure de décision pour le problème *SAT* [222]. En interne, la matrice est sous la forme d'un circuit logique (avec partage de structures), c'est-à-dire un *graphe acyclique orienté* avec nœuds n-aires.

`CirQit`		
Décision	*PSPACE*	
Sémantique	Polynomial en espace	QBF prénexe
	Monolithique	
	DAG	
	∃-learning	
	∀-learning	
	Réduction universelle	
	Simplification $\top$ et $\bot$	
	Propagation don't care	

clause. **1.** PROP & QBF. Une *clause* est une *formule propositionnelle* uniquement formée de *disjonctions* et de *littéraux*. Une *clause tautologique* est une clause qui contient un littéral et son *complémentaire*. Une clause tautologique est équivalente, par les *équivalences logiques* 0.10 et 0.20, à $\top$. Une clause est *unitaire* si elle est réduite à un seul littéral. (En QBF, une clause peut être considérée comme unitaire si elle ne contient qu'un seul littéral quantifié existentiellement et des littéraux universellement quantifiés tels que ceux-ci soient plus internes que le littéral « unitaire » existentiellement quantifié.) Une *clause vide* est une clause ne contenant aucun littéral et représente donc, par l'équivalence logique 0.15, $\bot$ (l'élément neutre de la disjonction). Une *clause définie* est une clause contenant exactement un *littéral positif*. Une *clause de Horn* est une clause contenant au plus un littéral positif. Une clause est *subsummée* par une autre clause si tout littéral de la seconde apparaît dans la première (par conséquent, tout *modèle* de la seconde est modèle de la première). **2.** PRED. Une *clause* est une *formule de la logique des prédicats prénexe* uniquement formée de disjonctions et de *littéraux universellement quantifiés*.

clause de Horn. cf. *clause*.

clause définie. cf. *clause*.

clause tautologique. cf. *clause*.

clause unitaire. cf. *clause*.

clause vide. cf. *clause*.

CLearn . [97] `CLearn` est une *procédure de décision* pour le *problème de validité* des QBF *prénexes* dont la *matrice* est sous *FNC* basée sur `QDLL`. La procédure intègre la *réduction universelle* suivie d'une *propagation des clauses unitaires* uniquement sur la clause menant à la propagation ; elle intègre aussi du *conflict learning* associé à du *conflict directed backjumping*. La procédure se focalise sur l'expérimentation de deux techniques dites « complètes » de *solution learning* (l'une locale et l'autre globale, extension de la première en en considérant les lemmes persistants pour le reste de la recherche) par opposition aux techniques précédemment proposées [130, 104, 230] qui sont qualifiées d'« incomplètes » car n'apprenant ni toutes les clauses ni tous les *cubes* possibles. Ces deux techniques introduisent dans la formule initiale des symboles propositionnels *universellement quantifiés* pour construire des cubes qui capturent l'ensemble des lemmes qui peuvent être appris sur les universelles. Ces techniques dont l'efficacité est modeste semblent complexes à mettre en œuvre.

`CLearn`

Décision	*PSPACE*	
Sémantique	Polynomial en espace	FNCA prénexe
	Monolithique	
	FNC prénexe	
	∃-learning	
	∀-learning	
	Propagation unitaire	
	Littéraux monotones	

clos. **1.** QBF. cf. définition 4. **2.** PRED. Une formule est *close* si toute occurrence de variable dans la formule est sous la portée d'un quantificateur associé à cette même variable. **3.** Un fragment d'un *langage* est *clos* pour une transformation (une fonction) si une fois appliquée la transformation sur un *mot* du langage, le nouveau mot appartient aussi au langage. Le fragment est *linéairement* (resp. *polynomialement*) *clos* par la transformation si le mot résultant de la transformation est linéaire (resp. polynomial) en taille par rapport au mot en entrée de la transformation.

clôture existentielle. cf. § 1.2.2.

clôture universelle. cf. § 1.2.2.

CNF. cf. *forme normale*.

combined conjunctive-disjunctive normal form. cf. `IQTest`.

complémentaire. **1.** Morph. cf. définition 3. **2.** Langage. cf. *machine de Turing*.

complet, **completude.** **1.** Un système de construction de preuves est *complet* vis-à-vis d'une sémantique lorsque toute formule qui appartient à la sémantique est prouvable dans le système. **2.** cf. *hiérarchie polynomiale*.

composante fonctionnelle. cf. définition 18.

composante propositionnelle. cf. définition 18.

conflict directed backjumping. cf. *retour-arrière*.

conflict learning. cf. *apprentissage de lemme*.

conjonction. cf. définition 1.

conjugué. cf. définition 3.

consistance d'arc quantifié. cf. *problème de satisfaction de constraintes quantifiées.*

contrainte. **1.** QBF. cf. définition 42. **2.** cf. *problème de satisfaction de contraintes.*. **3.** cf. *programme logique normal.*

contrainte quantifiée. cf. définition 42.

correct, **correction.** Un système de construction de preuves est *correct* vis-à-vis d'une sémantique lorsque toute formule prouvable dans le système appartient à la sémantique.

cube. Un *cube* (ou *terme*) est une *conjonction* de *littéraux*. Un *cube vide* est un cube ne contenant aucun littéral et représente donc, par l'*équivalence logique* (0.13), $\top$ (l'élément neutre de la conjonction).

cube vide. cf. *cube.*

DAG. cf. *graphe acyclique orienté.*

Davis-Logemann-Loveland. **1.** QBF. cf. § 1.5.1. **2.** PROP. [68] La procédure de *Davis-Logemann-Loveland* (ou *DLL*) est une *procédure de décision* pour le problème *SAT* qui parcourt en profondeur d'abord un *arbre sémantique* avec à la racine une formule propositionnelle sous *FNC*. La *substitution* d'un symbole propositionnel par $\top$ dans une formule sous FNC élimine toutes les *clauses* le contenant et élimine sa négation de toutes les clauses ; par dualité, la substitution d'un symbole propositionnel par $\bot$ dans une formule sous FNC élimine toutes les clauses contenant sa négation et l'élimine de toutes les clauses. Les cas d'arrêt de l'induction sont : soit un nœud associé à une formule sous FNC vide est parcouru et l'algorithme retourne $\top$ (la formule est *satisfiable* et le *modèle* est extrait de l'ensemble des arcs entre la racine et ce nœud) ; soit un nœud associé à une formule contenant une *clause vide* est parcouru et l'algorithme retourne $\bot$ (ce nœud est une *feuille* de l'arbre sémantique, cela correspond à une *valuation* qui rend la formule *insatisfiable* puisqu'elle contient une clause vide qui est équivalente à $\bot$, l'élément neutre de la *disjonction*). Le cas d'induction est le suivant : un symbole propositionnel est substitué dans la QBF par $\top$, si l'appel récursif retourne $\top$ alors $\top$ est retourné par contre si l'appel récursif retourne $\bot$ alors le résultat de l'appel récursif avec le symbole propositionnel substitué par $\bot$ est retourné. L'algorithme de *Davis-Logemann-Loveland* intègre la *propagation de clauses unitaires* et la *propagation de littéraux monotones*. Les deux sources d'indéterminisme dans un arbre sémantique sont le choix du symbole propositionnel pour la paire d'arcs sortants d'un noeud et l'ordre de ces deux arcs ; ces deux choix sont libres.

Davis-Putnam. **1.** QBF cf. § 1.5.2. **2.** PROP. La procédure de *Davis-Putnam* [69] (ou *DP*) est une *procédure de décision* pour le problème *SAT* fonctionnant par induction sur la *multi-résolution* qui applique toutes les *résolutions* possibles sur un symbole propositionnel (en éliminant ainsi toutes les occurrences du symbole propositionnel et donc implicitement le quantificateur, cette technique peut être vue

comme une élimination de quantificateurs (cf. § 1.5)). La multi-résolution n'est pas *linéairement close* en espace mais seulement *polynomialement close* en espace sur le fragment *FNC* [83]. Si une *clause vide* fait partie de la formule alors cette dernière est *insatisfiable*. Si l'ensemble de clauses qui forme la formule devient vide alors la formule est *satisfiable*. Le système `ZRES` [50] implémente l'algorithme de Davis-Putnam sur le formalisme des *ZBDD* [148].

`decide` [187] `decide` est une *procédure de décision* pour le problème de validité des QBF *prénexes* dont la *matrice* est sous *FNC* basée sur `QDLL`. Ce n'est pas explicite, mais la procédure `decide` intègre la *propagation des littéraux monotones*. Cette procédure intègre aussi les techniques suivantes : l'*inversion des quantificateurs*, l'*échantillonnage* et la *détection des littéraux en échec*. La procédure de décision `decide` a été implantée à partir de la procédure de décision `sat0` [133] pour le problème *SAT*.

`decide`

Décision	*PSPACE*	
Sémantique	Polynomial en espace	FNC prénexe
	Monolithique	
	FNC prénexe ∃-backtrack ∀-backtrack Propagation unitaire Littéraux monotones	

décidable, **décider.** cf. *machine de Turing*.

décomposition. cf. § 4.1.

dépendance. Une *dépendance* entre symboles propositionnels pour une QBF existe lorsque l'ordre d'*élimination des quantificateurs* ne peut être inversé sans perdre la correction. Une dépendance bride le choix d'élimination des quantificateurs sur lequel reposent les *heuristiques* efficaces pour le problème *SAT*. La relation d'ordre $\prec$ induite par une QBF est une relation de dépendance qui n'est pas nécessairement minimale en ce qu'elle contient des couples qui ne sont pas des dépendances. Le calcul de la dépendance minimale, qui est basée pour le cas des QBF sous *FNC* sur le graphe des connections des symboles propositionnels via la participation aux *clauses*, est *PSPACE-complet* [191]. Le *miniscoping* [17] qui est basé sur les *équivalences logiques* $((qx\ F) \vee G) \overset{1.5}{\equiv} (qx\ (F \vee G))$ et $((qx\ F) \wedge G) \overset{1.6}{\equiv} (qx\ (F \wedge G))$ (x n'apparaissant pas *libre* dans G) permet de réduire les dépendances. La *mise sous forme prénexe* introduit des dépendances par la mise sous forme linéaire en un lieur. À partir d'une forme prénexe peuvent être extraits un *arbre de quantificateurs* [26] ou une *structure quantifiée* [109] qui sont des sur-approximations de la relation de dépendance minimale.

dependency-directed backtracking. cf. *retour-arrière*.

détection des litteraux en échecs. [187] La *détection des littéraux en échec* est une technique pour les *procédures de décision* pour le *problème de validité* avec pour entrée des QBF *prénexes* sous FNC qui est issue de la technique de même nom pour

le problème SAT [133]. Un *quantificateur le plus interne* d'une QBF prénexe $QqxF$ est sélectionné et une *propagation des clauses unitaires* sur $[x \leftarrow \top](F)$ est opérée ; si la *clause vide* est présente dans le résultat alors si le quantificateur q est existentiel, $\neg x$ est ajouté à l'ensemble des clauses sinon la QBF est n'est pas *valide*. Le calcul peut être aussi réalisé avec $[x \leftarrow \bot](F)$ mais alors c'est x qui est ajouté à l'ensemble des clauses.

diagramme de décision binaire. [44] (en. *Binary Decision Diagram* ou *BDD*) Un *diagramme de décision binaire* est un graphe binaire orienté acyclique dont les nœuds sont étiquetés par des *symboles propositionnels*, les arcs étiquetés par des *valeurs de vérité* et deux nœuds sans arc sortant **0** et **1**. Un BDD est aussi vu comme une *formule propositionnelle* sous FNN dont les nœuds $\wedge$ sont des *nœuds d'assignation* et les nœuds $\vee$ sont des *nœuds de décision* [83]. Un *nœud d'assignation* est un nœud $\wedge$ de la forme $(x \wedge F)$ (resp. $(\neg x \wedge F)$) avec x un symbole propositionnel et F une formule. Un *nœud de décision* est un nœud $\vee$ de la forme $((\neg x \wedge F) \vee (x \wedge G))$ (resp. $(\neg x \wedge F)$) avec x un symbole propositionnel et F, G des formules. Un *OBDD* (en. *Ordered Binary Decision Diagram*) est un BDD dont l'ordre des symboles propositionnels dans chaque chemin dans le graphe respecte un ordre sur les symboles propositionnels. Un OBDD est *réduit* si chaque nœud représente une *fonction booléenne* distincte (c'est-à-dire il n'y a pas de sous-graphes isomorphes). Les OBDD réduits sont une représentation canonique des fonctions booléennes. Les *ZBDD* [148] (pour « Zero suppressed BDD ») sont une extension symbolique des BDD qui permettent de représenter de manière compact des ensembles de *clauses* ; la sémantique des arcs est modifiée en « présent/absent ». Si les BDD sont bien adaptés à la représentation de fonctions booléennes, les ZBDD sont bien adaptés à la représentation d'ensembles de sous-ensembles.

directed acyclic graph. cf. *graphe acyclique orienté.*

disjonction. cf. définition 1.

distributivité. cf. *équivalence logique.*

DLL cf. *Davis-Logemann-Loveland.*

DNF. cf. *forme normale.*

DP. cf. *Davis-Putnam.*

`Duaffle`. [190] `Duaffle` est une *procédure de décision* pour le *problème de validité* des QBF *prénexes* dont la *matrice* est sous la forme soit $(R_\exists \wedge (R_\forall \rightarrow V))$, soit $(R_\forall \rightarrow (R_\exists \wedge V))$. La QBF est interprétée comme codant un jeu à deux joueurs avec les règles $(R_\exists)$ et conditions de victoire (V) du joueur existentiel et les règles $(R_\forall)$ du joueur universel ; $R_\exists$, $R_\forall$ et V sous *FNC*. La procédure `Duaffle` est basée sur la procédure `Quaffle`. Par les *équivalences logiques* (0.1), (0.5) et (0.6), les règles du joueur universel sont converties sous *FND*. Le premier schéma correspond au fait que le joueur existentiel à tout de même besoin de réaliser la solution pour gagner tandis que le second schéma correspond au fait qu'il suffit que le joueur universel triche pour que le joueur existentiel gagne.

Duaffle

Décision	$PSPACE$	
Sémantique	Polynomial en espace	FNC/FND prénexe
	Monolithique	
	FNC/FND prénexe $\exists$-backtrack $\forall$-backtrack Propagation unitaire Littéraux monotones	

dual de la Q-résolution. cf. *Q-résolution.*

EBDDRES [203] Le système EBDDRES est une extension de la *procédure de décision* de même nom pour le problème SAT [119] étendue aux QBF qui génère, pour une QBF qui n'est pas *valide*, une preuve par *résolution* et qui construit, pour une QBF *valide*, le modèle QBF sous la forme d'un *BDD*.

échantillonnage. [187] L'*échantillonnage* (en. *sampling*) est une technique pour les *procédures de décision* basées sur une $\forall$-*expansion top-down*, de diminution de l'espace de recherche pour des QBF sous FNC de la forme $\exists x_1 \ldots \exists x_n \forall y_1 \ldots \forall y_p F$. Pour chaque x_i, $1 \leq i \leq n$ la boucle suivante est répétée un certain nombre de fois : pour des valeurs de $C_1, \ldots, C_p \in \{\top, \bot\}$ prises au hasard, $[x_i \leftarrow \top][y_1 \leftarrow C_1] \ldots [y_p \leftarrow C_p](F)$ subit une *propagation des clauses unitaires*, si la *clause vide* est présente dans le résultat alors $\neg x_i$ est ajouté à l'ensemble des *clauses* (et une nouvelle propagation des clauses unitaires peut s'en suivre). De la même manière, le calcul peut-être effectué avec la substitution $[x_i \leftarrow \bot]$.

élimination de quantificateurs. cf. § 1.5.

élimination d'un littéral unitaire globalement. cf. *propagation de clauses unitaires.*

élimination d'un littéral unitaire localement. cf. *propagation de clauses unitai res.*

ensemble des sous-formules et leurs complémentaires. cf. définition 3.

ensemble étendu de sous-formules. cf. définition 3.

équivalence logique. **1.** QBF. cf. définition 23 pour l'*équivalence qui préserve les modèles propositionnels* et définition 24 pour l'*équivalence qui préserve les modèles QBF*. **2.** PROP. L'*équivalence logique* (notée $(. \equiv .)$) est une *relation d'équivalence* définie ainsi : soient F et G deux formules, $F \equiv G$ si $\models (F \leftrightarrow G)$. De plus cette relation d'équivalence est une *congruence* : soient F, F', G et G' des formules, si $F \equiv F'$ et $G \equiv G'$ alors $\neg F \equiv \neg F'$, $(F \vee G) \equiv (F' \vee G')$, $(F \rightarrow G) \equiv (F' \rightarrow G')$, $(F \leftrightarrow G) \equiv (F' \leftrightarrow G')$ et $(F \wedge G) \equiv (F' \wedge G')$. Une propriété particulièrement souhaitable issue de la congruence est la *substitution* des équivalents : soit x un symbole propositionnel, F, G et H trois formules telles que $G \equiv H$ alors $[x \leftarrow G](F) \equiv [x \leftarrow H](F)$.
Les équivalences logiques classiques sont :

0.1 $(F{\rightarrow}G) \equiv (\neg F{\vee}G)$ [$\rightarrow$ en fonction du $\vee$ et de la $\neg$]
0.2 $(F{\rightarrow}G) \equiv \neg(F{\wedge}\neg G)$ [$\rightarrow$ en fonction du $\wedge$ et de la $\neg$]
0.3 $(F{\leftrightarrow}G) \equiv ((F{\rightarrow}G){\wedge}(G{\rightarrow}F))$ [$\leftrightarrow$ en fonction du $\wedge$ et de l'$\rightarrow$]
0.4 $\neg\neg F \equiv F$ [négation involutive]
0.5 $\neg(F{\vee}G) \equiv (\neg F{\wedge}\neg G)$ [loi de de Morgan]
0.6 $\neg(F{\wedge}G) \equiv (\neg F{\vee}\neg G)$ [loi de de Morgan]
0.7 $\neg\top \equiv \bot$ [$\bot$ en fonction de $\neg$ et $\top$]
0.8 $\neg\bot \equiv \top$ [$\top$ en fonction de $\neg$ et $\bot$]
0.9 $(\neg F{\wedge}F) \equiv \bot$ [$\bot$ en fonction de $\neg$ et $\wedge$]
0.10 $(\neg F{\vee}F) \equiv \top$ [$\top$ en fonction de $\neg$ et $\vee$]
0.11 $(F{\wedge}F) \equiv F$ [idempotence du $\wedge$]
0.12 $(F{\vee}F) \equiv F$ [idempotence du $\vee$]
0.13 $(\top{\wedge}F) \equiv F$ [$\top$ élément neutre de $\wedge$]
0.14 $(\top{\vee}F) \equiv \top$ [$\top$ élément absorbant de $\vee$]
0.15 $(\bot{\vee}F) \equiv F$ [$\bot$ élément neutre de $\vee$]
0.16 $(\bot{\wedge}F) \equiv \bot$ [$\bot$ élément absorbant de $\wedge$]
0.17 $(F{\wedge}(F{\vee}G)) \equiv F$ [Loi d'absorption]
0.18 $(F{\vee}(F{\wedge}G)) \equiv F$ [Loi d'absorption]
0.19 $(F{\wedge}G) \equiv (G{\wedge}F)$ [Commutativité du $\wedge$]
0.20 $(F{\vee}G) \equiv (G{\vee}F)$ [Commutativité du $\vee$]
0.21 $((F{\wedge}G){\wedge}H) \equiv (F{\wedge}(G{\wedge}H))$ [Associativité du $\wedge$]
0.22 $((F{\vee}G){\vee}H) \equiv (F{\vee}(G{\vee}H))$ [Associativité du $\vee$]
0.23 $(F{\wedge}(G{\vee}H)) \equiv ((F{\wedge}G){\vee}(F{\wedge}H))$ [Distributivité du $\wedge$ / $\vee$]
0.24 $(F{\vee}(G{\wedge}H)) \equiv ((F{\vee}G){\wedge}(F{\vee}H))$ [Distributivité du $\vee$ / $\wedge$]
0.25 $(F{\rightarrow}G) \equiv (\neg G{\rightarrow}\neg F)$ [Contraposée]
0.26 $(x{\wedge}F) \equiv (x{\wedge}[x \leftarrow \top](F))$ [Propagation du littéral pour $\wedge$]
0.27 $(\neg x{\wedge}F) \equiv (\neg x{\wedge}[x \leftarrow \bot](F))$ [Propagation du littéral pour $\wedge$]
0.28 $(x{\vee}F) \equiv (x{\vee}[x \leftarrow \bot](F))$ [Propagation du littéral pour $\vee$]
0.29 $(\neg x{\vee}F) \equiv (\neg x{\vee}[x \leftarrow \top](F))$ [Propagation du littéral pour $\vee$]
0.30 $(F{\oplus}G) \equiv ((F{\vee}G){\wedge}\neg(F{\wedge}G))$

3. PRED. La définition est identique à celle, ci-dessus, de **1. PROP** avec pour équivalences classiques (x n'apparaissant pas *libre* dans G) :

1.1 $(\forall x\ (\forall y\ F)) \equiv (\forall y\ (\forall x\ F))$
1.2 $(\exists x\ (\exists y\ F)) \equiv (\exists y\ (\exists x\ F))$
1.3 $\neg(\forall x\ F) \equiv (\exists x\ \neg F)$
1.4 $\neg(\exists x\ F) \equiv (\forall x\ \neg F)$
1.5 $((qx\ F){\vee}G) \equiv (qx\ (F{\vee}G))$
1.6 $((qx\ F){\wedge}G) \equiv (qx\ (F{\wedge}G))$
1.7 $((\forall x\ F){\wedge}(\forall x\ H)) \equiv (\forall x\ (F{\wedge}H))$
1.8 $((\exists x\ F){\vee}(\exists x\ H)) \equiv (\exists x\ (F{\vee}H))$
1.9 $((\exists x\ F){\rightarrow}G) \equiv (\forall x\ (F{\rightarrow}G))$
1.10 $((\forall x\ F){\rightarrow}G) \equiv (\exists x\ (F{\rightarrow}G))$
1.11 $(G{\rightarrow}(\exists x\ F)) \equiv (\exists x\ (G{\rightarrow}F))$
1.13 $(G{\rightarrow}(\forall x\ F)) \equiv (\forall x\ (G{\rightarrow}F))$

1.14 $((\forall x\ F) \rightarrow (\exists x\ H)) \equiv (\exists x\ (F \rightarrow H))$

equivalence reasoning. cf. *raisonnement sur équivalence.*

Evaluate. [47, 48, 46] `Evaluate` est une *procédure de décision* pour le *problème de validité* des QBF *prénexes* dont la *matrice* est sous *FNC* basée sur un `QDLL`. L'implantation de la procédure de décision `Evaluate` fait date puisqu'elle marque la première implantation décrite et testée [46] pour le problème de validité des QBF. Le cas d'arrêt considère une formule ne comptant plus que des symboles propositionnels existentiellement quantifiés et fait appel alors à une procédure de décision pour le problème SAT. Le cas d'induction du `QDLL` est précédé d'un test de *validité universelle triviale* puis d'une phase d'application jusqu'à obtention d'un point fixe d'une simplification nommée *forced assignement* et de la *propagation de littéraux monotones* comme dans `QDLL` mais aussi de la *falsifiabilité existentielle triviale*, de la *falsifiabilité universelle triviale* et de la *falsifiabilité par littéraux complémentaires*. La simplification *forced assignement* correspond implicitement à la *réduction universelle* suivie d'une *propagation des clauses unitaires* uniquement sur la *clause* menant à la propagation.

`Evaluate`

Décision	$PSPACE$	
Sémantique	Polynomial en espace	FNC prénexe
	Monolithique	
	FNC prénexe ∃-backtrack ∀-backtrack Propagation unitaire Littéraux monotones	

existentiel. **1.** Morph. cf. définition 1. **2.** Sem. définitions 15 et 20.

existentiellement quantifié. cf. définition 4.

expansion. cf. § 1.5.

failed literal detection. cf. *détection des littéraux en échec.*

falsifiabilité. cf. *sémantique de la logique propositionnelle.*

falsifiabilité existentielle triviale. (en. *Trivial falsity on* Σ) [48] Une QBF sous FNC dont l'ensemble des *clauses* ne contenant uniquement que des *littéraux existentiellement quantifiées* est *insatisfiable*, n'est pas *valide.*

falsifiabilité par littéraux complementaires. (en. *pairwise falsity*) [48] La technique de *falsifiabilité par littéraux complémentaires* reprend un cas particulier de la *Q-résolution* (pour des QBF *prénexes* sous FNC). Si la *matrice* de la QBF contient deux *clauses* telles qu'elles ne contiennent chacune qu'un seul *littéral existentiellement quantifié*, que ces deux littéraux sont *complémentaires* et que les autres littéraux sont des *littéraux universellement quantifiés* et si aucun littéral universellement quantifié qui précède selon l'ordre $\prec$ le littéral existentiellement quantifié n'a son complémentaire

dans l'autre clause alors la QBF n'est pas *valide*. Cette technique correspond à la recherche de la *falsifiabilité universelle triviale* après une étape de Q-résolution (sans la réaliser effectivement).

falsifiabilité universelle triviale. (en. *Trivial falsity on* Π) [48] Une QBF sous FNC dont une des *clauses* qui n'est pas *tautologique* n'est constituée que de *littéraux universellement quantifiés* n'est pas *valide*.

falsifier. cf. *sémantique de la logique propositionnelle*.

faux. cf. définition 11.

FNC. cf. *forme normale*.

FNCA. cf. *apprentissage de lemme*.

FND. cf. *forme normale*.

FNN. cf. *forme normale*.

FNND. cf. *forme normale*.

fonction booléenne. cf. définition 11.

fonction d'interprétation. **1.** QBF. cf. définition 31. **2.** PRED. cf. *sémantique de la logique des prédicats*.

fonction de substitution. cf. définition 9.

forced assignement. cf. `Evaluate`.

forme normale. **1.** QBF. cf. définition 8. **2.** PROP. Une *formule propositionnelle* est en *forme normale négative* (ou *FNN*, en. *Negation Normal Form* ou *NNF*) si c'est une formule uniquement constituée de *conjonctions*, de *disjonctions* et de *littéraux* (dans certains cas les constantes $\top$ et $\bot$ sont autorisées). Une autre définition [67] orientée théorie des graphes et qui fait apparaître les sous-formules partagées est la suivante : un graphe orienté acyclique disposant d'une racine et tel que chaque nœud feuille est étiqueté par $\top$, $\bot$ ou un littéral et chaque nœud interne, possédant un nombre quelconque de nœuds fils, est étiqueté par une conjonction ou une disjonction.

Toute formule propositionnelle peut-être *mise sous forme normale négative* en appliquant les *équivalences logiques* : $(A \oplus B) \equiv \neg(A \leftrightarrow B)$, $(A \rightarrow B) \overset{0.1}{\equiv} (\neg A \vee B)$, $(A \leftrightarrow B) \overset{0.3}{\equiv} ((A \rightarrow B) \wedge (B \rightarrow A))$ puis les *lois de de Morgan* et l'involution de la négation : $\neg\neg A \overset{0.4}{\equiv} A$. Cette transformation qui préserve la validité est polynomiale.

Une formule est sous *forme normale conjonctive* (ou *FNC*, en. *Conjunctive Normal Form* ou *CNF*) si c'est « une » conjonction de disjonctions de littéraux (soit encore « une » conjonction de *clauses*). Toute formule sous FNC est aussi une formule sous FNN. Toute formule propositionnelle peut-être *mise en forme normale conjonctive* en appliquant d'abord la mise sous FNN puis la distributivité : $(F \vee (G \wedge H)) \overset{0.24}{\equiv} ((F \vee G) \wedge (F \vee H))$. Cette transformation qui préserve la validité est dans le pire des cas exponentielle [43]. Une formule est sous *forme normale disjonctive* (ou *FND*, en. *Disjunctive Normal Form* ou *DNF* ou *Sum Of Products* ou *SOP*) si c'est « une » disjonction de conjonctions de littéraux. Toute formule propositionnelle peut-être *mise en forme normale disjonctive* en appliquant d'abord la mise sous FNN puis la distributivité : $(F \wedge (G \vee H)) \overset{0.23}{\equiv} ((F \wedge G) \vee (F \wedge H))$.

De part la commutativité et l'associativité de la conjonction et de la disjonction la FNC peut être vue comme un ensemble (« une » conjonction) d'ensembles (des dis-

jonctions) ou *clauses* de littéraux ; de même la FND est alors un ensemble (« une » disjonction) d'ensembles (des conjonctions) de littéraux ou *cubes*. Un ensemble vide considéré comme une conjonctive représente $\top$ (l'élément neutre de la conjonction, cf. l'équivalence logique 0.13) tandis qu'un ensemble vide considéré comme une disjonction représente $\bot$ (l'élément neutre de la disjonction, cf. l'équivalence logique 0.15).

Une formule sous FNN est *décomposable* [65, 66, 67] si pour toute *conjonction* $(F \wedge G)$, F et G ne partageant pas de symboles propositionnels. *Décider* si une formule sous *forme normale négative décomposable* (*FNND*, en. *Decomposable Negative Normal Form*, en. *DNNF*) est *satisfiable* peut être réaliser en temps linéaire [65].

3. PRED. Une formule de la *logique des prédicats* est sous *forme normale négative* (resp. *conjonctive*, *disjonctive*) si c'est une formule sous forme *prénexe* dont le *lieur* est uniquement constitué de quantificateurs universels et dont la *matrice* est uniquement constituée de *conjonctions*, de *disjonctions* et de *littéraux*. Les mêmes restrictions sur la matrice que pour le cas **1. PROP.** permettent de définir la *forme normale conjonctive* et la *forme normale disjonctive*.

forme normale conjonctive. **1. QBF.** cf. définition 8. **2. PROP&PRED.** cf. *forme normale*.

forme normale de contraintes. cf. définition 42.

forme normale disjonctive. **1. QBF.** cf. définition 8. **2. PROP&PRED.** cf. *forme normale*.

forme normale négative. **1. QBF.** cf. définition 8. **2. PROP&PRED.** cf. *forme normale*.

forme normale négative décomposable. cf. *forme normale*.

formule associée à une fonction. cf. définition 17.

formule booléenne quantifiée. cf. définition 2.

formule booléenne quantifiée prénexe. cf. définition 5.

formule de la logique des prédicats cf. *langage de la logique des prédicats*.

formule propositionnelle cf. *langage de la logique propositionnelle*.

formule propositionnelle associée à une substitution. cf. définition 9.

garde. cf. définition 34.

grd. cf. définition 34.

grds. cf. définition 34.

global unit literal elimination. cf. `qpro`.

graphe acyclique orienté. Un *graphe acyclique orienté* (en. « directed acyclic graph » ou « DAG ») est un graphe dont les arcs sont orientés et sans cycle (un *cycle* est une chaîne d'arcs telle que le premier sommet et le dernier sommet sont identiques).

heuristique. Une heuristique est une règle qui a intérêt à être utilisée en général, parce qu'on sait qu'elle conduit souvent à la solution, bien qu'il n'y ait aucune certitude sur sa pertinence dans tous les cas.

hiérarchie polynomiale. [146, 217, 88, 172] La *hiérarchie polynomiale* $\mathcal{HP}$ est définie grâce aux *machines de Turing* (déterministes ou non) *à oracle*. L'oracle répond en une transition si « oui » ou « non » un *mot* appartient à un certain *langage*. Selon un oracle donné O, deux classes de langage peuvent alors être définies ($M(O)$ représente le langage *décidé* par la machine de Turing à oracle selon l'oracle O) : $\mathcal{P}(O)$ est la classe des langages décidés en temps polynomial sur une machine de Turing déterministe à oracle O et $\mathcal{NP}(O)$ est la classe des langages décidés en temps polynomial sur une machine de Turing non déterministe à oracle O. Si $\mathcal{E}$ est un ensemble de langages sur un alphabet donné alors $co-\mathcal{E} = \{\overline{\mathcal{L}}$ tel que $\mathcal{L} \in \mathcal{E}\}$ avec $\overline{\mathcal{L}}$ le langage *complémentaire* de $\mathcal{L}$. (Dans le cadre d'une procédure de décision, le complémentaire du langage pour lequel la réponse est « oui » est le langage pour lequel elle est « non ».) Si $\mathcal{E}$ est un ensemble de langages alors $\mathcal{P}(\mathcal{E}) = \bigcup_{O\in\mathcal{E}} \mathcal{P}(O)$ et $\mathcal{NP}(\mathcal{E}) = \bigcup_{O\in\mathcal{E}} \mathcal{NP}(O)$. La *hiérarchie polynomiale* se définit comme étant $\{\Sigma_k^p, \Pi_k^p, \Delta_k^p, k \geq 0\}$ avec $\Sigma_0^p = \Pi_0^p = \Delta_0^p = \mathcal{P}$ et pour tout $k \geq 0$, $\Sigma_{k+1}^p = \mathcal{NP}(\Sigma_k^p)$, $\Pi_{k+1}^p = co-\mathcal{NP}(\Sigma_k^p)$, $\Delta_{k+1}^p = \mathcal{P}(\Sigma_k^p)$ (cf. *machine de Turing* pour la définition des classes $\mathcal{P}$ et $\mathcal{NP}$) ; de plus la *hiérarchie polynomiale* $\mathcal{HP} = \bigcup_{k\geq 0} \Sigma_k^p$. Clairement, $\Sigma_1^p = \mathcal{NP}$ et $\Pi_1^p = co-\mathcal{NP}$ et pour tout $k \geq 0$, $\Sigma_k^p \cup \Pi_k^p \subseteq \Delta_{k+1}^p \subseteq \Sigma_{k+1}^p \cup \Pi_{k+1}^p$. Un problème est *complet* pour une classe de complexité s'il appartient à cette classe de complexité et qu'il est au moins aussi difficile que tous les problèmes de la classe. Le *problème de validité* pour le fragment $\Sigma_k - \mathbf{QBF}$ (resp. $\Pi_k - \mathbf{QBF}$) est Σ_k^p-complet (resp. Π_k^p-complet) et le problème de validité pour les QBF est *PSPACE-complet*. Le problème *SAT* de satisfiabilité d'une formule propositionnelle est $\mathcal{NP}$-complet et le problème *TAUT* qui décide si une formule est une *tautologie* est $co-\mathcal{NP}$-complet. Il n'est toujours pas tranché de savoir si $\mathcal{P} \neq \mathcal{NP}$ ni si $\mathcal{HP} \subset PSPACE$ (ce qui est attendu pour le premier car « depuis le temps que l'on cherche » et pour le second, car sinon $\mathcal{HP} = PSPACE$ et *PSPACE* ayant des problèmes complets, la hiérarchie polynomiale en aurait aussi mais alors celle-ci s'« effondrerait » sur le niveau du problème complet le plus bas [172]).

hyper résolution binaire. 1. PROP. [20] L'*hyper résolution binaire* est une extension du principe de *résolution* propositionnelle qui applique simultanément l'étape de résolution sur une même *clause* (avec un nombre quelconque de *littéraux*) pour des clauses binaires ayant un littéral *complémentaire* dans cette première clause et pour second littéral, un même littéral. Cette étape d'hyper résolution binaire est basé sur la *tautologie* suivante :

$$\models [(l_1 \vee \ldots \vee l_i \vee l_{i+1} \vee \ldots \vee l_n) \wedge (l \vee \overline{l_1}) \wedge \ldots \wedge (l \vee \overline{l_i})] \rightarrow (l \vee l_{i+1} \vee \ldots \vee l_n).$$

Le littéral l est le littéral de *coupure*. L'hyper résolution binaire peut être vue comme une généralisation de la propagation unitaire en autorisant les *clauses unitaires* :

$$\models [(l_1 \vee \ldots \vee l_i \vee l_{i+1} \vee \ldots \vee l_n) \wedge \overline{l_1} \wedge \ldots \wedge \overline{l_i}] \rightarrow (l_{i+1} \vee \ldots \vee l_n).$$

L'hyper résolution binaire peut aussi être vue comme une stratégie d'application de la résolution composée avec l'*équivalence logique* $(F \vee F) \overset{0.12}{\equiv} F$:

$$\models [(l_1 \vee \ldots \vee l_i \vee l_{i+1} \vee \ldots \vee l_n) \wedge (l \vee \overline{l_1}) \wedge \ldots \wedge (l \vee \overline{l_i})] \rightarrow [(l \vee l_2 \vee \ldots l_i \vee l_{i+1} \vee \ldots \vee l_n) \wedge (l \vee \overline{l_1}) \wedge \ldots \wedge (l \vee \overline{l_i})].$$

ainsi de suite jusqu'à $\models [(l \vee l_i \vee l_{i+1} \vee \ldots \vee l_n) \wedge (l \vee \overline{l_i})] \rightarrow (l \vee l_{i+1} \vee \ldots \vee l_n)$. Après le calcul du point fixe sur l'hyper résolution binaire et la *propagation de clauses unitaires*, la *détection de littéraux en échecs* se révèle inutile. **2. QBF.** [193] L'*hyper résolution binaire* sur les QBF est similaire à l'*hyper résolution binaire* propositionnelle (cf. **1. PROP**) mais n'autorise, tout comme la *Q-résolution*, qu'une coupure sur les littéraux *existentiellement quantifiés* ; elle s'accompagne généralement d'une *réduction universelle*.

i, i^{-1}**.** cf. définition 13.

$i_\top$, $i_\bot$, $i_\neg$, $i_\wedge$**.**, $i_\vee$, $i_\rightarrow$, $i_\leftrightarrow$**.**, $i_\oplus$**.** cf. définition 13.

$I_\top$, $I_\bot$, $I_\neg$, $I_\wedge$, $I_\vee$, $I_\rightarrow$, $I_\leftrightarrow$, $I_\oplus$, $I_\exists$, $I_\forall$**.** cf. définition 14 et 19.

I^***.** cf. définition 15 et 20.

implication. cf. définition 1.

insatisfiabilité. cf. *sémantique de la logique propositionnelle.*

inversion des quantificateurs. [187] L'*inversion des quantificateurs* (en. *inversion of the quantifiers*) est une technique, pour les *procédures de décision* basée sur une $\forall$*-expansion top-down*, de diminution de l'espace de recherche pour des QBF de la forme $\exists x_1 \ldots \exists x_n \forall y_1 \ldots \forall y_p F$. Cette technique part du principe que les *modèles* de la QBF $\exists x_1 \ldots \exists x_n \forall y_1 \ldots \forall y_p F$ ne peuvent être construits qu'à partir des modèles de la QBF $\exists x_1 \ldots \exists x_n \exists y_1 \ldots \exists y_p F \equiv \exists y_1 \ldots \exists y_p \exists x_1 \ldots \exists x_n F$. Maintenant, la *détection des littéraux en échec* est appliquée en calculant $\exists x_1 \ldots \exists x_n\ [y_1 \leftarrow C_1][y_p \leftarrow C_p](F)$ $(C_1, \ldots, C_p \in \{\top, \bot\})$ pour restreindre, dans le parcours de l'*arbre sémantique* de $\exists x_1 \ldots \exists x_n \forall y_1 \ldots \forall y_p F$, les substitutions sur $x_1 \ldots x_n$.

inversion of the quantifiers. cf. *inversion des quantificateurs.*

`IQTest` **.** [227] `IQTest` est une *procédure de décision* pour le *problème de validité* des QBF *prénexe* dont la *matrice* est conjointement sous *FNC* et *FND* basée sur un `QDLL`. La mise sous forme conjointement FNC/FND d'une QBF prénexe QF présuppose l'existence d'une fonction de *mise sous forme normale conjonctive* fnc par renommage de formules qui calcule un couple $(\mathcal{C}, Y)$ constituée de la FNC $\mathcal{C}$ ainsi que l'ensemble des symboles existentiellement quantifiés Y. Si $fnc(M) = (\mathcal{C}, Y_\mathcal{C})$ alors $M \equiv \exists Y_\mathcal{C} \mathcal{C}$. La FND de la formule M est calculée sur la formule $\neg M$: si $fnc(\neg M) = (\mathcal{D}, Y_\mathcal{D})$ alors $\neg M \equiv \exists Y_\mathcal{D} \mathcal{D}$ et donc $\neg\neg M \overset{0.4}{\equiv} M \equiv \neg \exists Y_\mathcal{D} \mathcal{D} \overset{1.4}{\equiv} \forall Y_\mathcal{D} \neg \mathcal{D}$, $\neg\mathcal{D}$ étant aisément mise en FND. La forme conjointe FNC/FND est alors calculée ainsi (grâce à l'*équivalence logique* $M \equiv (M \vee \neg\neg M)$, cette forme est appelée « Combined Conjunctive-Disjunctive Normal Form » ou « CCDNF ») :

$QM \equiv Q(M \vee \neg\neg M) \equiv Q(\exists Y_{\mathcal{C}} \mathcal{C} \vee \forall Y_{\mathcal{D}} \neg \mathcal{D}) \equiv Q \exists Y_{\mathcal{C}} \forall Y_{\mathcal{D}} (\mathcal{C} \vee \neg \mathcal{D})$.
La motivation d'une telle structure de formule est le traitement symétrique aussi bien du « conflict learning »(50) par des *clauses* que du « *solution learning* » par des *cubes*.

`IQTest`

Décision	$PSPACE$	
Sémantique	Polynomial en espace	FNC/FND prénexe
	Monolithique	
	FNC/FND prénexe $\exists$-learning $\forall$-learning Propagation unitaire Littéraux monotones Réduction universelle	

langage. cf. *machine de Turing.*

langage de la logique des prédicats. Un *langage non-logique*, $\mathcal{L}$, associé à l'alphabet logique, est la donnée d'un ensemble $\mathcal{SF}_{\mathcal{L}}$ de *symboles de fonction* et d'un ensemble $\mathcal{SP}_{\mathcal{L}}$ de *symboles de prédicat* (c'est-à-dire une fonction à valeur dans **BOOL**). $\mathcal{SF}_{\mathcal{L}}$ et $\mathcal{SP}_{\mathcal{L}}$ sont dénombrables et possiblement vides. $\mathcal{SF}_{\mathcal{L}}$, $\mathcal{SP}_{\mathcal{L}}$ et l'ensemble des *variables* sont disjoints deux à deux. L'ensemble des symboles de fonction d'arité 0 ou *symboles de constante* est noté $\mathcal{SC}_{\mathcal{L}}$. L'ensemble des *termes* du langage $\mathcal{L}$ est le plus petit ensemble vérifiant les conditions suivantes :
- toute variable est un terme ;
- tout élément de $\mathcal{SC}_{\mathcal{L}}$ est un terme ;
- si $t_1, \ldots, t_n$ sont des termes et f est un élément $\mathcal{SF}_{\mathcal{L}}$ d'arité $n > 0$ alors $f t_1 \ldots t_n$ est un terme.

L'ensemble des *formules atomiques* (ou atomes) d'un langage $\mathcal{L}$ est constitué (exclusivement) ainsi :
- $\bot$ et $\top$ sont une formule atomique ;
- si p est un élément de $\mathcal{SP}_{\mathcal{L}}$ d'arité 0 alors p est une formule atomique ;
- si p est un symbole de prédicat d'arité $n > 0$ et $t_1, \ldots, t_n$ sont des *termes* construits sur $\mathcal{SF}_{\mathcal{L}}$ alors $p t_1 \ldots t_n$ est une formule atomique.

L'ensemble des *formules*, $\mathbf{PRED}_{\mathcal{L}}$, d'un langage $\mathcal{L}$ est le plus petit ensemble vérifiant les conditions suivantes :
- toute formule atomique est une formule ;
- si A est une formule alors $\neg A$ est aussi une formule ;
- si A et B sont des formules alors $(A \wedge B)$, $(A \vee B)$, $(A \rightarrow B)$ et $(A \leftrightarrow B)$ sont des formules ;
- si x est une variable et A une formule alors $(\forall x\ A)$ et $(\exists x\ A)$ sont des formules.

Tout ceci défini le *langage de la logique des prédicats* qui est aussi appelé *langage de la logique du premier ordre* . Une occurrence d'une variable est *libre* si elle n'apparaît pas sous la portée d'un quantificateur associé à cette variable (sinon elle est *liée*).

langage de la logique du premier ordre. cf. *langage de la logique des prédicats.*

langage de la logique propositionnelle. Le *langage de la logique propositionnelle* est constitué d'un ensemble de *symboles propositionnels*, noté $\mathcal{SP}$, deux *constantes logiques*, $\top$ et $\bot$ et un ensemble de *connecteurs logiques*, les plus souvent définis étant : l'unique *connecteur unaire* (avec un argument), la *négation* (de symbole $\neg$) et les *connecteurs binaires* (avec deux arguments), la *conjonction* (de symbole $\wedge$), la *disjonction* (de symbole $\vee$), l'*implication* (de symbole $\rightarrow$), la *bi-implication* et le *ou exclusif* (de symbole $\oplus$). L'ensemble des *formules propositionnelles*, **PROP**, est défini comme étant le plus petit ensemble vérifiant les conditions suivantes : tout élément de $\mathcal{SP} \cup \{\bot, \top\}$ est un élément de **PROP** ; si A est un élément de **PROP** alors $\neg A$ est aussi un élément de **PROP** ; si A et B sont des éléments de **PROP** alors $(A \wedge B)$, $(A \vee B)$, $(A \rightarrow B)$, $(A \leftrightarrow B)$ et $(A \oplus B)$ sont des éléments de **PROP**.

langage non-logique. cf. *langage de la logique des prédicats.*

lemme. cf. *apprentissage de lemme.*

libre. cf. définition 4.

librement substituable. cf. définition 10.

lié. cf. définition 4.

lieur. 1. QBF. cf. définition 5. **1. PRED.** cf. *prénexe.*

lieur. cf définition 34.

linéairement clos. cf. *clos.*

linéarisation. cf. § 1.3.3.

littéral. 1. QBF. cf. définition 7. **2. PROP** Un *littéral* est un *symbole propositionnel* (*littéral positif*) ou sa *négation* (*littéral négatif*). Un littéral est *monotone* (ou *pur*) dans une formule sous FNC s'il n'apparaît que dans une seule *polarité* : littéral monotone *positif* (resp. littéral monotone *négatif*) si le littéral apparaît uniquement sous sa forme positive (resp. négative). **3. PRED.** Un *littéral* est un *atome* (*littéral positif*) ou sa *négation* (*littéral négatif*).

littéral existentiellement quantifié. cf. définition 7.

littéral monotone. cf. *littéral.*

littéral négatif. cf. *littéral.*

littéral positif. cf. *littéral.*

littéral pur. cf. *litteral.*

littéral universellement quantifié. cf. définition 7.

littéraux complémentaires. cf. définition 7.

littéraux conjugués. cf. définition 7.

local unit literal elimination. cf. `qpro`.

logique des prédicats. 1. Morph. cf. *langage de la logique des prédicats.* **2. Sem.** cf. *sémantique de la logique des prédicats.*

logique propositionnelle. 1. Morph. cf. *langage de la logique propositionnelle.* **2. Sem.** cf. *sémantique de la logique propositionnelle.* **3. Syn.** cf. *calcul des séquents.*

lois de de Morgan. cf. *equivalence logique.*

machine de Turing. La *machine de Turing*, due à **Alan Turing** (1912-1954), a pour but de formaliser la notion de *procédure effective* ; elle s'appuie sur les définitions de la théorie des langages suivants : un *alphabet* est un ensemble fini de caractères, un *mot* est une suite finie d'éléments d'un alphabet, un *langage* est un ensemble de mots définis sur un même alphabet, le *complémentaire* d'un langage $\mathcal{L}$ sur un alphabet est le langage $\overline{\mathcal{L}}$ des mots sur ce même alphabet mais qui ne sont pas dans le langage $\mathcal{L}$. Les composants d'une machine de Turing sont décrits tout d'abord informellement puis son fonctionnement. Une machine de Turing est la donnée [225] : une tête de lecture qui indique le caractère suivant à lire sur le ruban d'entrée où est placé l'instance du problème ; un ensemble d'états finis (état initial et états accepteurs) ; une fonction de transition qui indique pour chaque état et chaque symbole lu l'état suivant, le symbole écrit et le sens de déplacement de la tête de lecture ; et s'exécute ainsi :

– Départ : état initial et tête de lecture sur le premier caractère du mot d'entrée ;
– A chaque étape :
  1. lecture du caractère sous la tête de lecture ;
  2. remplacement du caractère lu par celui de la fonction de transition ;
  3. déplacement de la tête de lecture selon la fonction de transition ;
  4. changement d'état selon la fonction de transition.
– Arrêt : Le mot est *accepté* si la machine de Turing atteint un état accepteur.

Une telle machine de Turing est qualifiée de *déterministe* car la transition est entièrement déterminée par le caractère lu, l'état courant et la fonction de transition. Une machine de Turing est *non déterministe* si la fonction de transition est remplacée par une *relation* de transition. Dans une telle machine, le choix de la transition, en accord avec la relation de transition, se fait soit selon une certaine stratégie, soit au hasard, soit selon un *oracle* qui guide dans le choix vers un état accepteur. Un mot est accepté par une machine de Turing non déterministe s'il existe au moins une exécution qui mène à un état accepteur. Un *langage* est *décidé* par une machine de Turing (déterministe) si la machine accepte le langage et n'a pas d'exécution infinie, c'est alors une *procédure de décision*. Une fonction ou procédure est *calculable* s'il existe une machine de Turing qui permet de la définir. La notion de machine de Turing est capitale en théorie de la complexité car elle permet de définir des classes de langages acceptés et particulièrement les classes $\mathcal{P}$, $\mathcal{NP}$, $PSPACE$ et la *hiérarchie polynomiale*. Le temps d'exécution d'une machine de Turing est compté en nombre d'applications de la fonction (relation) de transition tandis que l'espace d'exécution d'une machine de Turing est compté en nombre de cases du ruban d'entrée utilisées lors de l'exécution. Trois classes de machines de Turing sont définies pour exprimer les classes $\mathcal{P}$, $\mathcal{NP}$, et $PSPACE$: $DTIME(f(n))$ est la classe des langages acceptés par une machine de Turing déterministe en un temps borné par $f(n)$, n étant la taille du mot d'entrée ; de même, $NTIME(f(n))$ est la classe des langages acceptés par une machine de Turing non déterministe en un temps borné par $f(n)$; $DSPACE(f(n))$ est la classe des langages acceptés par une machine de

Turing déterministe dans un espace borné par $f(n)$. Alors $\mathcal{P} = \bigcup_{k\geq 1} DTIME(n^k)$ est la classe des langages acceptés polynomialement (en temps) par une machine de Turing déterministe, $\mathcal{NP} = \bigcup_{k\geq 1} NTIME(n^k)$ est la classe des langages acceptés polynomialement (en temps) par une machine de Turing non déterministe et $PSPACE = \bigcup_{k\geq 1} DSPACE(n^k)$ est la classe des langages acceptés polynomialement en espace par une machine de Turing déterministe. Un intéressant résultat, dont les conséquences sont patentes en pratique, est que $\mathcal{NP} = \bigcup_{k\geq 1} DTIME(2^{n^k})$ [88].

machine de Turing à oracle. cf. *hiérarchie polynomiale.*

machine de Turing déterministe. cf. *machine de Turing.*

machine de Turing non déterministe. cf. *machine de Turing.*

matrice. **1.** QBF. cf. définition 5. **2.** PRED. cf. *prénexe.*

métaheuristique. Les *métaheuristiques* forment une famille d'algorithmes visant à résoudre des problèmes dont il n'est pas connu d'algorithme en temps polynomial. Chaque métaheuristique forme un cadre dans lequel l'expression du problème doit être adapté. Les métaheuristiques ne parcourt qu'une partie de l'espace de recherche. Parmi les plus connues peuvent être citées : la *recherche locale* via la *recherche tabou* [110], l'*algorithmique génétique* [147] et l'*optimisation par colonies de fourmis* [39].

minimalité d'une QBF. cf. définition 38.

miniscoping. cf. *dépendance.*

mise sous forme normale conjonctive. cf. § 1.3.3.

mise sous forme normale disjonctive. cf. § 1.3.3.

mise sous forme normale négative. cf. § 1.3.3.

mise sous forme prénexe. cf. § 1.3.3.

modèle. **1.** QBF. cf. définition 21. **2.** PROP. cf. *sémantique de la logique propositionnelle.* **3.** PRED. cf. *sémantique de la logique des prédicats.*

modèle stable. cf. *sémantique des modèles stables.*

monotone. **1.** QBF. cf. *littéral monotone.* **2.** Un mécanisme de raisonnement est *monotone* si l'ajout de nouvelles hypothèses ne remet pas en cause ce qui a été déduit.

mot. cf. *machine de Turing.*

multi-résolution. cf. *Davis-Putnam.*

négation. cf. définition 1.

négation par défaut. cf. *programme logique normal*

Nenofex. [136] `Nenofex` est une *procédure de décision* pour le *problème de validité* des QBF *prénexes* dont la *matrice* est sous *FNN* qui étend la procédure de décision `quantor` qui est restreinte aux QBF prénexes dont la matrice est sous *FNC*. Une forme de *miniscoping* est opérée à partir des sous-formules permettant de calculer les parties de la QBF qui doivent être dupliquées lors de l'*expansion bottom-up* qui peut être effectuée sur le symbole propositionnel existentiellement ou universellement

quantifié le *plus interne*. L'expansion s'applique aussi aux symboles propositionnels universellement quantifiés tels que seuls des symboles propositionnels existentiellement quantifiés sont plus internes (alors ces symboles dépendant des symboles universellement quantifiés doivent être dupliqués). La procédure `Nenofex` est polynomiale en espace à l'inverse de la procédure `quantor` qu'elle étend car le fragment FNN est *linéairement clos* pour l'expansion d'un quantificateur existentiel le plus interne. Lorsque la QBF ne contient plus que des quantificateurs existentiels (resp. universels), la formule est mise sous FNC et une procédure pour le problème SAT (resp. $TAUT$) en *décide*.

`Nenofex`

Décision	$PSPACE$	
Sémantique	Polynomial en espace	FNN prénexe
	Monolithique	
	FNN $\exists$-expansion$\uparrow$ $\forall$-expansion$\uparrow$ Simplification $\top$ et $\bot$	

NNF. cf. *forme normale*.
non-monotone. cf. *monotone*.
non-validité. cf. *validité*.
normal logic program. cf. *programme logique normal*.
$\mathcal{NP}$. cf. *hiérarchie polynomiale*.
$\mathcal{NP}$-complet. cf. *hiérarchie polynomiale*.

OBDD. cf. *Diagramme de décision binaire*.
occurrence d'un symbole propositionnel. cf. définition 4.
optimalité d'une base littérale. cf. définition 37.
oracle. cf. *hiérarchie polynomiale*.
ou-exclusif. cf. définitions 1 et 13.

$\mathcal{P}$. cf. *hiérarchie polynomiale*.
$\mathcal{P} \neq \mathcal{NP}$. cf. *hiérarchie polynomiale*.
pairwise falsity. cf. *falsifiabilité par littéraux complementaires*.
plus petit modèle de Herbrand. cf. *programmation logique*.
polarité. cf. définition 1 pour les quantificateurs et définition 7 pour les littéraux.
poli. cf. définition 4.
polynomial en espace. cf. *hiérarchie polynomiale*.
polynomialement clos. cf. *clos*.

politique. [55, 56, 54] Une *politique* est une présentation arborescente des *valuations fonctionnelles* pour des QBF *prénexes* (*polies*) *closes*. L'ensemble des politiques (totales), $\mathcal{P}(Q)$, pour un lieur Q est défini inductivement par :

$$\begin{aligned}\mathcal{P}(\varepsilon) &= \{\lambda\}\\ \mathcal{P}(\exists yQ) &= \{l;\pi \mid l \in \{y, \neg y\}, \pi \in \mathcal{P}(Q)\}\\ \mathcal{P}(\forall yQ) &= \{y \mapsto \pi, \neg y \mapsto \pi' \mid \pi, \pi' \in \mathcal{P}(Q)\}\end{aligned}$$

L'opérateur « ; » représente la composition séquentielle des politiques. Le symbole λ représente la politique vide. Si π est une politique alors $\pi;\lambda$ est simplifiée en π. Une politique *valide* correspond à un *modèle* QBF. Dans [40], des notions de *stratégie* et *stratégie gagnante* similaires, respectivement, à celles de politique et politique valide ont été proposées.

portée d'un quantificateur. cf. définition 4.

`PQSOLVE` **.** cf. `QSOLVE`.

prédicat. cf. *langage de la logique des prédicats*.

prénexe. **1.** QBF**.** cf. définition 5. **2.** PRED**.** Une formule de la *logique des prédicats* est sous forme *prénexe* si elle est constituée d'un *lieur*, c'est-à-dire une chaîne de *quantificateurs* associées à leurs *variables*, et d'une *matrice*, c'est-à-dire une formule sans quantificateurs.

preuve. **1.** QBF**.** cf. définition 30. **2.** Syn**.** cf. *calcul des séquents*.

problème de décision. **1.** QBF**.** cf. définitions 16 et 22. **2.** cf. *machine de Turing*.

problème de satisfaction de contraintes. [141, 5] Un problème de satisfaction de contraintes est la donnée d'un triplet $\langle X, D, C\rangle$ avec X un ensemble fini de variables, D qui associe à chaque variable un domaine de valeurs et C un ensemble fini de contraintes, cette dernière étant un sous-ensemble du produit cartésien des domaines associés aux variables qui la constituent. La solution à un problème de satisfaction de contraintes est une affectation qui associe à chaque variable une valeur telle qu'elle satisfait toutes les contraintes. Dans le cas des contraintes sur des domaines finis, une contrainte satisfait la consistance d'arc si chaque valeur de chaque variable appartient à une solution de la contrainte. Les algorithmes de la famille *AC* éliminent les valeurs des domaines associés aux variables pour lesquelles la propriété de consistance d'arc n'est pas vérifiée pour toutes les contraintes. En particulier, l'algorithme *AC*3 manipule une file d'attente dans laquelle sont déposées les contraintes « réveillées » parce qu'au moins une des variables qui les composent ont eu leurs domaines restreints ; l'algorithme est initialisé avec l'ensemble des contraintes et termine lorsque la liste est vide ce qui signifie que toutes les contraintes restantes ont la propriété de consistance d'arc. La consistance d'arc a été étendue aux *problèmes de satisfaction de contraintes quantifiées* [41].

problème de satisfaction de contraintes quantifiées. [42, 96] Le domaine des *problèmes de satisfaction de contraintes quantifiées* (en. *quantified constraint satisfaction problem* ou *QCSP*) étend celui des *problèmes de satisfaction de problèmes* dans lesquels les variables des contraintes ne sont plus uniquement quantifiées existentiellement mais peuvent être aussi universellement quantifiées.

problème de validité. cf. définition 16.

problème du choix du prochain mouvement. cf. définition 36 et théorème 3.

procédure de décision. cf. *machine de Turing*.

programmation en clauses de Horn. cf. *programmation logique*.

programmation logique. L'invention du paradigme de la programmation logique [52, 125] a été une révolution dans le domaine de l'intelligence artificielle car il a montré que le *langage de la logique des prédicats*, jusqu'alors cantonnée à la représentation de la connaissance, pouvait aussi être un moyen efficace de calcul. Ce paradigme est d'abord présenté dans son cadre général puis l'instance la plus répandue, la programmation logique en clauses de Horn, est décrite et enfin son implantation, le langage Prolog, est brièvement évoquée.

Le paradigme de la programmation logique considère le calcul en tant que déduction dans un formalisme logique. Ce paradigme est à mettre en perspective avec celui de la programmation impérative qui considère le calcul comme la modification d'un état global par des instructions et celui de la programmation fonctionnelle qui considère le calcul comme le résultat de l'évaluation d'une fonction. Un langage de la programmation logique est constitué d'un langage des données, d'un langage des programmes et d'un langage des questions ; ces deux derniers étant des fragments plus ou moins grands d'un formalisme logique. Les langages de la programmation logique sont des langages déclaratifs (dits « de cinquième génération ») : le programme déclare la structure du problème mais ne fournit pas comment le résoudre. Le formalisme logique ne doit pas seulement pouvoir représenter la connaissance mais il doit aussi être suffisamment puissant pour pouvoir définir l'ensemble des fonctions (et procédures) calculables (cf *machine de Turing*).

Pour réaliser le calcul, un démonstrateur de théorèmes restreint aux fragments choisis est nécessaire mais il ne peut être quelconque car il doit contenir un principe de déduction. L'ensemble des hypothèses du théorème est décrit dans le langage des programmes associé au langage des données tandis que la formule qui doit être déduite des hypothèses est décrite dans le langage des questions associé au langage des données. Un programme logique exprime la connaissance du domaine en des formules dont les variables sont implicitement universellement quantifiées tandis que les questions sur cette connaissance s'appuient sur des variables existentiellement quantifiées. Une réponse attendue d'un tel programme est un ensemble d'instances pour les variables existentiellement quantifiées. L'activité de programmation pouvant être vue comme la somme d'une composante « logique », d'une composante « structure de donnée » et d'une composante « contrôle » [224, 126], cette dernière disparaît si le paradigme est poussé jusqu'à l'extrême, c'est-à-dire jusqu'à utiliser un démonstrateur de théorèmes dont le fonctionnement reste opérationnellement opaque. Un premier élément qui discrimine un démonstrateur de théorèmes quelconque d'un mécanisme sous-jacent à l'exécution d'un langage de la programmation logique est que le premier établit une décision si « oui » ou « non » le théorème est vrai tandis que le second, à partir d'une même connaissance, calcule des réponses pour des questions existentielles. Un second élément est que le démonstrateur de théorèmes qui sert de mécanisme à l'exécution du programme devant être basé sur de la déduction, il offre une sémantique procédurale au paradigme de la program-

mation logique.
Les langages de la programmation logique sont des langages relationnels : le calcul est défini en terme de relations comme dans les langages de base de données ; ainsi a contrario des fonctions qui ont des paramètres en entrée et un résultat en sortie, les paramètres d'une relation suivent un principe de réversibilité et ne sont statiquement ni en entrée ni en sortie mais dynamiquement soit l'un, soit l'autre, voire les deux. Une conséquence au caractère relationnel des langages de programmation logique est qu'ils sont indéterministes, c'est-à-dire que la réponse à la question posée n'est pas nécessairement unique et (une partie suffisante de) l'ensemble de ces réponses est calculé(e).
Les langages de la programmation logique sont des langages symboliques : le langage des données est un ensemble construit sur des symboles sans sémantique propre dont la signification est celle accordée par le programmeur (a contrario des valeurs numériques), ainsi le calcul est-il purement syntaxique et, hors l'ajout de domaines spécifiques tels que les entiers, des éléments du langage des données ne sont jamais mis en rapport sémantiquement mais uniquement syntaxiquement.
La programmation logique en clauses de Horn est sans doute l'instance la plus connue et répandue du paradigme de la programmation logique [135]. Le langage des données est ici le langage des *termes* construit inductivement sur un ensemble de symboles de fonction et un ensemble de symboles de constante. Les langages des programmes et des questions s'appuient sur la notion d'*atome*, de *littéral* et de *clause* selon un ensemble de symboles de prédicat. Une clause contenant exactement un seul littéral positif est une clause définie ; si une telle clause contient un unique littéral alors c'est un fait sinon elle est notée $A \leftarrow A_1, \ldots, A_n$ avec $A_1, \ldots, A_n$ les atomes des littéraux négatifs et A l'unique littéral positif de la clause. Le langage des programmes est celui de l'ensemble des clauses définies (programme logique défini). Cet ensemble a pour sémantique la conjonction des clauses qui le constituent. Le langage des questions est celui des conjonctions existentiellement quantifiées de littéraux positifs.
En *sémantique de la logique des prédicats*, un modèle est une interprétation des symboles d'une formule qui la rend vraie. Le mécanisme de déduction sous-jacent pour la programmation logique en clauses de Horn est celui de la conséquence logique : une question est conséquence logique d'un programme si toute interprétation qui est modèle du programme (c'est-à-dire simultanément modèle pour toutes les clauses), est aussi un modèle pour la question. Démontrer en logique qu'une formule est conséquence logique d'un ensemble de formules est équivalent à démontrer que la conjonction des formules de l'ensemble augmenté de la négation de la formule, qui doit être conséquence, est insatisfiable, c'est-à-dire n'admet pas de modèle. La négation d'une conjonction existentiellement quantifiée de littéraux positifs est équivalente à une clause constituée uniquement de littéraux négatifs. La réunion d'un programme et de la négation d'une question est donc un ensemble de clauses telles que chacune d'entre-elles contiendra au plus un littéral positif, ce qui est la définition de la clause de Horn et ce qui justifie le vocable de « programmation logique en clauses de Horn ».
Démontrer qu'un ensemble de formules n'admet pas de modèle, et ce dans n'importe quelle interprétation, est une tâche insurmontable dans sa grande généralité mais qui

ne l'est pas dans le cas très particulier d'un ensemble de clauses car il suffit alors de démontrer qu'il n'y a pas de modèle de Herbrand. Une interprétation de Herbrand est une interprétation très simple qui fait le pont entre la syntaxe et la sémantique : les symboles de constante et de fonction sont interprétés en eux-mêmes. L'univers de Herbrand pour un programme est l'ensemble des termes qui peuvent être construits sans variable (et en ajoutant un nouveau symbole de constante, s'il n'y en a pas). La base de Herbrand pour un programme est l'ensemble des atomes qui peuvent être construits à partir des éléments de l'univers de Herbrand et des symboles de prédicat. Une interprétation de Herbrand est alors simplement un sous ensemble de la base de Herbrand. Celle-ci est nécessairement un modèle pour un programme défini puisqu'il est constitué uniquement de clauses définies et l'intersection de tous les modèles de Herbrand est lui-même un modèle, le plus petit modèle de Herbrand qui correspond exactement à l'ensemble des éléments de la base de Herbrand conséquence logique du programme. Ainsi la sémantique déclarative d'un programme défini est-elle double : plus petit modèle, en tant qu'intersection des modèles, de Herbrand et ensemble des conséquences logiques du programme.
La structure des interprétations de Herbrand munie de l'inclusion des ensembles forme un treillis complet pour tout programme défini. Sur cette structure peut être définie une sémantique procédurale dite « en chaînage avant » car partant des faits, c'est-à-dire des clauses ne contenant aucun littéral négatif, comme point fixe d'une fonction T_P de l'ensemble des interprétations de Herbrand dans lui-même et définie pour un programme défini P comme suit : si I est une interprétation de Herbrand et $A \leftarrow A_1, \ldots, A_n$ est une instance sans variable d'une clause définie de P alors $A \in T_P(I)$. Cette fonction est non seulement monotone mais elle est aussi sup-continue, c'est-à-dire elle préserve les bornes supérieures. Le plus petit point fixe de cette fonction pour un programme P coïncide avec le plus petit modèle de Herbrand. La résolution SLD, ultime sémantique procédurale dite « en chaînage arrière » car partant de la question, est une restriction de la *résolution* de Robinson et utilise l'unification comme unique mécanisme de passage de paramètres de la programmation logique en clauses de Horn. La résolution SLD applique de manière indéterministe en une dérivation SLD la règle de résolution SLD suivante : si $Q_1, \ldots, Q_{m-1}, Q_m, Q_{m+1}, \ldots, Q_r$ est une conjonction d'atomes et $A \leftarrow A_1, \ldots, A_n$ est (une copie avec des variables nouvellement renommées de manière cohérente d') une clause définie telle que θ est l'unificateur le plus général de Q_m et A alors $\theta(Q_1, \ldots, Q_{m-1}, A_1, \ldots, A_n, Q_{m+1}, \ldots, Q_r)$ est une résolvante selon la clause définie et l'atome sélectionné Q_m. La résolution SLD [137, 140] est une restriction Linéaire avec fonction de Sélection pour des clauses Définies de la résolution de Robinson [189]. Une dérivation SLD d'une question mène à un échec si au moins un des atomes ne peut être éliminé et à un succès si la règle de résolution SLD parvient à éliminer tous les atomes et atteindre une résolvante vide, un succès. Le cumul des unificateurs appliqué à la question initiale est alors la réponse calculée. Une fois la stratégie de sélection de l'atome dans la résolvante fixée, l'ensemble des dérivations pour une question donnée forme un arbre SLD ; la stratégie de recherche étant la manière de parcourir cet arbre. Si la stratégie de recherche est équitable, c'est-à-dire

que sur chaque résolvante sera appliquée, si possible, la règle de résolution SLD alors la stratégie de sélection peut être quelconque. L'ensemble des éléments de la base de Herbrand d'un programme défini P tels qu'il existe pour chaque élément une dérivation SLD menant à un succès coïncide avec le plus petit modèle de Herbrand de P. Ainsi la sémantique procédurale d'un programme défini est-elle aussi double et coïncide avec les deux sémantiques déclaratives offrant à tout programme une double lecture, une double approche de la programmation. La sélection de l'atome dans la résolvante et la sélection de la clause à unifier avec l'atome ouvrent la possibilité à deux formes de parallélisme.

Le langage Prolog (pour « Programmation en Logique ») est une instance particulière de la programmation logique en clauses de Horn plus « programmation » que « logique ». La programmation logique en clauses de Horn est un outil théorique qui induit par l'indéterminisme, potentiellement une infinité de dérivation SLD à mener en parallèle. Seule une stratégie de recherche équitable garantit la complétude de la résolution SLD, comme par exemple par un parcours de l'arbre SLD en largeur d'abord, mais cela induit nécessairement un mécanisme trop coûteux pour être efficace ; en conséquence, c'est une stratégie de recherche par un parcours de l'arbre SLD en profondeur d'abord qui est présente dans le mécanisme de gestion de l'indéterminisme, faisant appel à une pile de *retour-arrière* pour sauvegarder les points de choix qui conservent les branches de l'arbre SLD non encore explorées, abandonnant par là même la complétude du langage en tant que démonstrateur de théorèmes. Une autre source d'inefficacité potentielle est l'utilisation du test d'occurrence dans l'unification qui interdit d'unifier une variable avec un terme la contenant, test nécessaire à la correction de la règle de résolution SLD ; ce test nécessitant le parcours d'un terme, il est abandonné dans Prolog et, par là même, la correction du langage en tant que démonstrateur de théorèmes.

Bien que la programmation logique en clauses de Horn ait la puissance du calculable, le fondement procédural qu'est la résolution SLD a été « étendu » pour offrir un plus grand pragmatisme, pouvoir d'expressivité et de contrôle. Le modèle purement syntaxique de la programmation logique est particulièrement contraignant lorsqu'il s'agit de la manipulation des entiers qui sont le plus souvent représentés par des nombres de Péano (basé sur un symbole de constante « zéro » et un symbole de fonction d'arité un « successeur »). Une telle représentation linéaire en taille par rapport à la valeur de l'entier induit des sauts de complexité qui pénalisent les algorithmes. Par pragmatisme, dans Prolog a été intégré un évaluateur pour l'arithmétique utilisant les « entiers de la machine ». Ces derniers n'étant pas définis par induction, les propriétés par rapport au théorème d'induction ne sont pas conservées ; de plus cet évaluateur étant fonctionnel, il brise le caractère relationnel de toute définition incluant l'utilisation de cet évaluateur et lui fait perdre la réversibilité. Dans le domaine de l'expressivité, l'extension majeure est la possibilité de déduire des informations négatives. Le programme étant défini, ceci est impossible sans rajouter une nouvelle règle. La plus communément rajoutée est la « négation par l'échec sous hypothèse de monde clos » : si un atome sans variable n'est pas conséquence logique d'un programme alors la négation de cet atome est déduit ; cette nouvelle règle ne recouvre

pas la négation de la logique car la négation de l'atome n'est pas conséquence logique du programme. Sans quitter la logique, le principal moyen de contrôle est l'ordre des atomes dans les clauses et l'ordre de celles-ci dans le programme. Un mécanisme de contrôle extra-logique, la coupure, permet d'éliminer des branches de l'arbre SLD. Si ce mécanisme s'avère très utile pour éliminer des branches infinies ou ne menant qu'à des échecs ou des succès déjà calculés, il se révèle très dangereux car il peut aussi supprimer des branches menant à des succès non encore calculés et modifier ainsi la sémantique du programme sans coupure.

Même si Prolog est un langage de programmation qui a la puissance du calculable, de par ses caractéristiques, il est particulièrement bien adapté aux domaines symboliques, tels que celui du traitement de la langue naturelle, domaine où la programmation logique est née, celui, très proche du précédent, des automates et de la compilation puisque Prolog est indéterministe, du domaine des systèmes manipulant symboliquement une base de connaissances puisqu'offrant un mécanisme de déduction, mais aussi dans le domaine du prototypage et de la spécification exécutable, puisqu'en première approche, le contrôle peut être ignoré. Les implantations abouties de Prolog intègrent de nombreux modules permettant d'utiliser Prolog comme un langage « de glu » entre des solveurs de contraintes, des applications internet, de l'interface graphique, etc mais aussi de dialoguer avec d'autres langages de programmation, le plus souvent impératifs.

programmation logique normale. cf. *programme logique normal.*

programmation par ensembles réponses. La *programmation par ensembles réponses* (en. *Answer Set Programming* ou *ASP*) est un représentant de la famille des langages de programmation déclaratifs qui a pour but de représenter un problème en un *programme logique* dont la sémantique définit un ensemble de *modèles stables* (des *ensembles réponses*) qui codent les solutions du problème. Un des représentants de ce style de langage de programmation est la *programmation logique normale* sous la *sémantique des modèles stables.*

programme logique défini. cf. *programmation logique.*

programme logique normal. Un *programme logique normal* est un ensemble de règles de la forme : $c \leftarrow a_1,\ \ldots,\ a_n,\ not\ b_1,\ \ldots,\ not\ b_m.$ où $n \geq 0$, $m \geq 0$, $\{a_1, \ldots, a_n, b_1, \ldots, b_m, c\}$ des *atomes* complètement instanciés. Le symbole *not b* représente la *négation par défaut.* La signification intuitive d'une telle règle est : « si vous avez tous les a_i et aucun b_j, alors vous pouvez conclure c ».

Pour une règle r : $corps^+(r) = \{a_1,\ \ldots,\ a_n\}$ est le pré-requis (ou corps) positif de la règle r, $corps^-(r) = \{b_1,\ \ldots,\ b_m\}$ est le pré-requis (ou corps) négatif de la règle r et $tête(r) = c$ est la conclusion (ou tête) de la règle r et $r^+ = (tête(r) \leftarrow corps^+(r).)$ est la projection positive de la règle r. Une contrainte sans tête est une *contrainte.* Une des sémantiques très utilisée pour les programmes logiques normaux est la *sémantique des modèles stables.*

Prolog. cf. *programmation logique.*

PROP. cf. *logique propositionnelle.*

propagation de clauses unitaires. **1.** QBF. Une QBF sous *forme normale conjonctive* qui contient une *clause unitaire*, c'est-à-dire une *clause* constituée d'un unique

littéral existentiellement quantifié, est équivalente au sens de la préservation des modèles propositionnels, par $Q^{\equiv}.2$ et $Q^{\equiv}.3$, à la QBF dont toutes les clauses contenant le littéral sont supprimées (sauf la clause unitaire dans le cas de la préservation des modèles QBF, par $Q^{\cong}.3$ et $Q^{\cong}.4$ ainsi que toutes les occurrences du *complémentaire* du littéral. Si la *réduction uni- verselle* est appliquée avant la détection des clauses unitaires, la définition ci-dessus de la clause unitaire est suffisante sinon une clause est unitaire si l'unique littéral existentiellement quantifié de la clause précède l'ensemble des littéraux *universellement quantifiés* de celle-ci. Une QBF sous *forme normale disjonctive* qui contient un *cube unitaire* (c'est-à-dire un *cube* constituée d'un unique *littéral*) pour un littéral universellement quantifié est équivalente au sens de la préservation des modèles propositionnels, par $Q^{\equiv}.4$ et $Q^{\equiv}.5$ et des modèles QBF, par $Q^{\cong}.5$ et $Q^{\cong}.6$, à la QBF dont tous les cubes contenant le littéral sont supprimés ainsi que toutes les occurrences du *complémentaire* du littéral. Dans le cas d'une QBF sous *forme normale négative*, la propagation de clauses unitaires est étendue en l'*élimination d'un littéral unitaire localement* (en. *local unit literal elimination*) et en l'*élimination d'un littéral unitaire globalement* (en. *global unit literal elimination*) [80]. L'élimination d'un littéral unitaire localement remplace, tout en préservant les modèles (propositionnels), une sous-formule $(x \wedge \phi)$ (resp. $(\neg x \wedge \phi)$, $(x \vee \phi)$, $(\neg x \vee \phi)$) par $[x \leftarrow \top](\phi)$ (resp. $[x \leftarrow \bot](\phi)$, $[x \leftarrow \bot](\phi)$, $[x \leftarrow \top](\phi)$). L'élimination d'un littéral unitaire globalement remplace, tout en préservant les modèles (propositionnels), une sous-formule $(\exists x\ (x \wedge \phi))$ (resp. $(\exists x\ (\neg x \wedge \phi))$, $(\forall x\ (x \vee \phi))$, $(\forall x\ (\neg x \vee \phi))$) par $[x \leftarrow \top](\phi)$ (resp. $[x \leftarrow \bot](\phi)$, $[x \leftarrow \bot](\phi)$, $[x \leftarrow \top](\phi)$). **2.** PROP. Une *formule propositionnelle* sous forme normale conjonctive qui contient une *clause unitaire* (c'est-à-dire une *clause* constituée d'un unique *littéral*) est équivalente, par *propagation d'une clause unitaire* à la formule propositionnelle dont toutes les clauses contenant le littéral sont supprimées ainsi que toutes les occurrences du *complémentaire* de ce littéral.

propagation de littéraux monotones. **1.** QBF. Le lemme "Monotone literals in $\exists$" [48] de *propagation d'un littéral monotone* (pour les QBF sous FNC) dont le symbole propositionnel est *existentiellement quantifié* est étendu aux QBF *prénexes* quelconques ainsi : soit une QBF prénexe $Q\exists|l|Q'M$ telle que l est *un littéral essentiellement positif*, c'est-à-dire positif dans la FNC (resp. négatif, c'est-à-dire négatif dans la FNC) dans M alors $Q\exists|l|Q'M \equiv QQ'[|l| \leftarrow \top](M)$ (resp. $Q\exists|l|Q'M \equiv QQ'[|l| \leftarrow \bot](M)$). Dans le cas d'une formule sous FNC, si un littéral existentiellement quantifié est monotone alors toutes les clauses le contenant peuvent être ôtées. Dans le cas d'une formule sous FND, si un littéral existentiellement quantifié est monotone alors toutes ses occurrences peuvent être ôtées. De même, le lemme "Monotone literals in $\forall$" [48] de *propagation d'un littéral monotone* (pour des QBF sous FNC) dont le symbole propositionnel est universellement quantifié est étendu aux QBF prénexes quelconques ainsi : soit une QBF $Q\forall|l|Q'M$ telle que l est un littéral essentiellement positif (resp. négatif) dans M alors $Q\forall|l|Q'M \equiv QQ'[|l| \leftarrow \bot](M)$ (resp. $Q\forall|l|Q'M \equiv QQ'[|l| \leftarrow \top](M)$). Dans le cas d'une formule sous FNC, si un littéral universellement quantifié est monotone alors toutes ses occurrences peuvent être ôtées. Dans le cas d'une formule sous FND, si un littéral universellement

quantifié est monotone alors tous les cubes le contenant peuvent être ôtes. Ces lemmes ne s'étendent pas à la relation d'équivalence qui préservent les modèles QBF : pour le premier lemme, trivialement, $(\exists x\ (\exists y\ (x \vee y))) \not\cong (\exists x\ (\exists y\ (\top \vee y)))$ car $(v, \{(x \mapsto \textbf{faux}), (y \mapsto \textbf{faux})\})$ est un modèle de la seconde sans en être un pour la première ; de même, $(\forall z\ (\exists y\ (z \vee y))) \not\cong (\forall z\ (\exists y\ (\bot \vee y)))$ car $(v, \{(y \mapsto \{(\textbf{faux} \mapsto \textbf{vrai}), (\textbf{vrai} \mapsto \textbf{faux})\})\})$ est un modèle de la première sans en être un pour la seconde (pour v une *valuation* propositionnelle quelconque). **2.** PROP. La *propagation d'un littéral monotone* est une transformation qui ôte toutes les clauses contenant le *littéral monotone* d'une *formule propositionnelle* sous forme normale conjonctive. C'est une transformation qui ne préserve que la *satisfiabilité* mais pas les *modèles* de la formule.

propagation « don't care ». La propagation « don't care » pour les QBF qui ne sont pas sous FNC [112] est identique à la propagation « don't care » pour SAT [222] : si une sous-formule a déjà sa *valeur de vérité* déterminée et qu'elle est justifiée par la valeur de vérité d'un de ses arguments alors la valeur de vérité de l'autre argument, une sous-formule, est sans importance et cette information peut être propagée à travers cette sous-formule jusqu'aux symboles propositionnels. La *propagation de clauses unitaires* est une forme de propagation « don't care ». La propagation « don't care », qui provient d'une *non-observabilité d'une sous-formule* sous certaines conditions, peut être étendue aux QBF sous FNC grâce aux littéraux introduits lors de la mise sous FNC [220].

$PSPACE$**.** cf. *hiérarchie polynomiale.*

$PSPACE$**-complet.** cf. *hiérarchie polynomiale.*

`QBDD`**.** [171] `QBDD` est une *procédure de décision* pour le *problème de validité* des QBF sous $FNC^{(103)}$ qui construit pour chaque $clause^{(39)}$ un $OBDD^{(170)}$. Un *cluster* de clauses pour un symbole propositionnel est l'ensemble des clauses qui contiennent ce symbole. Les OBDD créés sont groupés en clusters pour chaque symbole propositionnel, en commençant par le *quantificateur le plus interne* et les clauses du cluster dont le quantificateur va être éliminé sont combinées par l'application de l'opérateur de conjonction (pour les BDD) générant un nouveau OBDD dont le quantificateur est éliminé en en appliquant sa sémantique. La procédure termine avec uniquement le nœud **1** signifiant que la QBF est valide et uniquement avec le nœud **0** sinon.

`QBDD`

Décision	$PSPACE$	
Symbolique	Exponentiel en espace	FNC prénexe
	Monolithique	
	DAG ∃-expansion↑ ∀-expansion↑	

`QBDD(DLL).` cf. `QBDD(X)`.

`QBDD(LS).` cf. `QBDD(X)`.

`QBDD(X).` [15] `QBDD(X)` (dont l'instance `QBDD(DLL)` est aussi appelée `QBFBDD` [13]) est un schéma de *procédure de décision* pour le *problème de validité* des QBF *prénexes* dont la *matrice* est sous FNC. `QBDD(X)` utilise un générateur de modèles propositionnels pour les agréger en un modèle QBF construit sous la forme d'un BDD. `QBDD(X)` est un « schéma » car le générateur peut varier dans sa manière de calculer : ainsi ce schéma se décline en deux instances, l'une sous la forme d'une procédure de décision `QBDD(DLL)` avec pour générateur un `QDLL` entrelacé avec la construction du BDD et l'autre sous la forme d'une procédure incomplète au sens du premier gène (cf. chapitre 2), `QBDD(LS)`, avec pour générateur une recherche locale. Le BDD représente un modèle QBF en construction par agrégation des modèles propositionnels générés. Le BDD est réduit au fur et à mesure selon la règle de la redondance pour l'élimination des quantificateurs universels et une règle *ad hoc* d'élimination des quantificateurs existentiels.

`QBDD(DLL)`

Décision	$PSPACE$	
Sémantique	Exponentiel en espace	FNC prénexe
	À Oracle	
	$\mathcal{NP}$	
	Agrégation	

`QBDD(LS)`

Incomplet	$PSPACE$	
Sémantique	Exponentiel en espace	FNC prénexe
	À Oracle	
	$\mathcal{NP}$	
	Agrégation	

QBF. cf. définition 2.

`QBFBDD.` cf. `QBDD(X)`.

`QDLL.` cf. § 1.5.

`QKN.` cf. *résolution.*

`QMRES.` [171] `QMRES` est une *procédure de décision* pour le *problème de validité* des QBF *prénexes* dont la *matrice* est sous *FNC* basé sur l'algorithme 3 d'élimination de quantificateurs (cf. § 1.5.2). La procédure `QMRES` utilise les *ZBDD* pour représenter les clauses étendant ainsi les travaux similaires sur le problème SAT [50, 51].

`QMRES`

Décision	$PSPACE$	
Symbolique	Exponentiel en espace	FNC prénexe
	Monolithique	
	FNC prénexe	
	$\forall$-expansion$\uparrow$	
	Multi-résolution	

`Q-PREZ`. [49] `Q-PREZ` est une *procédure de décision* pour le *problème de validité* des QBF *prénexes* dont la *matrice* est sous *FNC* basé sur le `QDLL`. La procédure `Q-PREZ` utilise les *ZBDD* pour représenter les *clauses*; l'*expansion top-down* des quantificateurs est réalisée par des opérations ensemblistes sur des ZBDD. Pour réduire l'impact de l'*expansion existentielle top-down*, l'ensemble de clauses est partitionné en un ensemble de ZBDD.

`Q-PREZ`

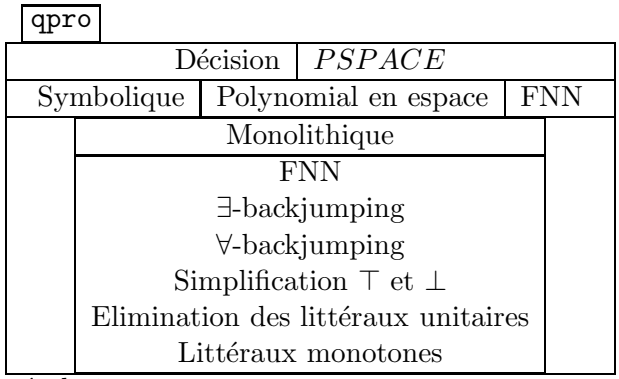
Décision	$PSPACE$	
Symbolique	Exponentiel en espace	DAG
	Monolithique	
	FNC prénexe $\exists$-expansion$\downarrow$ $\forall$-expansion$\downarrow$	

`qpro`. [79, 80] `qpro` est une *procédure de décision* pour le *problème de validité* les QBF sous FNN non *prénexes*. Cette procédure de décision s'appuie sur un système de *calcul de séquent*, le système $GQBF$ (Les QBF F, G, $(\exists x\ H)$ et $(\forall x\ H)$ sont *closes*) :

$$\frac{\Rightarrow F}{\Rightarrow (F \vee G)}(\vee') \qquad \frac{\Rightarrow G}{\Rightarrow (F \vee G)}(\vee'') \qquad \frac{\Rightarrow F \quad \Rightarrow G}{\Rightarrow (F \wedge G)}(\wedge)$$

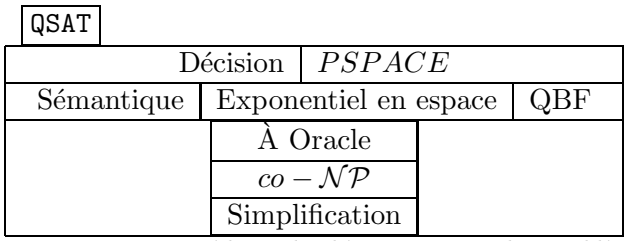
$$\frac{\Rightarrow H[x \leftarrow \bot]}{\Rightarrow (\exists x\ H)}(\exists') \qquad \frac{\Rightarrow H[x \leftarrow \top]}{\Rightarrow (\exists x\ H)}(\exists'') \qquad \frac{\Rightarrow H[x \leftarrow \bot] \quad \Rightarrow H[x \leftarrow \top]}{\Rightarrow (\forall x\ H)}(\forall)$$

L'*axiome* est $\Rightarrow \top$.
À l'instar du *système* $\mathbb{G}$ *de Gentzen*, il y a des règles d'*élimination des connecteurs* et des règles d'*élimination des quantificateurs*. Trois règles d'élimination des connecteurs : la règle d'inférence $(\vee')$ (resp. $(\vee'')$) exprime que si la QBF F (resp. G) est valide alors nécessairement aussi la QBF $(F \vee G)$; la règle d'inférence $(\wedge)$ exprime que pour que la QBF $(F \wedge G)$ soit valide il faut que les QBF F et G le soient aussi ; et trois règles d'inférence pour l'élimination des quantificateurs qui ne sont que l'application des règles d'élimination des connecteurs à la sémantique des quantificateurs. Elles apparaissent aussi comme des spécialisations des règles d'élimination à droite des quantificateurs dans le système $\mathbb{G}$. Les règles d'inférence $(\vee')$ et $(\vee'')$ reprennent les mécanismes de *retour-arrière* appliqués à l'*expansion existentielle top-down* tandis que la règle d'inférence $(\wedge)$ reprend les mécanismes de retour-arrière de l'*expansion universelle top-down*. La procédure de décision `qpro` étend la *propagation de clauses unitaires* en l'*élimination des littéraux unitaires localement* ou *globalement* et le *backjumping* (appelé *dependency-directed backtracking*) aux formules sous FNN. La procédure `qpro` intègre un *miniscoping* dynamique.

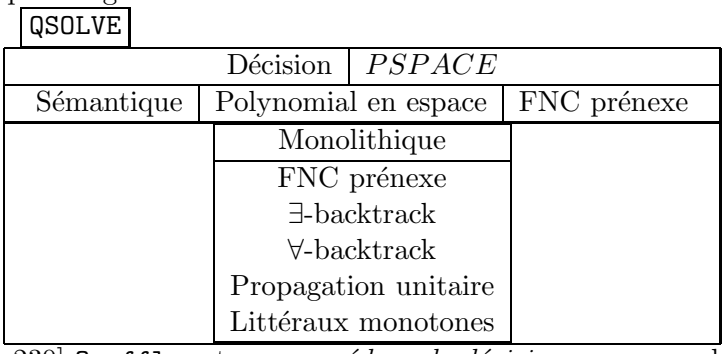


Q-résolution. cf. *résolution.*

`QSAT`. [176] `QSAT` est une *procédure de décision* pour le *problème de validité* des QBF par élimination des *quantificateurs les plus internes* vers les *quantificateurs les plus externes*. Cette procédure opère itérativement sur une formule $Q(\exists x\ (F \wedge G))$ telle que x n'apparaît pas dans F ; l'équivalence $(\exists x\ (F \wedge G)) \equiv (F \wedge (\exists x\ G))$ permet d'isoler la sous formule $(\exists x\ G)$ qui va être mise en *équivalence logique* par une procédure *simp* avec une formule G' ne contenant pas le symbole propositionnel x ; par *substitution des équivalents* $Q(\exists x\ (F \wedge G)) \equiv Q(F \wedge G')$. Le procédé est similaire avec une quantification universelle. Le processus est itéré jusqu'à élimination de tous les quantificateurs. La procédure *simp* a besoin d'une procédure de décision pour le problème SAT et d'une procédure de décision pour le problème TAUT pour être effective. Cette procédure construit une FNC G' sur les symboles propositionnels *libres* de G telle que $G' \equiv (\exists x\ G)$. Elle opère comme un *Davis-Logemann-Loveland* par séparation de l'espace de recherche sur un symbole propositionnel libre y de G par appel récursif sur $[y \leftarrow \top](G)$ et $[y \leftarrow \bot](G)$. Si $[y_1 \leftarrow C_1] \ldots [y_n \leftarrow C_n](G)$ est *insatisfiable* alors $simp((\exists x\ [y_1 \leftarrow C_1] \ldots [y_n \leftarrow C_n](G)))$ retourne la clause $((y_1 \oplus C_1) \vee \ldots \vee (y_n \oplus C_n))$; sinon si $[y_1 \leftarrow C_1] \ldots [y_n \leftarrow C_n](G)$ est une *tautologie* alors $simp((\exists x\ [y_1 \leftarrow C_1] \ldots [y_n \leftarrow C_n](G)))$ retourne $\top$ (attention : ici le symbole propositionnel x a été quantifiée universellement) ; sinon si $(\exists x\ [y_1 \leftarrow C_1] \ldots [y_n \leftarrow C_n](G))$ ne contient plus de symbole propositionnel libre alors soit cette formule est insatisfiable (et ce cas a déjà été traité) soit elle est *satisfiable* et $simp((\exists x\ [y_1 \leftarrow C_1] \ldots [y_n \leftarrow C_n](G)))$ retourne $\top$; sinon un nouveau symbole propositionnel libre y_{n+1} est considéré et $G_\top = simp((\exists x\ [y_1 \leftarrow C_1] \ldots [y_n \leftarrow C_n][y_{n+1} \leftarrow \top](G)))$ et $G_\bot = simp((\exists x\ [y_1 \leftarrow C_1] \ldots [y_n \leftarrow C_n][y_{n+1} \leftarrow \bot](G)))$ sont calculés, $(G_\top \wedge G_\bot)$ est retourné. L'algorithme de la procédure *simp* est basé sur la construction classique de la FNC comme étant la conjonction de la négation des lignes de la table de vérité qui *falsifie* la formule considérées comme des *termes*. Seule l'implantation d'une double restriction de `QSAT` est décrite dans [176] : restrictions à SAT et aux formules sous FNC. Dans le cadre de cette dernière restriction, la recherche d'une forme $Q(\exists x\ (F \wedge G))$ telle que x n'apparaît pas dans G est triviale ; seul le choix de x demeure et permet l'édification de stratégies.



`QSOLVE`. [84] `QSOLVE` est une *procédure de décision* pour le *problème de validité* des QBF *prénexes* dont la *matrice* est sous FNC basée sur un `QDLL` utilisant les structures de données d'une procédure de décision pour le problème SAT [38]. Cette procédure de décision intègre la *validité universelle triviale* et la *falsifiabilité existentielle triviale*, techniques issues de `Evaluate` ainsi qu'une forme d'*inversion des quantificateurs* issue de `decide`. Cette procédure a une extension parallèle `PQSOLVE` [84] et c'est sa principale originalité.



`Quaffle`. [228, 230] `Quaffle` est une *procédure de décision* pour pour le *problème de validité* des QBF *prénexes* dont la *matrice* est sous *FNC* basée sur un `QDLL`, par extension de la procédure de décision `ZChaff` [229] pour le problème SAT. `Quaffle` intègre un *retour-arrière intelligent* basé sur de l'*apprentissage de lemme* pour l'échec suivant le « conflict-driven learning » [228] et le succès suivant le « satisfiability-directed learning »[230]. `Quaffle` manipule en interne une formule sous *FNCA* constituée initialement du problème pour la partie FNC et d'une partie *FND* vide. Le « conflict-driven learning » apprend des *clauses*, grâce à la « long distance resolution » qui est une révision de la *Q-résolution*, pour les rajouter à la partie de la formule FNCA sous FNC tandis que le « satisfiability-directed learning » apprend de manière dual, c'est-à-dire via le *dual de la Q-résolution*, des *cubes* mais pour les rajouter à la partie de la formule FNCA sous FND. La « long distance resolution » autorise l'apprentissage de *clauses tautologiques* qui sont, dans le cadre de la Q-résolution, inutiles. Lorsqu'il n'y a aucun *cube vide* lors d'un succès pour appliquer le dual de la Q-résolution, ce cube initial est généré à partir des littéraux assignés ; l'absence de cube vide lors d'un succès est possible puisque la formule est sous FNCA et que la détection du succès est obtenue par satisfaction de toutes les clauses de la partie FNC ; l'absence de *clause vide* lors d'un échec est impossible puisque la détection de cet échec est réalisée par cette même clause vide. Dans [220], une extension à `Quaffle` est proposée intégrant la *propagation « don't care »*. Dans [190], une restriction morphologique (complète) à `Quaffle` est proposée : la procédure `Duaffle`

qui est spécifique à la programmation de jeux à deux joueurs.

`Quaffle`

Décision	*PSPACE*	
Sémantique	Polynomial en espace	FNC prénexe
	Monolithique	
	FNCA prénexe ∃-learning ∀-learning Propagation unitaire Littéraux monotones	

quantificateur. **1.** QBF. cf. définition 1. **2.** PRED cf. *langage de la logique des prédicats.*

quantificateur existentiel. **1.** QBF. cf. définition 1. **2.** PRED. cf. *langage de la logique des prédicats.*

quantificateur plus interne. cf. définition 6.

quantificateur plus externe. cf. définition 6.

quantificateur universel. **1.** QBF. cf. définition 1. **2.** PRED. cf. *langage de la logique des prédicats.*

`quantor.` [36] `quantor` est une *procédure de décision* pour le *problème de validité* des QBF *prénexes* dont la *matrice* est sous *FNC* qui applique selon des *heuristiques*, après une *réduction universelle* et jusqu'à obtention d'une *formule propositionnelle*, soit la *multi-résolution* sur le *quantificateur le plus interne* qui est alors existentiel (en éliminant les *clauses tautologiques*), soit l'*expansion universelle bottom-up* sur le quantificateur universel le plus interne. La *formule propositionnelle* qui est sous FNC est alors donnée en entrée à une procédure de décision pour le problème SAT. La procédure `quantor` intègre aussi la *propagation des littéraux monotones* et une forme binaire de *raisonnement sur les équivalences* ; elle utilise une gestion fine des clauses, en particulier la gestion des clauses *subsummées* ; elle dispose d'un lieur non linéaire mais qui ne semble pas aussi sophistiqué que celui de la procédure `sKizzo` ; elle n'est pas polynomiale en espace car le fragment FNC n'est pas *linéairement clos* mais seulement *polynomialement clos* pour la multi-résolution [83]. La procédure `quantor` peut être efficace grâce à des heuristiques de choix du symbole propositionnel à éliminer particulièrement fines basées sur le coût de cette élimination.

`quantor`

Décision	*PSPACE*	
Symbolique	Exponentiel en espace	FNC prénexe
	Monolithique	
	FNC ∀-expansion↑ Multi-résolution Littéraux monotones Réduction universelle	

`QUBE.` [103] `QUBE` est une *procédure de décision* pour le *problème de validité*es QBF *prénexes* dont la *matrice* est sous *FNC* basée sur un `QDLL`, par extension de la

procédure de décision `SIM` [100] pour le problème SAT. Le cas d'induction du `QDLL` est précédé d'une phase d'application jusqu'à un point fixe de la *validité universelle triviale*, de la *falsifiabilité existentielle triviale* et de la *falsifiabilité universelle triviale*. Le mécanisme de *retour-arrière* initialement sous forme de *backjumping* (version « BJ » [105]) a été étendu par un mécanisme d'*apprentissage de lemme* [104, 108], la matrice étant sous *FNCA* pour stocker les lemmes portant sur les quantificateurs universels. Depuis [109] `QUBE` n'opère plus sur des QBF réellement prénexes en interne grâce à un *arbre de quantificateurs*. `QUBE++` [107] est une version intermédiaire de `QUBE` qui intègre des structures de données paresseuses [93].

`QUBE`

Décision	$PSPACE$	
Sémantique	Polynomial en espace	FNC prénexe
	Monolithique	
	FNCA $\exists$-learning $\forall$-learning Propagation unitaire Littéraux monotones	

`QuBIS`. [181] `QuBIS` est une *procédure de décision* pour le *problème de validité* des QBF *prénexes* sous *FNC* qui étend la procédure de décision de *Davis-Putnam* basée sur la *multi-résolution* aux QBF. La procédure `QuBIS` peut-être vue comme un préprocesseur puisqu'il est défini comme étant incomplet par la limitation de la taille des Q-résolvantes et du nombre de celles-ci générées par l'élimination d'un symbole propositionnel *existentiellement quantifié*.

`QuBIS`

Décision	$PSPACE$	
Symbolique	Exponentiel en espace	FNC prénexe
	Monolithique	
	FNC prénexe Multi-résolution Réduction universelle	

`QUBOS`. [17] `QUBOS` est une *procédure de décision* pour le *problème de validité* des QBF sous *FNN*, mais qui ne sont pas nécessairement sous forme *prénexe*, qui applique, jusqu'à élimination soit de tous les quantificateurs universels soit de tous les quantificateurs existentiels, l'itération suivante : réduction de la portée des quantificateurs par *miniscoping* (ce miniscoping n'est pas fait uniquement avant l'application de la procédure mais comme un mécanisme à part entière qui permet de considérer cette méthode comme ayant en entrée des QBF non prénexes) ; élimination d'un quantificateur par *expansion* ; simplification selon les *équivalences logiques* propositionnelles basées sur les propriétés de $\top$ et $\bot$: (0.13) à (0.16) ; l'*élimination des littéraux unitaires* ; application de la *propagation de littéraux monotones*. Bien que l'application itérative de ces étapes soit complète, lorsqu'il ne reste plus que des quantificateurs d'un seul type (dans la forme prénexe), `QUBOS` fait appel à une procédure pour

le problème SAT sous *FNC* soit directement s'il ne reste plus que des quantificateurs existentiels, soit par une négation de la formule (les quantificateurs universels deviennent alors des quantificateurs existentiels par l'équivalence logique (1.3), et l'*insatisfiabilité* est alors recherchée).

QUBOS

Décision	$PSPACE$	
Symbolique	Exponentiel en espace	FNN

Monolithique

FNN
∃-expansion↑
∀-expansion↑
Simplification ⊤ et ⊥
Elimination des littéraux unitaires
Littéraux monotones

`QUIP`. Certains raisonnements propositionnels *non-monotones* ont une complexité dans $PSPACE$ mais au-delà de $\mathcal{NP}$; ainsi un problème soumis à de tels mécanismes de raisonnement peut être réduit polynomialement en un *problème de validité* d'une QBF. QUIP est un ensemble d'outils d'inférence pour différents raisonnements non-monotones basé sur ce principe ; les raisonnements non-monotones traités sont les suivants : l'*abduction* [201] dans [77, 76], la *logique des défauts* [184] dans [77, 76], la *logique autoépistémique* [150] et autres logiques modales dans [77, 76, 82] et enfin, la *programmation logique disjonctive* [91, 179] et autres raisonnements non-monotones dans [75, 173] dans une approche inverse à celle présentée au chapitre 5. Chaque outil à la structure ternaire suivante : un filtre pour lire le problème dans une des logiques non-monotones et le traduire en une QBF ; l'appel à une procédure de décision pour le problème de validité des QBF ; une interprétation du résultat en les termes du problème d'entrée. L'approche offre des ensembles de tests pour les premiers niveaux de la *hiérarchie polynomiale*.

raisonnement sur équivalence. **1.** QBF. [36] Le *raisonnement sur équivalence* est une transformation pour QBF *prénexes* FNC qui, pour un couple de *clauses* $(\neg x \vee y)$ et $(x \vee \neg y)$ tel que $y \prec x$ et le symbole propositionnel x est *existentiellement quantifié* substitue dans la formule x par y. (Cette transformation ne préserve que les *modèles* propositionnels ; $(\forall y\ (\exists x\ ((\neg x \vee y) \wedge (x \vee \neg y)))) \equiv (\forall y\ [x \leftarrow y](((\neg x \vee y) \wedge (x \vee \neg y)))) \equiv (\forall y\ \top) \equiv \top$ mais $(\exists y\ (\forall x\ ((\neg y \vee x) \wedge (y \vee \neg x)))) \equiv \bot \not\equiv (\exists y\ [x \leftarrow y](((\neg y \vee x) \wedge (y \vee \neg x)))) \equiv \top$). **2.** PROP. [19] Le *raisonnement sur équivalence* est une transformation pour le fragment FNC conservant la *satisfiabilité*, qui, pour un couple de *clauses* $(\neg x \vee y)$ et $(x \vee \neg y)$ substitue dans la formule x par y et élimine alors toutes les clauses tautologiques.

réduction existentielle. cf. `Quaffle`.

réduction universelle. La *réduction universelle* est une transformation initialement introduite dans [120] pour les QBF sous FNC qui préserve la *validité* : dans une *clause*, les *symboles propositionnels universellement quantifiés* qui ont des *quantificateurs plus internes* que tout autre symbole propositionnel *existentiellement quantifié* peuvent être ôtes. Cette propriété est basée sur l'*équivalence logique* suivante, en appliquant la *sémantique du quantificateur* universel :

$$\begin{array}{ll} & Q\forall x(x_1\vee\ldots\vee x_n\vee x) \\ \equiv & Q([x\leftarrow\top](x_1\vee\ldots\vee x_n\vee x)\wedge[x\leftarrow\bot](x_1\vee\ldots\vee x_n\vee x)) \\ \overset{0.14;0.15;0.13}{\equiv} & Q(x_1\vee\ldots\vee x_n). \end{array}$$

La *réduction universelle* est étendue au cas FNN [112] : si les arguments d'un connecteur sont universellement quantifiés (ou existentiellement quantifiés mais déjà déterminés) et la *valeur de vérité* de ce connecteur déjà déterminée alors il y a contradiction ; dans une sous-formule constituée uniquement de conjonctions ou de disjonctions, si un argument existentiellement quantifié non déterminé a un *quantificateur plus externe* que les arguments universellement quantifiés alors si la valeur de vérité de la sous-formule est à **faux** et le connecteur est une conjonction (resp. **vrai** et disjonction) la valeur de vérité du symbole propositionnel existentiellement quantifié est nécessairement **faux** (resp. **vrai**).

règle. **1.** QBF. cf. définition 43. **2.** cf. *programmation logique normale.*

règle de la coupure. cf. *calcul des séquents.*

règle d'élimination de fusion de classes. cf. définition 27.

règle d'élimination des doubles négations. cf. définition 27.

règle d'élimination du quantificateur existentiel. cf. définition 27.

règle d'élimination du quantificateur universel. cf. définition 27.

relation d'équivalence. Une *relation d'équivalence* est une relation *réflexive* (tout élément est en relation avec lui-même), *symétrique* (si un élément e est en relation avec e' alors e' est en relation avec e) et *transitive* (si e est en relation avec e' et e' est en relation avec e'' alors e est en relation avec e'').

relation de formules. cf. définition 25.

relation de formules explicitement contradictoire. cf. définition 26.

relation d'ordre associée à une QBF. cf. définition 6.

résolution. **1.** QBF. La *Q-résolution* [120] est une *procédure de décision* pour les QBF *prénexes* dont la *matrice* est sous FNC qui combine la *résolution propositionnelle* (cf. **2.** PROP) à la *réduction universelle* et qui termine sur une (Q-)résolvante vide si et seulement si la QBF est *non-valide*. Le *dual de la Q-résolution* est une procédure de décision pour les QBF sous FND prénexes qui combine le *consensus*, qui est le dual de la résolution en cela qu'il agit par élimination des *littéraux complémentaires* de deux *cubes*, et la *réduction existentielle*, qui est le dual de la *réduction universelle* en cela qu'elle élimine les symboles propositionnels existentiellement quantifiés d'un cube qui ne contient pas de symboles propositionnels universellement quantifiés plus internes. La procédure `QKN` est une implémentation de la Q-résolution.

QKN

Décision	$PSPACE$	
Symbolique	Exponentiel en espace	FNC prénexe
	Monolithique	
	FNC prénexe Résolution Réduction universelle	

2. PROP. La *résolution propositionnelle* est une *procédure de décision* pour les *formules propositionnelles* sous FNC qui *décide* de l'*insatisfiabilité* par obtention d'une *clause vide* d'un ensemble de *clauses* selon l'*équivalence logique*

$$((A \vee B) \wedge (\neg A \vee C) \wedge \mathcal{C}) \equiv ((B \vee C) \wedge (A \vee B) \wedge (\neg A \vee C) \wedge \mathcal{C})$$

(la formule $(B \vee C)$ est la *résolvante* de l'application de l'étape de résolution sur les clauses $(A \vee B)$ et $(\neg A \vee C)$; l'enchaînement des étapes de résolution qui mène à la clause vide est une *réfutation*). **3.** PRED. La *résolution de Robinson* [189] est basée sur la *résolution propositionnelle* (cf. **2.** PROP). La formule initiale est sous *forme clausale* (une conjonction de *clauses*). L'identité des *littéraux* qui disparaissent est réalisée par l'*unification* qui instancie *a minima* les *variables* (logiques). Un *unificateur* pour deux littéraux est une *substitution* (une fonction des variables dans les *termes*), qui a pour objectif de les rendre, une fois appliquée, égaux. Si deux termes sont unifiables alors il existe nécessairement un *unificateur le plus général*, substitution qui instancie le moins possible les deux termes, unique au renommage des variables près. Un *test d'occurrence* qui vérifie qu'une variable ne figure pas dans le terme avec lequel elle s'unifie est indispensable pour la préservation de la *correction*. La résolution de Robinson est à la naissance du paradigme de la *programmation logique*. **4.** cf. *multi-résolution* et *hyper résolution binaire*.

retour-arrière. Le *retour-arrière* (en. *backtracking*) est une technique pour simuler le non-déterminisme (choisir sans la garantie de faire un choix pertinent pour la suite) dans une procédure : le choix, sur lequel la procédure pourra revenir, ainsi que son contexte sont conservés dans une *pile de retour-arrière* (en. *backtrack stack*). L'utilisation de la pile de retour-arrière revient à un parcours d'arbre dans lequel les nœuds sont les points de choix. Cette technique est à la base de l'implantation des algorithmes de recherche (cf. § 1.5.1). Le *retour-arrière intelligent* (en. *intelligent backtracking* ou *backjumping* ou *dependency-directed backtracking*) en QBF est une technique issue de la résolution du problème SAT sous FNC [144] (et avant cela, du domaine des *problèmes de satisfaction de contraintes* [178]) qui modifie le retour-arrière en éliminant la seconde branche d'un *arbre sémantique* d'un nœud dont le symbole propositionnel n'a pas participé à l'*insatisfiabilité* de la première branche (i.e. en modifier sa *valeur de vérité* ne changera pas l'insatisfiabilité). (L'ensemble des symboles propositionnels tels qu'en en changeant d'un la valeur de vérité, la formule passe d'insatisfiable à satisfiable et inversement est appelé dans [105, 101] *raison* (en. *reason*) de l'insatisfiabilité ou de la satisfiabilité.) En cela cette technique s'apparente à celle de l'*apprentissage de lemme*. Elle s'applique de la même

façon pour les symboles propositionnels quantifiées existentiellement pour les QBF sous FNC que pour SAT sous FNC (c'est le *conflict-directed backjumping* [105, 101] ou *dependency-directed backtracking for false subproblem* [130]). Deux variantes de retour-arrière intelligent lors de la détection de l'insatisfiabilité : la plus simple mais aussi la moins puissante consiste à déclarer un symbole propositionnel pertinent pour le retour-arrière (en. *relevant*) s'il a participé à *falsifier* une *clause*, c'est la technique de l'étiquetage (en. *labeling*) ; la plus coûteuse est la technique des ensembles de symboles pertinents (en. *relevance sets*). Cette dernière technique associe à chaque nœud de l'arbre sémantique un ensemble de symboles pertinents : un nœud qui présente une clause falsifiée affecte à son ensemble de symboles pertinents les symboles de cette clause ; si pour un nœud interne le symbole propositionnel de branchement n'est pas présent dans l'ensemble de symboles pertinents de son premier fils alors la seconde branche est considérée comme insatisfiable, sinon la seconde branche est calculée et l'ensemble de symboles pertinents de ce second fils l'est aussi, si le symbole n'appartient pas à cet ensemble uniquement ce dernier est affecté à l'ensemble des symboles pertinents de ce nœud interne sinon c'est l'union des deux ensembles qui lui est affecté. Par la dualité des paires quantificateur existentiel/insatisfiabilité et quantificateur universel/*validité*, le retour-arrière intelligent pour la seconde paire est naturel et s'exprime par la symétrie des théorèmes de [105, 101] : À la clause qui falsifie la formule répond le *modèle*, mais si la première est suffisante, le second contient trop d'éléments et rend la méthode moins opérante. Pour que le principe du retour-arrière intelligent soit parfaitement dual, il faut que cette dualité s'étende aussi au fragment logique choisi, or le dual de la forme normale conjonctive est la *forme normale disjonctive*. Il est tout de même possible de considérer une version appauvrie pour la FNC qui revient à faire de la *validité universelle triviale* à la volée (*dependency-directed backtracking by model intersections* [130]) : une seconde branche portant sur un symbole universellement quantifié n'est pas considérée si toutes les clauses non satisfaites du nœud de l'arbre sémantique sont satisfaites par l'intersection des modèles de la première branche. Il existe une version plus évoluée mais non-déterministe (*dependency-directed backtracking by model unions* [130] ou *solution-directed backjumping* [105, 101]) qui recherche un modèle plus petit. Enfin, dans [79, 80] une version parfaitement symétrique est proposée dans le cadre des QBF sous *forme normale négative* et insérée dans la procédure de décision `qpro` et son système $GQBF$. Le retour-arrière intelligent y est explicité comme une propriété d'inversion de l'ordre des quantificateurs d'une QBF sans en affecter la validité.

retour-arrière intelligent. cf. *retour-arrière*.

sampling. cf. *échantillonnage*.

SAT. cf. *sémantique de la logique propositionnelle*.

sat-certificat. cf. § 1.5.3.

satisfiabilité. **1.** QBF. cf. définition 22. **2.** PROP cf. *sémantique de la logique proposi-*

tionnelle. **3.** PRED cf. *sémantique de la logique des prédicats*.

satisfiability-directed learning. cf. *apprentissage de lemme*.

scénario. cf. définition 21.

schéma. cf. définition 43.

semantic tree. cf. *arbre sémantique*.

sémantique de la logique des prédicats. Nous présentons la *sémantique* du *langage de la logique des prédicats* [86] en détail car la sémantique des QBF s'en inspire et peut y être ramenée en se restreignant à son interprétation aux *booléens*. La sémantique de la logique des prédicats nécessite pour le langage non-logique $\mathcal{L}$ de lui associer une *fonction d'interprétation* dans un *domaine sémantique* via une *structure* $\mathcal{S} = (\mathcal{D}, I)$ telle que $\mathcal{D}$ soit un ensemble (le domaine sémantique) et I une fonction d'interprétation qui associe à chaque *symbole de constante*, *symbole de fonction* ou *symbole de prédicat* une fonction comme suit : pour chaque symbole de fonction f avec n arguments, $n \geq 1$, $I(f)$ est une fonction de $\mathcal{D}^n$ dans $\mathcal{D}$; pour chaque symbole de constante c, $I(c)$ est un élément de $\mathcal{D}$; pour chaque symbole de prédicat p avec n arguments, $n \geq 1$, $I(p)$ est une fonction de $\mathcal{D}^n$ dans **BOOL** ; pour chaque symbole de prédicat p sans argument, $I(p)$ est un élément de **BOOL**.

Le calcul de la sémantique d'une formule du langage des prédicats fait apparaître lors de l'élimination des quantificateurs des *variables libres* qui doivent recevoir une valeur du domaine sémantique via une *fonction d'assignation* (ou *assignation*) définie ainsi : une assignation α est une fonction de l'ensemble des variables dans l'ensemble $\mathcal{D}$. Soit d un élément de $\mathcal{D}$ et x une variable, $\alpha[x := d]$ dénote une nouvelle assignation de l'ensemble des variables dans $\mathcal{D}$ telle que :

$$\begin{cases} \alpha[x := d](y) & = & d \text{ si } x = y \\ & = & \alpha(y) \text{ sinon.} \end{cases}$$

Ainsi pourvu nous pouvons définir la sémantique d'un *terme* t comme étant une fonction $I^*(t) : (\mathcal{V} \to \mathcal{D}) \to \mathcal{D}$ définie inductivement, pour toute assignation α par :

$$\begin{array}{rcll} I^*(x)(\alpha) & = & \alpha(x) & \text{pour tout } x \in \mathcal{V} \\ I^*(c)(\alpha) & = & I(c) & \text{pour tout } c \in \mathcal{SC} \\ I^*(ft_1 \ldots t_n)(\alpha) & = & I(f)(I^*(t_1)(\alpha) \ldots I^*(t_n)(\alpha)) & \text{pour tout } f \in \mathcal{SF}, \\ & & \text{d'arité } n \geq 1, \text{ et } t_1, \ldots, t_n \text{ des termes du langage } \mathcal{L}. & \end{array}$$

Par extension de la *sémantique de la logique propositionnelle*, à chaque *connecteur logique* et *quantificateur* est associée une fonction à valeur dans **BOOL** qui en définit sa sémantique :

$$I_\top, I_\bot : (\mathcal{V} \to \mathcal{D}) \to \mathbf{BOOL}$$

$$I_\bot(\alpha) = i_\bot \quad I_\top(\alpha) = i_\top$$

$$I_\neg : ((\mathcal{V} \to \mathcal{D}) \to \mathbf{BOOL}) \times (\mathcal{V} \to \mathcal{D}) \to \mathbf{BOOL}$$

$$I_\neg(f)(\alpha) = i_\neg(f(\alpha))$$

$$I_\circ : ((\mathcal{V} \to \mathcal{D}) \to \mathbf{BOOL})^2 \times (\mathcal{V} \to \mathcal{D}) \to \mathbf{BOOL}$$

$$I_\circ(f,g)(\alpha) = i_\circ(f(\alpha), g(\alpha)) \text{ pour tout connecteur } \circ \in \{\wedge, \vee, \to, \leftrightarrow, \oplus\}$$

$$I^x_\exists, I^x_\forall : ((\mathcal{V} \to \mathcal{D}) \to \mathbf{BOOL}) \times (\mathcal{V} \to \mathcal{D}) \to \mathbf{BOOL}$$

$I^x_\exists(f)(\alpha) = \mathbf{vrai}$ si et seulement si $f(\alpha[x := d]) = \mathbf{vrai}$ pour un $d \in \mathcal{D}$.
$I^x_\forall(f)(\alpha) = \mathbf{faux}$ si et seulement si $f(\alpha[x := d]) = \mathbf{faux}$ pour un $d \in \mathcal{D}$.

La sémantique d'une formule F d'un langage $\mathcal{L}$ dans une structure $\mathcal{S} = (\mathcal{D}, I)$ est une fonction $I^*(F) : (\mathcal{V} \to \mathcal{D}) \to \mathbf{BOOL}$ définie inductivement, pour toute assignation α par :

$$\begin{array}{rcl}
I^*(\bot)(\alpha) & = & I_\bot(\alpha) \\
I^*(\top)(\alpha) & = & I_\top(\alpha) \\
I^*(p)(\alpha) & = & I(p) \\
 & & \text{pour tout } p \in \mathcal{SP} \text{ d'arité } 0 \\
I^*((G \circ H))(\alpha) & = & I_\circ(I^*(G), I^*(H))(\alpha) \\
 & & \text{pour tout } G, H \in \mathbf{PRED}_\mathcal{L} \text{ et } \circ \in \{\wedge, \vee, \to, \leftrightarrow, \oplus\} \\
I^*(\neg G)(\alpha) & = & I_\neg(I^*(G))(\alpha) \\
 & & \text{pour tout } G \in \mathbf{PRED}_\mathcal{L} \\
I^*(pt_1 \ldots t_n)(\alpha) & = & I(p)(I^*(t_1)(\alpha) \ldots I^*(t_n)(\alpha)) \\
 & & \text{pour tout } p \in \mathcal{SP} \text{ d'arité } n \geq 1, \text{ et } t_1, \ldots, t_n \text{ des termes} \\
I^*((\exists x\ G))(\alpha) & = & I^x_\exists(I^*(G))(\alpha) \\
 & & \text{pour tout } G \in \mathbf{PRED}_\mathcal{L} \text{ et } x \in \mathcal{V} \\
I^*((\forall x\ G))(\alpha) & = & I^x_\forall(I^*(G))(\alpha) \\
 & & \text{pour tout } G \in \mathbf{PRED}_\mathcal{L} \text{ et } x \in \mathcal{V}
\end{array}$$

Soient un langage $\mathcal{L}$ et F une formule de $\mathcal{L}$: la structure $\mathcal{S} = (\mathcal{D}, I)$ *satisfait* la formule F pour l'assignation α si $I^*(F)(\alpha) = \mathbf{vrai}$, ce qui est noté $(\mathcal{S} \models^\alpha F)$; la structure $\mathcal{S} = (\mathcal{D}, I)$ *falsifie* la formule F pour l'assignation α si $I^*(F)(\alpha) = \mathbf{faux}$, ce qui est noté $(\mathcal{S} \not\models^\alpha F)$; la formule F est *satisfiable* s'il existe une structure $\mathcal{S} = (\mathcal{D}, I)$ et une assignation α telles que $I^*(F)(\alpha) = \mathbf{vrai}$; la formule F est *valide dans la structure* $\mathcal{S} = (\mathcal{D}, I)$ si pour toute assignation α, $I^*(F)(\alpha) = \mathbf{vrai}$, la structure $\mathcal{S}$ est alors un *modèle* pour la formule F, ce qui est noté $(\mathcal{S} \models F)$; la formule F est falsifiable dans la structure $\mathcal{S}$ s'il existe une assignation α telle que $I^*(F)(\alpha) = \mathbf{faux}$, ce qui est noté $(\mathcal{S} \not\models F)$; la formule F est *valide* si pour toute structure $\mathcal{S} = (\mathcal{D}, I)$ et pour toute assignation α, $I^*(F)(\alpha) = \mathbf{vrai}$, ce qui est noté $\models F$; la formule F est *insatisfiable dans la structure* $\mathcal{S}$ si pour toute assignation α, $I^*(F)(\alpha) = \mathbf{faux}$; la formule F est *insatisfiable* si pour toute structure $\mathcal{S}$ et pour toute assignation α , $I^*(F)(\alpha) = \mathbf{faux}$.

sémantique de la logique du premier ordre. cf. *sémantique de la logique des prédicats.*

sémantique de la logique propositionnelle. La *sémantique de la logique propositionnelle* [86] définit la *valeur de vérité*, c'est-à-dire un *booléen* d'une *formule propositionnelle* selon une *valuation*, i.e. une fonction qui associe à chaque *symbole propositionnel* une valeur de vérité. A chaque *connecteur logique* est associé une fonction à valeur dans **BOOL** qui en définit sa sémantique.

$$i_\top, i_\bot :\rightarrow \mathbf{BOOL}$$

$$i_\top = \mathbf{vrai} \qquad i_\bot = \mathbf{faux}$$

$$i_\neg : \mathbf{BOOL} \rightarrow \mathbf{BOOL}$$

x	$i_\neg(x)$
vrai	**faux**
faux	**vrai**

$$i_\wedge, i_\vee, i_\rightarrow, i_\leftrightarrow, i_\oplus : \mathbf{BOOL} \times \mathbf{BOOL} \rightarrow \mathbf{BOOL}$$

x	y	$i_\wedge(x,y)$	$i_\vee(x,y)$	$i_\rightarrow(x,y)$	$i_\leftrightarrow(x,y)$	$i_\oplus(x,y)$
vrai	**vrai**	**vrai**	**vrai**	**vrai**	**vrai**	**faux**
vrai	**faux**	**faux**	**vrai**	**faux**	**faux**	**vrai**
faux	**vrai**	**faux**	**vrai**	**vrai**	**faux**	**vrai**
faux	**faux**	**faux**	**faux**	**vrai**	**vrai**	**faux**

$$I_\top, I_\bot : (\mathcal{SP} \rightarrow \mathbf{BOOL}) \rightarrow \mathbf{BOOL}$$

$$I_\bot(v) = i_\bot \quad I_\top(v) = i_\top$$

$$I_\neg : ((\mathcal{SP} \rightarrow \mathbf{BOOL}) \rightarrow \mathbf{BOOL}) \times (\mathcal{SP} \rightarrow \mathbf{BOOL}) \rightarrow \mathbf{BOOL}$$

$$I_\neg(f)(v) = i_\neg(f(v))$$

$$I_\circ : ((\mathcal{SP} \rightarrow \mathbf{BOOL}) \rightarrow \mathbf{BOOL})^2 \times (\mathcal{SP} \rightarrow \mathbf{BOOL}) \rightarrow \mathbf{BOOL}$$

$$I_\circ(f,g)(v) = i_\circ(f(v), g(v)) \text{ pour tout connecteur } \circ \in \{\wedge, \vee, \rightarrow, \leftrightarrow, \oplus\}$$

La sémantique d'une formule propositionnelle F est une fonction $I^*(F) : (\mathcal{SP} \rightarrow \mathbf{BOOL}) \rightarrow \mathbf{BOOL}$ définie inductivement par :

$$\begin{array}{rcll} I^*(\bot)(v) & = & I_\bot(v) & \\ I^*(\top)(v) & = & I_\top(v) & \\ I^*(p)(v) & = & v(p) & \text{pour tout } p \in \mathcal{SP} \\ I^*(\neg G)(v) & = & I_\neg(I^*(G))(v) & \text{pour tout } G \in \mathbf{PROP} \\ I^*((G \circ H))(v) & = & I_\circ(I^*(G), I^*(H))(v) & \\ & & \text{pour tout } G, H \in \mathbf{PROP} \text{ et } \circ \in \{\wedge, \vee, \rightarrow, \leftrightarrow, \oplus\} & \end{array}$$

Cette version de la sémantique est non conventionnelle car elle fait apparaître un niveau intermédiaire (celui des fonctions $I_\top$, $I_\bot$, $I_\neg$ et $I_\circ$) qui se révèle inutile au propositionnel (puisque l'argument, la valuation, ne change jamais) mais très utile au niveau du premier ordre et des QBF. Par application de cette remarque, une définition plus classique peut-être déduite (avec $v^*(F) = I^*(F)(v)$) :

$$\begin{array}{rcll} v^*(\bot) & = & i_\bot & \\ v^*(\top) & = & i_\top & \\ v^*(p) & = & v(p) & \text{pour tout } p \in \mathcal{SP} \\ v^*(\neg G) & = & i_\neg(v^*(G)) & \text{pour tout } G \in \mathbf{PROP} \\ v^*((G \circ H)) & = & i_\circ(v^*(G), v^*(H)) & \\ & & \text{pour tout } G, H \in \mathbf{PROP} \text{ et } \circ \in \{\wedge, \vee, \rightarrow, \leftrightarrow, \oplus\} & \end{array}$$

L'extension de la valuation v pour une formule F, $v^*(F)$, est l'*interprétation booléenne* de F par v. Une valuation v *satisfait* une formule F si $v^*(F) = \mathbf{vrai}$, cette valuation est alors un *modèle* de F, ce qui est noté $v \models F$. une formule est *satisfiable* si elle admet (au moins) un modèle ; une valuation v *falsifie* une formule F si $v^*(F) = \mathbf{faux}$, ce qui est noté $v \not\models F$; une formule est *falsifiable* si elle admet (au moins) une valuation qui la falsifie ; une formule F est une *tautologie* si toute valuation en est un modèle, ce qui est noté $\models F$ (et dans le cas contraire $\not\models F$) ; une formule est *insatisfiable* si aucune valuation n'en est un modèle ; une formule A est *conséquence sémantique* d'un ensemble de formules Σ si pour tout modèle de l'ensemble de formules Σ, il est aussi modèle de la formule A, ce qui est noté $\Sigma \models A$. Le problème de *décider* si une formule propositionnelle est satisfiable ou non, problème « SAT », est un problème canonique pour la classe de complexité $\mathcal{NP}$-complet [53]. Le problème de décider si une formule propositionnelle est une tautologie ou non, problème « $TAUT$ », est un problème canonique pour la classe de complexité $co - \mathcal{NP}$-complet.

sémantique des QBF. cf. définition 15 et 20.

sémantique des connecteurs et quantificateurs. cf. définition 14 et 19.

sémantique des modèles stables. La *sémantique des modèles stables* [91] (en. *stable model semantics*) d'un *programme logique normal* est un ensemble d'*atomes* complètement instanciés X tel que le *plus petit modèle de Herbrand* du *réduit* du programme par X est égal à X. Le réduit d'un programme P par un ensemble X est le *programme logique défini* $P^X = \{r^+ \mid r \in P,\ corps^-(r) \cap X = \emptyset\}$.

sémantique propositionnelle des connecteurs logiques. cf. définition 13.

`Semprop`**.** [130] `Semprop` est une *procédure de décision* pour le *problème de validité* des QBF *prénexes* dont la *matrice* est sous FNC basée sur un `QDLL`. Le mécanisme de *retour-arrière* est étendu par du *dependency-directed backtraking* instancié en du *dependency-directed backtraking for false subproblems* pour le *quantificateur existentiel* et du *dependency-directed backtraking for true subproblems* pour le *quantificateur universel* ainsi que de l'*apprentissage de lemme* instancié en du *caching lemma* pour le quantificateur existentiel et adapté au quantificateur universel en du *caching model*, toutes ces techniques participant au *retour-arrière intelligent*.

Semprop		
Décision	*PSPACE*	
Sémantique	Polynomial en espace	FNC prénexe
	Monolithique	
	FNC prénexe ∃-learning ∀-learning Propagation unitaire Littéraux monotones	

séquent. cf. *calcul des séquents*.

`sKizzo`. [23, 27, 24] `sKizzo` est une *procédure de décision* pour le *problème de validité* des QBF *prénexes* dont la *matrice* est sous FNC basée sur la *skolémisation propositionnelle* mise en série avec une procédure de décision pour le problème SAT. Malheureusement, la formule sous FNC est dans la plupart du temps trop grande pour être même générée. Une représentation dite symbolique, version compactée de la formule, est obtenue avant la génération finale et des techniques de raisonnement symbolique (non nécessairement complètes) sont appliquées dessus jusqu'à ce que la génération finale soit possible. Ces techniques sur la représentation symbolique, qui reprennent des techniques classiques sur les *formules propositionnelles* sous FNC, sont la *propagation de clause unitaire* symbolique, la *propagation de littéral monotone* symbolique, l'*hyper résolution binaire* symbolique et le *raisonnement sur équivalence* symbolique. Depuis [26] `sKizzo` n'opère plus sur des QBF réellement prénexes en interne grâce à un *arbre de quantificateurs* obtenu par un algorithme de *miniscoping*. La procédure `sKizzo` est associée à un générateur/vérificateur/inspecteur de **sat**-*certificats*, le système `ozziks` [25].

sKizzo		
Décision	*PSPACE*	
Symbolique	Exponentiel en espace	FNC prénexe
	Transformation	
	Skolémisation	
	SAT	

skolémisation. La skolémisation est une transformation de la *logique des prédicats* qui élimine dans une formule prénexe tout quantificateur existentiel (resp. universel) et remplace dans la matrice la variable quantifiée par un terme constitué d'un nouveau symbole de fonction et des variables universellement (resp. existentiellement) quantifiées qui la précède. La formule obtenue par skolémisation des quantificateurs existentielles est insatisfiable si et seulement si la formule initiale l'est aussi. Ce théorème est à la base de l'application de la *résolution de Robinson*. La skolémisation des quantificateurs universels est utilisée de manière symétrique pour la construction de preuves.

skolémisation propositionnelle. [23] La *skolémisation propositionnelle* prolonge la *skolémisation* par élimination des *quantificateurs existentiels* d'une QBF en « aplatissant » les fonctions *booléennes* pour qu'elles deviennent des *formules propositionnelles*

venant se substituer aux symboles propositionnels *existentiellement quantifiés* de la *matrice* : à chaque combinaison sur les *valeurs de vérité* des symboles propositionnels *universellement quantifiés* précédant dans le *lieur* le symbole propositionnel existentiellement quantifié et à chaque *symbole de fonction* est associé une nouveau symbole propositionnel existentiellement quantifié. Une fois la *substitution* effectuée la matrice est à nouveau une formule propositionnelle qui est alors mise sous FNC. Ces nouveaux quantificateurs et symboles propositionnels associés sont tous introduits avant les quantificateurs universels ainsi le nouveau lieur ne contient plus qu'un préfixe de quantificateurs existentiels (les nouveaux symboles propositionnels) et un suffixe de quantificateurs universels (les symboles propositionnels universellement quantifiés de la QBF initiale). La *réduction universelle* est alors appliquée et élimine tous les quantificateurs universels. L'application de la skolémisation propositionnelle fait croître dans le pire des cas exponentiellement la taille de formule propositionnelle par rapport à la QBF initiale. Les *modèles* QBF sont fournis explicitement par le calcul des modèles propositionnels. La procédure de décision (et logiciel) `sKizzo` [23] a pour principe de base la skolémisation propositionnelle.

solution directed backjumping. cf. *retour-arrière intelligent.*

solution learning. cf. *apprentissage de lemme.*

SOP. cf. *forme normale.*

sous-formule d'une QBF. cf. définition 3.

sum of products. cf. *forme normale.*

$\mathcal{SP}$**.** cf. définition 1.

$\mathcal{SPL}$**.** cf. définition 4.

$\mathcal{SPB}$**.** cf. définition 4.

$\mathcal{SP}_{\exists}$**.** cf. définition 3.

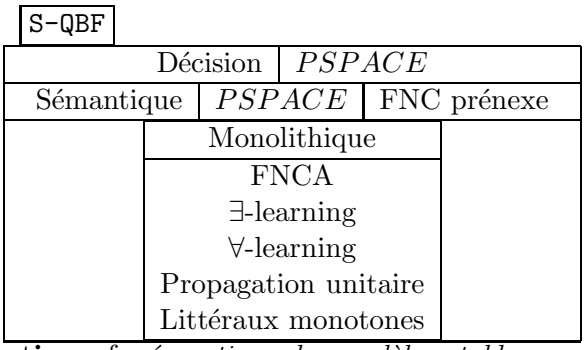
$\mathcal{SP}_{\forall}$**.** cf. définition 3.

sf**.** cf. définition 3.

sfe**.** cf. définition 3.

sfc**.** cf. définition 3.

`S-QBF` [192] `S-QBF` est une *procédure de décision* pour le *problème de validité* des QBF *prénexes* dont la *matrice* est sous FNC basée sur un `QDLL`. La procédure `S-QBF` fait appel à une procédure de décision pour le problème SAT pour déterminer si la matrice est *satisfiable* ou non, si ce n'est pas le cas le *retour-arrière* est déclenché sinon le *modèle* propositionnel obtenu et les informations qui lui sont liées (les *clauses unitaires*, les *lemmes*) sont utilisées dans la partie QBF de la procédure pour guider la recherche.



stable model semantics. cf. *sémantique des modèles stables.*

Stålmarck. *algorithme de Stålmarck.*

stratégie. cf. *politique.*

subsumer. cf. *clause.*

substitution. **1.** QBF. cf. définition 9. **2.** PROP. Une *fonction de substitution* est une fonction de l'ensemble des symboles propositionnels $\mathcal{SP}$ dans l'ensemble des formules propositionnelles, **PROP**. L'*ensemble support* d'une fonction de substitution σ est l'ensemble des symboles propositionnels p tels que $\sigma(p) \neq p$. Soit $\{p_1, \ldots, p_n\}$ l'ensemble support d'une fonction de substitution σ alors la fonction σ sera notée $[p_1 \leftarrow \sigma(p_1), \ldots, p_n \leftarrow \sigma(p_n)]$. Par abus de langage la fonction de substitution (ou *substitution*) dénote aussi son extension à toute formule propositionnelle ($G_1, \ldots, G_n$ un ensemble de formules propositionnelles) :

$$\begin{array}{rcll}
\sigma(\bot) & = & \bot & \\
\sigma(\top) & = & \top & \\
\sigma(p) & = & p & \text{si } \sigma = [p_1 \leftarrow G_1, \ldots, p_n \leftarrow G_n] \\
 & & & \text{et } p \in \mathcal{SP}, p \notin \{p_1, \ldots, p_n\} \\
\sigma(p_i) & = & G_i & \text{si } \sigma = [p_1 \leftarrow G_1, \ldots, p_n \leftarrow G_n] \\
 & & & \text{et } p_i \in \{p_1, \ldots, p_n\} \\
\sigma(\neg G) & = & \neg\sigma(G) & \text{si } G \in \mathbf{PROP} \\
\sigma((G \circ H)) & = & (\sigma(G) \circ \sigma(H)) & \\
\multicolumn{4}{c}{\text{si } G, H \in \mathbf{PROP} \text{ et } \circ \in \{\wedge, \vee, \rightarrow, \leftrightarrow, \oplus\}}
\end{array}$$

3. PRED. Soit un langage non-logique $\mathcal{L}$. Une *fonction de substitution* σ est une fonction des variables dans les termes. L'ensemble support d'une fonction de substitution σ est l'ensemble des variables x telles que $\sigma(x) \neq x$. Soit $\{x_1, \ldots, x_n\}$ l'ensemble support d'une fonction de substitution σ alors la fonction de substitution σ sera notée $[x_1 \leftarrow \sigma(x_1), \ldots, x_n \leftarrow \sigma(x_n)]$. La substitution de support vide est notée ϵ. Par abus de langage la fonction de substitution (ou *substitution*) dénote aussi son extension à tout terme ($\sigma = [x_1 \leftarrow t_1, \ldots, x_n \leftarrow t_n]$, $t_1, \ldots, t_n$ des termes, $\{x_1, \ldots, x_n\} \subset \mathcal{V}$) :

$$\begin{array}{rcll}
\sigma(x) & = & x & \text{si } x \in \mathcal{V} \text{ avec } x \notin \{x_1, \ldots, x_n\} \\
\sigma(x_i) & = & t_i & \text{si } x_i \in \{x_1, \ldots, x_n\} \\
\sigma(c) & = & c & \text{si } c \in \mathcal{SC} \\
\sigma(f t'_1 \ldots t'_m) & = & f\sigma(t'_1)\ldots\sigma(t'_m) & \text{si } f \in \mathcal{SF}, \text{ d'arité non nulle et} \\
 & & & t'_1, \ldots, t'_m \text{ des termes.}
\end{array}$$

et à toute formule
(avec $\sigma' = [x_1 \leftarrow t_1, \ldots, x_{i-1} \leftarrow t_{i-1}, x_{i+1} \leftarrow t_{i+1}, \ldots, x_n \leftarrow t_n]$) :

$$
\begin{array}{rcll}
\sigma(\bot) & = & \bot. & \\
\sigma(\top) & = & \top & \\
\sigma(p) & = & p & \text{si } p \in \mathcal{SP},\ \text{d'arité nulle} \\
\sigma(pt'_1 \ldots t'_m) & = & p\sigma(t'_1)\ldots\sigma(t'_m) & \\
\multicolumn{4}{l}{\text{si } p \in \mathcal{SP},\ \text{d'arité non nulle et } t'_1, \ldots, t'_m \ \text{ des termes}} \\
\sigma(\neg G) & = & \neg\sigma(G) & \text{si } G \in \mathbf{PRED}_{\mathcal{L}} \\
\sigma((G \circ H)) & = & (\sigma(G) \circ \sigma(H)) & \\
\multicolumn{4}{l}{\text{si } G, H \in \mathbf{PRED}_{\mathcal{L}} \text{ et } \circ \in \{\wedge, \vee, \rightarrow, \leftrightarrow, \oplus\}} \\
\sigma((qx_i\ G)) & = & (qx_i\ \sigma'(G)) & \text{si } G \in \mathbf{PRED}_{\mathcal{L}} \text{ et } q \in \{\forall, \exists\} \\
\sigma((qx\ G)) & = & (qx\ \sigma(G)) & \\
\multicolumn{4}{l}{\text{si } G \in \mathbf{PRED}_{\mathcal{L}}, q \in \{\forall, \exists\} \text{ et } x \notin \{x_1, \ldots, x_n\}}
\end{array}
$$

substitution des équivalents. cf. *equivalence logique.*
substitution obtenue à partir d'une valuation fonctionnelle. cf. définition 18.
support. cf. *substitution.*
symbole de constante. cf. *langage de la logique des prédicats.*
symbole de fonction. cf. *langage de la logique des prédicats.*
symbole de prédicat. cf. *langage de la logique des prédicats.*
symbole propositionnel. 1. QBF. cf. définition 1. **2. PROP.** cf. *langage de la logique propositionnelle.*
symbole propositionnel libre. cf. définition 4.
symbole propositionnel lié. cf. définition 4.
système $\mathbb{G}$ de Gentzen. cf. *calcul des séquents.*
système S_{QBF}. cf. définition 27.
système S_{QBF} enrichi. cf. définition 32.

TAUT. cf. *sémantique de la logique propositionnelle.*
tautologie. cf. *sémantique de la logique propositionnelle.*
terme. 1. QBF & PROP. cf. *cube.* **2. PRED.** cf. *langage de la logique des prédicats.*
test d'occurrence. cf. *résolution.*
top-down. cf. §. 1.5.

unification. cf. *résolution.*
universel. 1. Morph. cf. définition 1. **2. Sem.** cf. définitions 15 et 20.
universellement quantifié. cf. définition 4.

VAL_FONC. cf. définition 18.
VAL_QBF. cf. définition 18.
VAL_PROP. cf. définition 12.
valeur de vérité. cf. définition 11.
validité. **1.** QBF. cf. définitions 16 et 22 **2. Syn.** cf. *calcul des séquents*. **3.** PRED. cf. *sémantique de la logique des prédicats*.
validité universelle triviale. (en. Trivial truth) [48] Une QBF *prénexes* sous FNC (ne vérifiant pas la *falsifiabilité universelle triviale*), dont la *matrice* privée des instances des *symboles propositionnels universellement quantifiés* et des *quantificateurs* est *satisfiable*, est *valide*. Cette technique peut être appliquée à la volée (*dependency-directed backtracking by model intersections* [130]).
valuation fonctionnelle. cf. définition 18.
valuation propositionnelle. **1.** QBF. cf. définition 12. **. 2.** PROP. cf. *sémantique de la logique propositionnelle*.
valuation propositionnelle dans valuation fonctionnelle. cf. définition 18.
valuation QBF. cf. définition 18.
valué. cf. définition 18.
variable. cf. *langage de la logique des prédicats*.
variable libre. cf. *langage de la logique des prédicats*.
variable logique. cf. *programmation logique*.
vrai. cf. définition 11.

`WalkQSAT`. [95] La procédure `WalkQSAT` est une procédure pour le *problème de validité* des QBF *prénexes* dont la *matrice* est sous FNC basée sur un `QDLL` mis en coopération avec la procédure incomplète par recherche locale `WalkSAT` [199] pour le problème SAT. La procédure est similaire à la *procédure de décision* ***QUBE*** dans sa version « BJ » [105] si ce n'est que la procédure `WalkSAT` fournit des *modèles* propositionnels de la matrice de la QBF pour guider la recherche. Si la procédure `WalkSAT` n'est pas en mesure de trouver un modèle, la construction du *scénario* courant est interrompue et le *retour-arrière* est déclenché ; dans ce cas la *non-validité* sous l'hypothèse du scénario courant n'est pas *décidée* et la procédure est donc incomplète.

`WalkQSAT`

Incomplet	$PSPACE$	
Sémantique	Polynomial en espace	FNC prénexe
	Monolithique	
	FNC prénexe ∃-backjumping ∀-backjumping Propagation unitaire Littéraux monotones	

ZBDD. cf. *Diagramme de décision binaire.*

ZQSAT. [98, 99] ZQSAT est une *procédure de décision* pour le *problème de validité* des QBF *prénexes* dont la *matrice* est sous FNC basée sur un QDLL. La procédure ZQSAT utilise les ZBDD pour représenter les *clauses* ; l'*expansion top-down* des quantificateurs est réalisée par des opérations ensemblistes sur des ZBDD ; les sous-formules déjà démontrées *valides* sont stockées dans le ZBDD.

ZQSAT

Décision	*PSPACE*	
Symbolique	Exponentiel en espace	DAG
	Monolithique	
	FNC prénexe ∃-backtrack ∀-backtrack Propagation unitaire Littéraux monotones	

Bibliographie

[1] M.F. Ali, S. Safarpour, A. Veneris, M.S. Abadir, and R. Drechsler. Post-Verification Debugging of Hierarchical Designs. In *Proceedings of the International conference on Computer-aided design (ICCAD'05)*, pages 871–876, 2005.

[2] C. Anger, M. Gebser, T. Linke, A. Neumann, and T. Schaub. The nomore++ System. In *Proceedings of the 8th Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'05)*, pages 422–426, 2005.

[3] C. Ansotegui, C. Gomes, and B. Selman. Achilles' heel of QBF. In *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI'05)*, pages 275–281, 2005.

[4] B. Apsvall, M. Plass, and R. Tarjan. A Linear-Time Algorithm for Testing the Truth of certain Quantified Boolean Formulas and its Evaluation. *Information Processing Letters*, 8 :121–123, 1979.

[5] K.R. Apt. The essence of constraint propagation. *Theoretical Computer Science*, 221(1-2) :179–210, 1999.

[6] K.R. Apt. Some Remarks on Boolean Constraint Propagation. In *New trends in constraints*, volume 1865. Springer-Verlag, 2000.

[7] K.R. Apt and E. Monfroy. Constraint Programming viewed as Rule-based Programming. *Theory and Practice of Logic Programming (TPLP)*, 1(6) :713–750, 2001.

[8] G. Audemard, L. Bordeaux, Y. Hamadi, S. Jabbour, and L. Saïs. Un cadre général pour l'analyse des conflits. In *Actes des Quatrièmes Journées Francophones de Programmation par Contraintes (JFPC'08)*, pages 267–276, 2008.

[9] G. Audemard, S. Jabbour, and L. Saïs. Exploitation des symétries dans les formules booléennes quantifiées. In *Actes des Deuxièmes Journées Francophones de Programmation par Contraintes (JFPC'06)*, pages 257–266, 2006.

[10] G. Audemard, S. Jabbour, and L. Saïs. Efficient Symmetry Breaking Predicates for Quantified Boolean Formulae. In *Workshop on Symmetry and Constraint Satisfaction Problems*, pages 14–21, 2007.

[11] G. Audemard, S. Jabbour, and L. Saïs. Symmetry Breaking in Quantified Boolean Formulae. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI'07)*, pages 2262–2267, 2007.

[12] G. Audemard, B. Mazure, and L. Saïs. Dealing with symmetries in quantified Boolean formulas. In *Poster of the 7th International Confrerence on Theory and Applications of Satisfiability Testing (SAT'04)*, 2004.

[13] G. Audemard and L. Saïs. SAT Based BDD Solver for Quantified Boolean Formulas. In *Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'04)*, pages 82–89, 2004.

[14] G. Audemard and L. Saïs. A Symbolic Search Based Approach for Quantified Boolean Formulas. In *Proceedings of the 8th International Conference on Theory and Applications of Satisfiability Testing (SAT'05)*, 2005.

[15] G. Audemard and L. Saïs. Une approche symbolique pour les formules booléennes quantifiées. In *Actes des Premièmes Journées Francophones de Programmation par Contraintes (JFPC'05)*, pages 59–68, 2005.

[16] A. Ayari and D. Basin. Bounded model construction for monadic second-order logics. In *Proceedings of the 12th International Conference on Computer-Aided Verification (CAV'00)*, pages 99–113, 2000.

[17] A. Ayari and D. Basin. QUBOS : Deciding Quantified Boolean Logic using Propositional Satisfiability Solvers. In *Proceedings of the 4th International Conference on Formal Methods in Computer-Aided Design (FMCAD'02)*, pages 187–201, 2002.

[18] A. Ayari, D. Basin, and S. Friedrich. Structural and behavioral modeling with monadic logics. In Rolf Drechsler and Bernd Becker, editors, *Proceedings of the 29th IEEE International Symposium on Multiple-Valued Logic*, pages 142–151, Freiburg, Germany, 1999. IEEE Computer Society.

[19] F. Bacchus. Enhancing davis putnam with extended binary clause reasoning. In *Proceedings of the 8th National Conference on Artificial Intelligence (AAAI'02)*, pages 613–619, 2002.

[20] F. Bacchus and J. Winter. Effective preprocessing with hyper-resolution and equality reduction. In *Proceedings of the 6th International Conference on Theory and Applications of Satisfiability Testing (SAT'03)*, pages 341–355, 2003.

[21] R.J. Bayardo and J.D. Pehoushek. Counting Models Using Connected Components. In *Proceedings of the 17th National Conference on Artificial Intelligence (AAAI'00) and 12th Conference on Innovative Applications of Artificial Intelligence (IAAI'00)*, pages 157–162, 2000.

[22] R.J. Bayardo and R.C. Schrag. Using CSP look-back techniques to solve real-world SAT instances. In *Proceedings of the 14th National Conference on Artificial Intelligence and 9th Conference on Innovative Applications of Artificial Intelligence (AAAI'97)*, pages 203–208, 1997.

[23] M. Benedetti. Evaluating QBFs via Symbolic Skolemization. In *Proceedings of the 11th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'05)*, pages 285–300, 2005.

[24] M. Benedetti. Experimenting with QBF-based formal verification. In *Proceedings of the 3rd International Workshop on Constraints in Formal Verification*, 2005.

[25] M. Benedetti. Extracting Certificates from Quantified Boolean Formulas. In *Proceedings of 9th International Joint Conference on Artificial Intelligence (IJCAI'05)*, pages 47–53, 2005.

[26] M. Benedetti. Quantifier trees for QBFs. In *Poster of the 8th International Conference on Theory and Applications of Satisfiability Testing (SAT'05)*, 2005.

[27] M. Benedetti. skizzo : a suite to evaluate and certify QBFs. In *Proceedings of the 20th International Conference on Automated Deduction (CADE'05)*, pages 369–376, 2005.

[28] M. Benedetti. Abstract branching for quantified formulas. In *Proceedings of the 21th National Conference on Artificial Intelligence (AAAI'06)*, 2006.

[29] M. Benedetti, A. Lallouet, and J. Vautard. Reusing csp propagators for qcsps. In *Recent Advances in Constraints, 11th Annual ERCIM International, Workshop on Constraint Solving and Contraint Logic Programming (CSCLP'06)*, pages 63–77, 2006.

[30] M. Benedetti, A. Lallouet, and J. Vautard. Qcsp made practical by virtue of restricted quantification. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI'07)*, pages 38–43, 2007.

[31] M. Benedetti, A. Lallouet, and J. Vautard. Modeling adversary scheduling with qcsp+. In *Proceedings of the 23th ACM Symposium on Applied Computing (SAC'08)*, pages 151–155, 2008.

[32] M. Benedetti, A. Lallouet, and J. Vautard. Quantified constraint optimization. In *Proceedings of the 14th International Conference on Principles and Practice of Constraint Programming (CP'08)*, pages 463–477, 2008.

[33] M. Benedetti and H. Mangassarian. Experience and Perspectives in QBF-Based Formal Verification. *Journal on Satisfiability, Boolean Modeling and Computation*, 5 :133–191, 2008.

[34] P. Besnard, A. Hunter, and S. Woltran. Encoding Deductive Argumentation in Quantified Boolean Formulae. *Journal of Artificial Intelligence*, 173 :1406–1423, 2009.

[35] P. Besnard, T. Schaub, H. Tompits, and S. Woltran. Paraconsistent reasoning via quantified Boolean formulas, i : Axiomatising signed systems. In *Proceedings of the 8th European Conference on Logics in Artificial Intelligence (JELIA'02)*, pages 320–331, 2002.

[36] A. Biere. Resolve and Expand. In *Proceedings of the 7th International Confrerence on Theory and Applications of Satisfiability Testing (SAT'04)*, pages 59–70, 2004.

[37] A. Biere, A. Cimatti, E. M. Clarke, and Y. Zhu. Symbolic Model Checking without BDDs. In *Proceedings of the 5th International Conference on Tools and Algorithms for Construction and Analysis of Systems (TACAS'99)*, pages 193–207, 1999.

[38] M. Böhm and E. Speckenmeyer. A fast parallel sat-solver - efficient workload balancing. *Annals of Mathematics and Artificial Intelligence*, 17(3-4) :381–400, 1996.

[39] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm intelligence : from natural to artificial systems*. Oxford University Press, Inc., 1999.

[40] L. Bordeaux. Boolean and Interval Propagation for Quantified Constraints. In *Proceedings of the 1st International Workshop on Quantification in Constraint Programming*, 2005.

[41] L. Bordeaux and E. Monfroy. Beyond NP : Arc-consistency for quantified constraints. In *Proceedings of the 8th International Conference on Principles and Practice of Constraint Programming (CP'02)*, pages 371–386, 2002.

[42] F. Börner, A.A. Bulatov, P. Jeavons, and A.A. Krokhin. Quantified Constraints : Algorithms and Complexity. In *Proceedings of the 17th International Workshop on Computer Science Logic (CSL'03)*, pages 58–70, 2003.

[43] T. Boy de la Tour. An Optimality Result for Clause Form Translation. *Journal of Symbolic Computation*, 14(4) :283–302, 1992.

[44] R.E. Bryant. Graph-Based Algorithms for Boolean Function Manipulation. *IEEE Transactions on Computers*, 35(8) :677–691, 1986.

[45] U. Bubeck and H. Kleine Büning. Models and quantifier elimination for quantified Horn formulas. *Discrete Applied Mathematics*, 156/10 :1606–1622, 2008.

[46] M. Cadoli, A. Giovanardi, and M. Schaerf. Experimental Analysis of the Computational Cost of Evaluating Quantified Boolean Formulae. In *Proceedings of the 5th Conference of the Italian Association for Artificial Intelligence (AIIA'97)*, pages 207–218, 1997.

[47] M. Cadoli, A. Giovanardi, and M. Schaerf. An Algorithm to Evaluate Quantified Boolean Formulae. In *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI'98)*, pages 262–267, 1998.

[48] M. Cadoli, M. Schaerf, A. Giovanardi, and M. Giovanardi. An Algorithm to Evaluate Quantified Boolean Formulae and Its Experimental Evaluation. *Journal of Automated Reasoning*, 28(2) :101–142, 2002.

[49] K. Chandrasekar and M.S. Hsiao. Q-PREZ : QBF Evaluation Using Partition, Resolution and Elimination with ZBDDs. In *Proceedings of the 18th International Conference on VLSI Design (VLSI Design'05)*, pages 189–194, 2005.

[50] P. Chatalic and L. Simon. Multi-Resolution on Compressed Sets of Clauses. In *Proceedings of the 12th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'00)*, 2000.

[51] P. Chatalic and L. Simon. Résolution sur des ensembles de clauses compressés : comment doper DP. In *Actes des Journées Nationales sur la Résolution Pratique de Problèmes NP-Complets (JNPC'00)*, page 107–118, 2000.

[52] A. Colmerauer, A. Kanoui, P. Roussel, and R. Pasero. Un système de communication homme-machine en français. Technical report, Groupe de Recherche en Intelligence Artificielle, Université d'Aix-Marseille, 1973.

[53] S.A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd annual ACM symposium on Theory of computing (STOC'71)*, pages 151–158, 1971.

[54] S. Coste-Marquis, D. Le Berre, F. Letombe, and P. Marquis. Complexity Results for Quantified Boolean Formulae Based on Complete Propositional Languages. *Journal on Satisfiability, Boolean Modeling and Computation*, 1 :61–88, 2006.

[55] S. Coste-Marquis, H. Fargier, J. Lang, D. Le Berre, and P. Marquis. Résolution de formules booléennes quantifiées : problèmes et algorithmes. In *Actes du 13e Congrès AFRIF-AFIA Reconnaissance des Formes et Intelligence Artificielle (RFIA-02)*, Angers, France, 2002.

[56] S. Coste-Marquis, H. Fargier, J. Lang, D. Le Berre, and P. Marquis. Representing Policies for Quantified Boolean Formulae. In *Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR'06)*, pages 286–296, 2006.

[57] S. Coste-Marquis, D. Le Berre, and F. Letombe. A Branching Heuristics for Quantified Renamable Horn Formulas. In *Poster of the 8th International Conference on Theory and Applications of Satisfiability Testing (SAT'05)*, pages 393–399, 2005.

[58] B. Da Mota, P. Nicolas, and I. Stéphan. A new parallel architecture for qbf tools. In *Proceedings of the Workshop on Parallel Satisfiability Solving : SAT and beyond SAT, Parallel Solving on New Architectures (WPSS'10)*, à paraître, 2010.

[59] B. Da Mota, P. Nicolas, and I. Stéphan. Une nouvelle architecture parallèle pour le problème de validité des qbf. In *Actes des Cinquièmes Journées Francophones de Programmation par Contraintes (JFPC'10)*, à paraître, 2010.

[60] B. Da Mota, I. Stéphan, and P. Nicolas. From (Quantified) Boolean Formulas to Answer Set Programming. In *Proceedings of the 7th International Answer Set Programming Workshop (ASP'07)*, 2007.

[61] B. Da Mota, I. Stéphan, and P. Nicolas. Des formules booléennes (quantifiées) à la programmation par ensembles réponses. In *Reconnaissance des Formes et Intelligence Artificielle*, 2008.

[62] B. Da Mota, I. Stéphan, and P. Nicolas. Une mise sous forme prénexe préservant les résultats intermédiaires pour les formules booléennes quantifiées. In *Journées Nationales de l'IA Fondamentale (IAF'08)*, 2008.

[63] B. Da Mota, I. Stéphan, and P. Nicolas. A new prenexing strategy for quantified boolean formulae with bi-implications. In *Proceedings of the 6th International Workshop on Constraints in Formal Verification*, 2009.

[64] B. Da Mota, I. Stéphan, and P. Nicolas. Une nouvelle stratégie de mise sous forme prénexe pour des formules booléennes quantifiées avec bi-implications. *Revue internationale I3 (Information - Interaction - Intelligence)*, à paraître en 2010.

[65] A. Darwiche. Compiling knowledge into decomposable negation normal form. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI'99)*, pages 284–289. Morgan Kaufmann, 1999.

[66] A. Darwiche. Decomposable negation normal form. *Journal of the ACM*, 48(4) :608–647, 2001.

[67] A. Darwiche and P. Marquis. A Knowledge Compilation Map. *Journal of Artificial Intelligence Research*, 17 :229–264, 2002.

[68] M. Davis, G. Logemann, and D. Loveland. A machine program for theorem-proving. *Communication of the ACM*, 5 :394–397, 1962.

[69] M. Davis and H. Putnam. A computing procedure for quantification theory. *Journal of the ACM*, 7(3) :201–215, July 1960.

[70] E.D. Demaine and R.A. Hearn. Playing games with algorithms : Algorithmic combinatorial game theory. In *Proceedings of the 26th Symposium on Mathematical Foundations in Computer Science*, pages 18–32, 2001.

[71] N. Dershowitz, Z. Hanna, and J. Katz. Bounded Model Checking with QBF. In *Proceedings of the 8th International Conference on Theory and Applications of Satisfiability Testing (SAT'05)*, pages 408–414, 2005.

[72] M. Dorigo, E. Bonabeau, and G. Theraulaz. Ant algorithms and stimergy. *Future Generation Computer Systems*, 16 :851–871, 2000.

[73] N. Een and N. Sörensson. An extensible sat-solver. In *Proceedings of the 6th International Conference on Theory and Applications of Satisfiability Testing (SAT'03)*, 2003.

[74] U. Egly. On Different Structure-Preserving Translations to Normal Form. *Journal of Symbolic Computation*, 22(2) :121–142, 1996.

[75] U. Egly, T. Eiter, V. Klotz, H. Tompits, and S. Woltran. Computing Stable Models with Quantified Boolean Formulas : Some Experimental Results. In *Proceedings of the 2001 AAAI Spring Symposium on Answer Set Programming*, pages 53–59, 2001.

[76] U. Egly, T. Eiter, H. Tompits, and S. Woltran. QUIP - A Tool for Computing Nonmonotonic Reasoning Tasks. In *Proceedings of the 8th International Workshop on Non-Monotonic Reasoning (NMR'00)*, 2000.

[77] U. Egly, T. Eiter, H. Tompits, and S. Woltran. Solving Advanced Reasoning Tasks Using Quantified Boolean Formulas. In *Proceedings of the 17th National Conference on Artificial Intelligence (AAAI'00) and 12th Conference on Innovative Applications of Artificial Intelligence (IAAI'00)*, pages 417–422, 2000.

[78] U. Egly, M. Seidl, H. Tompits, S. Woltran, and M. Zolda. Comparing Different Prenexing Strategies for Quantified Boolean Formulas. In *Proceedings of the 6th International Conference on Theory and Applications of Satisfiability Testing (SAT'03)*, pages 214–228, 2003.

[79] U. Egly, M. Seidl, and S. Woltran. A Solver for QBFs in Nonprenex Form. In *Proceedings of the 17th European Conference on Artificial Intelligence (ECAI'06)*, pages 477–481, 2006.

[80] U. Egly, M. Seidl, and S. Woltran. A solver for QBFs in negation normal form. *Constraints*, 14(1) :38–79, 2009.

[81] U. Egly, H. Tompits, and S. Woltran. On Quantifier Shifting for Quantified Boolean Formulas. In *Proceedings of the SAT-02 Workshop on Theory and Applications of Quantified Boolean Formulas (QBF'02)*, pages 48–61, 2002.

[82] T. Eiter, V. Klotz, H. Tompits, and S. Woltran. Modal Nonmonotonic Logics Revisited : Efficient Encodings for the Basic Reasoning Tasks. In *Proceedings of the 11th Conference on Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX'02*, pages 100–114, 2002.

[83] H. Fargier and P. Marquis. On the Use of Partially Ordered Decision Graphs in Knowledge Compilation and Quantified Boolean Formulae. In *Proceedings of the 21th National Conference on Artificial Intelligence (AAAI'06)*, 2006.

[84] R. Feldmann, B. Monien, and S. Schamberger. A Distributed Algorithm to Evaluate Quantified Boolean Formulae. In *Proceedings of the 17th National Conference on Artificial Intelligence (AAAI'00) and 12th Conference on Innovative Applications of Artificial Intelligence (IAAI'00)*, 2000.

[85] T. Frühwirth. Theory and practice of constraint handling rules. *Journal of Logic Programming*, 37(1-3) :95–138, 1998.

[86] J.H. Gallier. *Logic for computer science : foundations of automatic theorem proving.* Harper & Row Publishers, Inc., 1985.

[87] L. Garcia and R. Sabbadin. Complexity results and algorithms for possibilistic influence diagrams. *Artificial Intelligence*, 172(8-9) :1018–1044, 2008.

[88] M.R. Garey and D.S. Johnson. *Computers and Intractability ; A Guide to the Theory of NP-Completeness.* W.H. Freeman & Co., 1990.

[89] M. Gebser, B. Kaufmann, A. Neumann, and T. Schaub. Conflict-Driven Answer Set Solving. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI'07)*, pages 386–392, 2007.

[90] M. Gebser, T. Schaub, and S. Thiele. Gringo : A new grounder for answer set programming. In *Proceedings of the 9th Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'07)*, pages 266–271, 2007.

[91] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In *Proceedings of the 5th International Conference on Logic Programming*, pages 1070–1080, 1988.

[92] M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9 :365–385, 1991.

[93] I. Gent, E. Giunchiglia, M. Narizzano, A. Rowley, and A. Tacchella. Watched Data Structures for QBF Solvers. In *Proceedings of the 6th International Conference on Theory and Applications of Satisfiability Testing (SAT'03)*, 2003.

[94] I. Gent and G.D. Rowley. Encoding Connect-4 using Quantified Boolean Formulae. In *Proceedings of the 2nd International Workshop on Modelling and Reformulating Constraint Satisfaction Problems*, pages 78–93, 2003.

[95] I.P. Gent, H.H. Hoos, A.G.D. Rowley, and K. Smyth. Unsing Stochastic Local Search to Solve Quantified Boolean Formulae. In *Proceedings of the 9th International Conference on Principles and Practice of Constraint Programming (CP'03)*, 2003.

[96] I.P Gent, P. Nightingale, A. Rowley, and K. Stergiou. Solving quantified constraint satisfaction problems. *Journal of Artificial Intelligence*, 172(6-7) :738–771, 2008.

[97] I.P. Gent and A.G.D. Rowley. Local and Global Complete Solution Learning Methods for QBF. In *Proceedings of the 8th International Conference on Theory and Applications of Satisfiability Testing (SAT'05)*, pages 91–106, 2005.

[98] M. GhasemZadeh, V. Klotz, and C. Meinel. Embedding Memoization to the Semantic Tree Search for Deciding QBFs. In *Proceedings of the 17th Australian Joint Conference on Artificial Intelligence (AJCAI'04)*, pages 681–693, 2004.

[99] M. GhasemZadeh, V. Klotz, and C. Meinel. FZQSAT : A QSAT Solver for QBFs in Prenex NNF (A Useful Tool for Circuit Verification). In *Proceedings of the 13th International Workshop on Logic and Synthesis (IWLS'04)*, pages 135–142, 2004.

[100] E. Giunchiglia, M. Maratea, A. Tacchella, and D. Zambonin. Evaluating search heuristics and optimization techniques in propositional satisfiability. In *Proceedings of the 1st International Joint Conference on Automated Reasoning (IJCAR'01)*, pages 347–363, 2001.

[101] E. Giunchiglia, M. Narizzano, and A. Tacchella. Backjumping for Quantified Boolean Logic Satisfiability. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI'01)*, pages 275–281, 2001.

[102] E. Giunchiglia, M. Narizzano, and A. Tacchella. Quantified Boolean Formulas satisfiability library (QBFLIB), 2001. `www.qbflib.org`.

[103] E. Giunchiglia, M. Narizzano, and A. Tacchella. QUBE : A System for Deciding Quantified Boolean Formulas Satisfiability. In *Proceedings of the 1st International Joint Conference on Automated Reasoning (IJCAR'01)*, pages 364–369, 2001.

[104] E. Giunchiglia, M. Narizzano, and A. Tacchella. Learning for quantified Boolean logic satisfiability. In *Proceedings of the 18th national conference on Artificial intelligence*, pages 649–654, 2002.

[105] E. Giunchiglia, M. Narizzano, and A. Tacchella. Backjumping for Quantified Boolean Logic Satisfiability. *Artificial Intelligence*, 145 :99–120, 2003.

[106] E. Giunchiglia, M. Narizzano, and A. Tacchella. QBF Reasoning on Real-World Instances. In *Proceedings of the 7th International Confrerence on Theory and Applications of Satisfiability Testing (SAT'04)*, page 105–121, 2004.

[107] E. Giunchiglia, M. Narizzano, and A. Tacchella. QuBE++ : an Efficient QBF solver. In *Formal Methods in Computer-Aided Design*, 2004.

[108] E. Giunchiglia, M. Narizzano, and A. Tacchella. Clause/Term Resolution and Learning in the Evaluation of Quantified Boolean Formulas. *Journal of Artificial Intelligence Research*, 26 :371–416, 2006.

[109] E. Giunchiglia, M. Narizzano, and A. Tacchella. Quantifier structure in search based procedures for QBFs. In *Proceedings of the conference on Design, automation and test in Europe (DATE'06)*, pages 812–817, 2006.

[110] F. Glover. Future paths for integer programming and links to artificial intelligence. *Computer & Operations Research*, 13(5) :533–549, 1986.

[111] G. Gottlob. Complexity results for nonmonotonic logics. *journal of logic and computation*, 2(3) :397–425, 1992.

[112] A. Goultiaeva, V. Iverson, and F. Bacchus. A Compact Representation for Syntactic Dependencies in QBFs. In *Proceedings of the 12th International Conference on Theory and Applications of Satisfiability Testing (SAT'09)*, pages 412–426, 2009.

[113] M. Hietalahti, F. Massacci, and I. Niemelä. DES : a challenge problem for nonmonotonic reasoning systems. In *Proceedings of the 8th International Workshop on Non-Monotonic Reasoning (NMR'00)*, 2000.

[114] C. Holzbaur. Specification of constraint based inference mechanism through extended unification, dissertation, dept. of medical cybernetics & ai, university of vienna, 1990.

[115] N. Hristov and A. Remshagen. Local search for quantified Boolean formulas. In *Proceedings of the 43rd annual Southeast regional conference (ACM-SE'05)*, pages 116–120, 2005.

[116] H. Jain, C. Bartzis, and E. Clarke. Satisfiability checking of non-clausal formulas using general matings. In *Proceedings of the 9th International Conference on Theory and Applications of Satisfiability Testing (SAT'06)*, 2006.

[117] T. Jussila and A. Biere. Compressing BMC Encodings with QBF. *Electronic Notes in Theoretical Computer Science*, 174(3) :45–56, 2007.

[118] T. Jussila, A. Biere, C. Sinz, D. Kröning, and C. Wintersteiger. A First Step Towards a Unified Proof Checker for QBF. In *Proceedings of the 10th International Confrerence on Theory and Applications of Satisfiability Testing (SAT'07)*, pages 201–214, 2007.

[119] T. Jussila, C. Sinz, and A. Biere. Extended resolution proofs for symbolic SAT solving with quantification. In *Proceedings of the 9th International Conference on Theory and Applications of Satisfiability Testing (SAT'06)*, pages 54–60, 2006.

[120] H. Kleine Büning, M. Karpinski, and A. Flögel. Resolution for quantified Boolean formulas. *Information and Computation*, 117(1) :12–18, 1995.

[121] H. Kleine Büning, K. Subramani, and X. Zhao. Boolean Functions as Models for Quantified Boolean Formulas. *Journal of Automated Reasoning*, 39(1) :49–75, 2007.

[122] H. Kleine Büning and X. Zhao. Equivalence Models for Quantified Boolean Formulas. In *Proceedings of the 7th International Confrerence on Theory and Applications of Satisfiability Testing (SAT'04)*, 2004.

[123] H. Kleine Büning and X. Zhao. On Models for Quantified Boolean Formulas. In *Logic versus Approximation, In Lecture Notes in Computer Science 3075*, 2004.

[124] R. Kontchakov, V. Ryzhikov, F. Wolter, and M. Zakharyaschev. Checking DL-Lite Modularity with QBF Solvers. In *Description Logics (DL'08)*, 2008.

[125] R.A. Kowalski. Predicate logic as a programming language. *Information Processing*, 1974.

[126] R.A. Kowalski. Algorithm = logic + control. *Communication of the ACM*, 22(7) :424–436, 1979.

[127] C. Lefèvre and P. Nicolas. A First Order Forward Chaining Approach for Answer Set Computing. In *Proceedings of the 10th Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'09)*, 2009.

[128] N. Leone, G. Pfeifer, W. Faber, T. Eiter, G. Gottlob, S. Perri, and F. Scarcello. The dlv system for knowledge representation and reasoning. *ACM Transactions on Computational Logic*, 7(3) :499–562, 2006.

[129] F. Letombe. *De la validité des formules booléennes quantifiées : étude de complexité et exploitation de classes traitables au sein d'un prouveur QBF*. PhD thesis, Thèse de l'Université d'Artois, 2005.

[130] R. Letz. Lemma and Model Caching in Decision Procedures for Quantified Boolean Formulas. In *Proceedings of the International Conference on Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX'02)*, pages 160–175, 2002.

[131] M. Lewis, P. Marin, T. Schubert, M. Narizzano, B. Becker, and E. Giunchiglia. PaQuBE : Distributed QBF Solving with Advanced Knowledge Sharing. In *Proceedings of the 12th International Conference on Theory and Applications of Satisfiability Testing (SAT'09)*, pages 509–523, 2009.

[132] M. Lewis, T. Schubert, and B. Becker. QMiraXT - A Multithreaded QBF Solver. In *Methoden und Beschreibungssprachen zur Modellierung und Verifikation von Schaltungen und Systemen (MBMV'09)*, pages 7–16, 2009.

[133] C.-M. Li and Anbulagan. Heuristics based on unit propagation for satisfiability problems. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI'97)*, pages 366–371, 1997.

[134] A.C. Ling, D.P. Singh, and S.D. Brown. FPGA Logic Synthesis Using Quantified Boolean Satisfiability. In *Proceedings of the 8th International Conference on Theory and Applications of Satisfiability Testing (SAT'05)*, pages 444–450, 2005.

[135] J.W. Lloyd. *Foundations of logic programming*. Springer-Verlag New York, Inc., 1987.

[136] F. Lonsing and A. Biere. Nenofex : Expanding NNF for QBF Solving. In *Proceedings of the Eleventh International Conference on Theory and Applications of Satisfiability Testing (SAT'08)*, pages 196–210, 2008.

[137] D.W. Loveland. A linear format for resolution. In *Proceedings of the Symposium on Automatic Demonstration*, 1968.

[138] F. Lu, M. K. Iyer, G. Parthasarathy, and K.-T. Cheng. An Efficient Sequential SAT Solver With Improved Search Strategies. In *Proceedings of the 8th Conference on Design, Automation and Test in Europe (DATE'05)*, 2005.

[139] F. Lu, L.-C. Wang, J. Moondanos, and Z. Hanna. A Signal Correlation Guided Circuit-SAT Solver. *Journal of Universal Computer Science*, 10(12) :1629–1654, 2004.

[140] D. Luckham. Refinements theorems in resolution theory. In *Proceedings of the Symposium on Automatic Demonstration*, 1968.

[141] A.K. Mackworth. Consistency in networks of relations. In *Artificial Intelligence*, volume 8, pages 99–118, 1977.

[142] H. Mangassarian, A. Veneris, S. Safarpour, M. Benedetti, and D. Smith. A performance-driven QBF-based iterative logic array representation with applications to verification, debug and test. In *Proceedings of the international conference on Computer-aided design (ICCAD'07)*, pages 240–245, 2007.

[143] W. Marek and M. Truszczyński. Autoepistemic logic. *Journal of the ACM*, 38(3) :587–618, 1991.

[144] J.P. Marques-Silva and K.Sakallah. GRASP - A New Search Algorithm for Satisfiability. In *Proceedings of IEEE/ACM International Confenrence on Computer-Aided Design (ICCAD'96)*, pages 220–227, 1996.

[145] A. Meier and V. Sorge. Applying sat solving in classification of finite algebras. *J. Autom. Reason.*, 35(1-3) :201–235, 2005.

[146] A.R. Meyer and L.J. Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential space. In *Proceedings of the 13th Annual Symposium on Switching and Automata Theory (SWAT'72)*, pages 125–129, 1972.

[147] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer Verlag, 1996.

[148] S. Minato. Zero-suppressed BDDs for set manipulation in combinatorial problems. In *Proceedings of the 30th Design Automation Conference (DAC'93)*, 1993.

[149] M.N. Mneimneh and K.A. Sakallah. Computing Vertex Eccentricity in Exponentially Large Graphs : QBF Formulation and Solution. In *Proceedings of the 6th International Conference on Theory and Applications of Satisfiability Testing (SAT'03)*, pages 411–425, 2003.

[150] R.C. Moore. Semantical considerations on nonmonotonic logic. *Journal of Artificial Intelligence*, 25(1) :75–94, 1985.

[151] M.W. Moskewicz, C.F. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff : Engineering an Efficient SAT Solver. In *Proceedings of the 38th Design Automation Conference (DAC'01)*, pages 530–535, 2001.

[152] M. Narizzano, L. Pulina, and A. Tacchella. Report of the Third QBF Solvers Evaluation. *Journal on Satisfiability, Boolean Modeling and Computation*, 2 :145–164, 2006.

[153] M. Narizzano, L. Pulina, and A. Tacchella. Second QBF solvers competition (QBFEVAL'07). *Poster of the 10th International Confrerence on Theory and Applications of Satisfiability Testing (SAT'07)*, 2007.

[154] P. Nicolas, L. Garcia, and I. Stéphan. A Possibilistic Inconsistency Handling in Answer Set Programming. In *European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, volume 3571 of LNCS, pages 402–414, 2005.

[155] P. Nicolas, L. Garcia, and I. Stéphan. Possibilistic Stable Models. In *Proceedings of 9th International Joint Conference on Artificial Intelligence (IJCAI'05)*, pages 248–253, 2005.

[156] P. Nicolas, L. Garcia, I. Stéphan, and C. Lefèvre. Traitement de l'incertitude pour la programmation par ensemble réponses. In *Journées Francophones de Programmation en Logique et Programmation par Contraintes (JFPLPC'04)*, 2004.

[157] P. Nicolas, L. Garcia, I. Stéphan, and C. Lefèvre. Possibilistic Uncertainty Handling for Answer Set Programming. *Annals of Mathematics and Artificial Intelligence*, 47(1-2) :139–181, 2006.

[158] P. Nicolas, L. Garcia, I. Stéphan, and C. Lefèvre. Traitement de l'incertitude pour la programmation par ensemble réponses. In *Reconnaissance des Formes et Intelligence Artificielle (RFIA'06)*, 2006.

[159] P. Nicolas, F. Saubion, and I. Stéphan. Combining Heuristics for Default Logic Reasoning Systems. In *Proceedings of the 12th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'00)*, 2000.

[160] P. Nicolas, F. Saubion, and I. Stéphan. GADEL : a Genetic Algorithm to Compute Default Logic Extensions. In *Proceedings of the 14th European Conference on Artificial Intelligence (ECAI'00)*, 2000.

[161] P. Nicolas, F. Saubion, and I. Stéphan. Genetic Algorithm for Extension Search in Default Logic. In *Proceedings of the 8th International Workshop on Non-Monotonic Reasoning (NMR 2000)*, 2000.

[162] P. Nicolas, F. Saubion, and I. Stéphan. Genes and Ants for Default Logic. In *Proceedings of the 1st International Answer Set Programming Workshop (ASP'01)*, 2001.

[163] P. Nicolas, F. Saubion, and I. Stéphan. Heuristics for a Default Logic Reasoning System. *International Journal on Artificial Intelligence Tools*, 10(4) :503–523, 2001.

[164] P. Nicolas, F. Saubion, and I. Stéphan. New Generation Systems for Non-monotonic Reasoning. In *Proceedings of the 6th Conference on Logic Programming and Non-monotonic Reasoning (LPNMR'01)*, pages 309–321, 2001.

[165] P. Nicolas, F. Saubion, and I. Stéphan. Answer Set Programming by Ant Colony Optimization. In *Proceedings of the 8th European Conference on Logics in Artificial Intelligence (JELIA'02)*, pages 481–492, 2002.

[166] P. Nicolas, F. Saubion, and I. Stéphan. Optimisation par colonies de fourmis pour la programmation logique étendue. In *Actes des Journées Francophones de Programmation en Logique avec Contraintes (JFPLC'02)*, pages 57–70, 2002.

[167] I. Niemelä. Logic programs with stable model semantics as a constraint programming paradigm. *Annals of Mathematics and Artificial Intelligence*, 25(3-4) :241–273, 1999.

[168] I. Niemelä and P. Simons. Evaluating an algorithm for default reasoning. In *Proceedings of the Workshop on Applications and Implementations of Nonmonotomic Reasonigs Systems*, pages 66–72, 1995.

[169] A. Nonnengart and C. Weidenbach. *Computing small clause normal forms*, volume 1, chapter 6, pages 335–367. Elsevier, 2001.

[170] C. Otwell, A. Remshagen, and K. Truemper. An effective QBF solver for planning problems. In *Proceedings of the International Conference on Algorithmic Mathematics and Computer Science (ICAMCS'04)*, pages 311–316, 2004.

[171] G. Pan and M.Y. Vardi. Symbolic Decision Procedures for QBF. In *International Conference on Principles and Practice of Constraint Programming*, 2004.

[172] C.M. Papadimitriou. *Computational complexity.* Addison-Wesley, 1994.

[173] D. Pearce, H. Tompits, and S. Woltran. Encodings for Equilibrium Logic and Logic Programs with Nested Expressions. In *Proceedings of the 10th Portuguese Conference on Artificial Intelligence on Progress in Artificial Intelligence, Knowledge Extraction, Multi-agent Systems, Logic Programming and Constraint Solving (EPIA'01)*, pages 306–320, 2001.

[174] C. Peschiera, L. Pulina, and A. Tacchella. Qbfeval'08 (`http://www.qbflib.org/`), 2008.

[175] F. Pigorsch and C. Scholl. Exploiting Structure in an AIG Based QBF Solver. In *Proceedings of the conference on Design, automation and test in Europe (DATE'09)*, 2009.

[176] D.A. Plaisted, A. Biere, and Y. Zhu. A satisfiability procedure for quantified Boolean formulae. *Discrete Applied Mathematics*, 130 :291–328, 2003.

[177] D.A. Plaisted and S. Greenbaum. A Structure-Preserving Clause Form Translation. *Journal of Symbolic Computation*, 2(3) :293–304, 1986.

[178] P. Prosser. Hybrid algorithms for the constraint satisfaction problem. *Computational Intelligence*, 9 :268–299, 1993.

[179] T.C. Przymusinski. Stable semantics for disjunctive programs. *New Generation Computing*, 9 :401–424, 1991.

[180] L. Pulina and A. Tacchella. A Multi-engine Solver for Quantified Boolean Formulas. In *Proceedings of the 12th International Conference on Principles and Practice of Constraint Programming, Lecture Notes in Computer Science (CP'06)*, pages 574–589, 2007.

[181] L. Pulina and A. Tacchella. QuBIS : An (In)complete Solver for Quantified Boolean Formulas. In *Proceedings of the 7th Mexican International Conference on Artificial Intelligence (MICAI'08)*, pages 34–43, 2008.

[182] L. Pulina and A. Tacchella. A self-adaptive multi-engine solver for quantified Boolean formulas. *Constraints*, 14(1) :80–116, 2009.

[183] D.P. Ranjan, D. Tang, and S. Malik. A Comparative Study of 2QBF Algorithms. In *Poster of the 7th International Confrerence on Theory and Applications of Satisfiability Testing (SAT'04)*, 2004.

[184] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13(1-2) :81–132, 1980.

[185] O. Ridoux. *λProlog de A à Z ... ou presque*. PhD thesis, Habilitation à diriger les recherches de l'Université de Rennes 1, avril 1998.

[186] J. Rintanen. Constructing conditional plans by a theorem-prover. *Journal of Artificial Intelligence Research*, 10 :323–352, 1999.

[187] J. Rintanen. Improvements to the Evaluation of Quantified Boolean Formulae. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI'99)*, pages 1192–1197, 1999.

[188] J. Rintanen. Asymptotically Optimal Encodings of Conformant Planning in QBF. In *Proceedings of the 22th National Conference on Artificial Intelligence (AAAI'07)*, pages 1045–1050, 2007.

[189] J.A. Robinson. A machine-oriented logic based on the resolution principle. *Journal of the ACM*, 12(1) :23–41, 1965.

[190] A. Sabharwal, C. Ansótegui, C.P. Gomes, J.W. Hart, and B. Selman. QBF Modeling : Exploiting Player Symmetry for Simplicity and Efficiency. In *Proceedings of the 9th International Conference on Theory and Applications of Satisfiability Testing (SAT'06)*, pages 382–395, 2006.

[191] M. Samer and S. Szeider. Backdoor Sets of Quantified Boolean Formulas. *Journal of Automated Reasoning*, 42(1) :77–97, 2009.

[192] H. Samulowitz and F. Bacchus. Using SAT in QBF. In *Proceedings of the 11th International Conference on Principles and Practice of Constraint Programming (CP'05)*, pages 578 – 592, 2005.

[193] H. Samulowitz and F. Bacchus. Binary Clause Reasoning in QBF. In *Proceedings of the 9th International Conference on Theory and Applications of Satisfiability Testing (SAT'06)*, pages 353–367, 2006.

[194] H. Samulowitz, J. Davies, and F. Bacchus. Preprocessing QBF. In *Proceedings of the 12th International Conference on Principles and Practice of Constraint Programming (CP'06)*, pages 514–529, 2006.

[195] C. Scholl and B. Becker. Checking Equivalence for Partial Implementations. In *Proceedings of the Design Automation Conference (DAC'01)*, pages 238–243, 2001.

[196] A. Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, 1998.

[197] T. Schubert, M. Lewis, and B. Becker. PaMiraXT : Parallel sat solving with threads and message passing. *Journal on Satisfiability, Boolean Modeling and Computation*, 6 :203–222, 2009.

[198] C. Schulte and G. Tack. Views and Iterators for Generic Constraint Implementations. In *Proceedings of the 11th International Conference on Principles and Practice of Constraint Programming (CP'05)*, pages 817–821, 2005.

[199] B. Selman and H.A. Kautz. Domain-Independent Extensions to GSAT : Solving Large Structured Satisfiability Problems. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence (IJCAI'93)*, pages 290–295, 1993.

[200] B. Selman, H.A. Kautz, and B. Cohen. Local search strategies for satisfiability testing. In *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 521–532, 1993.

[201] B. Selman and H.J. Levesque. Abductive and default reasoning : A computational core. In *Proceedings of the 8th National Conference on Artificial Intelligence and 2nd Conference on Innovative Applications of Artificial Intelligence (AAAI'90)*, pages 343–348, 1990.

[202] M. Sheeran and G. Stålmarck. A tutorial on Stålmarck's proof procedure for propositional logic. In G. Gopalakrishnan and P. Windley, editors, *Formal Methods in Computer-Aided Design*, volume 1522, pages 82–99. Springer-Verlag, Berlin, 1998.

[203] C. Sinz and A. Biere. Extended resolution proofs for conjoining bdds. In *Proceedings of the 1st International Computer Science Symposium in Russia (CSR'06)*, pages 600–611, 2006.

[204] M. Snir, S. Otto, D. Walker, J. Dongarra, and S. Huss-Lederman. *MPI : The Complete Reference*. 1995.

[205] F. Somenzi. CUDD : CU decision diagram package release 2.3.0. university of colorado at boulder, 1998.

[206] S. Staber and R. Bloem. Fault Localization and Correction with QBF. In *Proceedings of the 10th International Confrerence on Theory and Applications of Satisfiability Testing (SAT'07)*, pages 244–257, 2007.

[207] I. Stéphan. Algorithmes d'élimination de quantificateurs pour le calcul des politiques des formules booléennes quantifiées. In *Actes des Premières Journées Francophones de Programmation par Contraintes (JFPC'05)*, 2005.

[208] I. Stéphan. Finding models for quantified Boolean formulae. In *Proceedings of the 1st International Workshop on Quantification in Constraint Programming*, 2005.

[209] I. Stéphan. Boolean Propagation Based on Literals for Quantified Boolean Formulae. In *Proceedings of the 17th European Conference on Artificial Intelligence (ECAI'06)*, 2006.

[210] I. Stéphan. Propagation logique pour les formules booléennes quantifiées. In *Actes des Deuxièmes Journées Francophones de Programmation par Contraintes (JFPC'06)*, 2006.

[211] I. Stéphan. Une (presque) génération automatique d'un compilateur de tables de vérité vers un solveur pour formules booléennes quantifiées prénexes. In *Actes des Troisièmes Journées Francophones de Programmation par Contraintes (JFPC'07)*, 2007.

[212] I. Stéphan. Sémantique et calcul syntaxique pour les formules booléennes quantifiées. In *Actes des Quatrièmes Journées Francophones de Programmation par Contraintes (JFPC'09)*, 2009.

[213] I. Stéphan and B. Da Mota. Base littérale et certificat pour les formules booléennes quantifiées. In *Actes des Troisièmes Journées Francophones de Programmation par Contraintes (JFPC'08)*, 2008.

[214] I. Stéphan and B. Da Mota. A unified framework for Certificate and Compilation for QBF. In *Proceedings of the 3rd Indian Conference on Logic and its Applications*, 2009.

[215] I. Stéphan, B. Da Mota, and P. Nicolas. From (Quantified) Boolean Formulas to Answer Set Programming. *Journal of logic and computation*, 19(4) :565–590, 2009.

[216] I. Stéphan, F. Saubion, and P. Nicolas. Description of GADEL. In *System description of the 8th International Workshop on Non-Monotonic Reasoning (NMR 2000)*, 2000.

[217] L.J. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3 :1–22, 1977.

[218] L.J. Stockmeyer and A.R. Meyer. Word problems requiring exponential time. In *Proceedings of the 5th annual ACM symposium on Theory of computing (STOC '73)*, pages 1–9, 1973.

[219] T. Syrjänen. Implementation of local grounding for logic programs for stable model semantics. Technical report, Helsinki University of Technology, 1998.

[220] D. Tang and S. Malik. Solving Quantified Boolean Formulas with Circuit Observability Don't Cares. In *Proceedings of the 9th International Conference on Theory and Applications of Satisfiability Testing (SAT'06)*, pages 368–381, 2006.

[221] R. Tarjan. Depth-First Search and Linear Graph Algorithms. *SIAM Journal on Computing*, 1(2) :146–160, 1972.

[222] C. Thiffault, F. Bacchus, and T. Walsh. Solving Non-clausal Formulas with DPLL search. In *Proceedings of the 7th International Confrerence on Theory and Applications of Satisfiability Testing (SAT'04)*, 2004.

[223] G. S. Tseitin. On the complexity of derivation in propositional calculus. In A. O. Slisenko, editor, *Studies in Constructive Mathematics and Mathematical Logic, Part 2*, pages 115–125. Consultants Bureau, New York, 1970.

[224] N. Wirth. *Algorithms + Data Structures = Programs*. Prentice Hall PTR, 1978.

[225] P. Wolper. *Introduction à la calculabilité*. InterEditions, 1991.

[226] H. Yoshida, M. Ikeda, and K. Asada. Exact Minimum Logic Factoring via Quantified Boolean Satisfiability. In *Proceedings of IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, pages 1065–1068, 2006.

[227] L. Zhang. Solving QBF with combined conjunctive and disjunctive normal form. In *National Conference on Artificial Intelligence (AAAI'06)*, 2006.

[228] L. Zhang and S. Malik. Conflict Driven Learning in a Quantified Boolean Satisfiability Solver. In *International Conference on Computer Aided Design (ICCAD'02)*, 2002.

[229] L. Zhang and S. Malik. The quest for efficient boolean satisfiability solvers. In *CADE*, pages 295–313, 2002.

[230] L. Zhang and S. Malik. Towards symmetric treatment of conflits and satisfaction in quantified Boolean satisfiability solver. In *Proceedings of the 8th International Conference on Principles and Practice of Constraint Programming (CP'02)*, 2002.

Propos sur les formules booléennes quantifiées

Résumé

Le formalisme des « Formules Booléennes Quantifiées » étend la logique propositionnelle en permettant d'associer des quantificateurs existentiels ou universels aux symboles propositionnels. Vue comme un jeu fini à deux joueurs, une formule représente simultanément les règles et les conditions de victoire. La validité de la formule exprime que le joueur existentiel a une stratégie pour gagner quelles que soient les décisions de son adversaire. Mes travaux tentent de répondre aux questions suivantes : existe-t-il au moins une stratégie ? comment le décider efficacement ? comment représenter de manière compacte une stratégie ? comment la calculer ? comment être sûr que ce qui a été calculé est correct ? comment compiler la formule dans le même formalisme ou un autre déjà bien connu pour obtenir plus efficacement les stratégies. À toutes ces questions, des réponses aussi bien théoriques que pratiques ont été proposées.

Mots-clés : Formules booléennes quantifiées, compilation, validité.

Talk about Quantified Boolean Formulae

Abstract

The "Quantified Boolean Formula" formalism extends propositional logic by allowing to associate some existential or universal quantifiers to propositional symbols. Considered as a finite two-player game, a formula expresses simultaneously the rules and the victory conditions. Validity of a formula expresses that the existential player has a strategy to win whatever are the decisions of the adversary. My works try to answer the following questions : Is there at least one strategy ? How to decide it efficiently ? How to compactly represent a strategy ? How to compute one ? How to be sure that what has been computed is correct ? How to compile the formula in the same formalism or in another one already well known to obtain more efficiently the strategies ? To all these questions, theoretical and practical answers have be been proposed.

Keywords : Quantified Boolean Formulae, compilation, validity.

Démonstrations

Lemme 1 *Soient F une QBF sans quantificateur et v une valuation (propositionnelle). Alors $I^*(F)(v) = v^*(F)$.*

Démonstration du lemme 1 *La démonstration est immédiate par induction structurelle grâce aux définitions 13, 14 et 15 et à la définition de la* sémantique de la logique propositionnelle[(247)]*.*

Lemme 2 *Soient F et G deux QBF, F librement substituable pour un symbole propositionnel x dans G et v une valuation propositionnelle. Alors*

$$I^*([x \leftarrow F](G))(v) = I^*(G)(v[x := I^*(F)(v)])$$

Démonstration du lemme 2 *Le lemme se démontre par induction structurelle sur G.*

$G = x$

$$\begin{array}{ll} & I^*([x \leftarrow F](x))(v) \\ = & I^*(F)(v) \\ = & I^*(x)(v[x := I^*(F)(v)]) \end{array}$$

$G = y \neq x$

$$\begin{array}{ll} & I^*([y \leftarrow F](x))(v) \\ = & I^*(y)(v) \\ = & I^*(y)(v[x := I^*(F)(v)]) \end{array}$$

$G = \top$ *(Preuve similaire pour $\bot$)*

$$\begin{array}{ll} & I^*([\top \leftarrow F](x))(v) \\ = & I^*(\top)(v) \\ = & \textbf{vrai} \\ = & I^*(\top)(v[x := I^*(F)(v)]) \end{array}$$

$G = \neg H$ *(Preuves similaires pour $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$ et $\oplus$)*

$$\begin{array}{ll} & I^*([x \leftarrow F](\neg H))(v) \\ = & I^*(\neg[x \leftarrow F](H))(v) \\ = & i_\neg(I^*([x \leftarrow F](H))(v)) \\ & \textit{par hypothèse d'induction} \\ = & i_\neg(I^*(H)(v[x := I^*(F)(v)])) \\ = & I^*(\neg H)(v[x := I^*(F)(v)]) \end{array}$$

$G = (\exists y\ H)$, $x \neq y$, y *pas libre dans* F

$$
\begin{array}{ll}
& I^*([x \leftarrow F]((\exists y\ H)))(v) \\
= & I^*((\exists y\ [x \leftarrow F](H)))(v) \\
= & i_\vee(I^*([x \leftarrow F](H))(v[y := \mathbf{vrai}]), I^*([x \leftarrow F](H))(v[y := \mathbf{faux}])) \\
& \textit{par hypothèse d'induction} \\
= & i_\vee(I^*(H)(v[y := \mathbf{vrai}][x := I^*(F)(v[y := \mathbf{vrai}])]), \\
& I^*(H)(v[y := \mathbf{faux}][x := I^*(F)(v[y := \mathbf{faux}])])) \\
& y \textit{ pas libre dans } F \\
= & i_\vee(I^*(H)(v[y := \mathbf{vrai}][x := I^*(F)(v)]), \\
& I^*(H)(v[y := \mathbf{faux}][x := I^*(F)(v)])) \\
= & i_\vee(I^*(H)(v[x := I^*(F)(v)][y := \mathbf{vrai}]), \\
& I^*(H)(v[x := I^*(F)(v)][y := \mathbf{faux}])) \\
= & I^*((\exists y\ H))(v[x := I^*(F)(v)])
\end{array}
$$

$G = (\exists y\ H)$, $x \neq y$, y *libre dans* F, F *étant librement substituable pour* x *dans* $G = (\exists y\ H)$ *donc* x *n'apparaît pas libre dans* H *donc*

$$
\begin{array}{ll}
& I^*([x \leftarrow F]((\exists y\ H)))(v) \\
= & I^*((\exists y\ H))(v[x := I^*(F)(v)])
\end{array}
$$

$G = (\exists x\ H)$

$$
\begin{array}{ll}
& I^*([x \leftarrow F]((\exists x\ H)))(v) \\
= & I^*((\exists x\ H))(v) \\
= & I^*((\exists x\ H))(v[x := I^*(F)(v)])
\end{array}
$$

$G = (\forall y\ H)$. *La preuve est similaire au cas existentiel.*

Lemme 3 *Soit une QBF prénexe* F *et une valuation* v. *Alors* $I^*(F)(v) = I^*(F)(v, \emptyset)$.

Démonstration du lemme 3 *La démonstration est immédiate par induction structurelle grâce aux définitions 13, 14 et 15 pour la sémantique décisionnelle et aux définitions 13, 14 et 20 pour la sémantique fonctionnelle puisque la différence entre les définitions 15 et 20 réside dans le cas où la composante fonctionnelle est non vide.*

Lemme 4 *Soient* F *une QBF prénexe close et* sk *une valuation fonctionnelle. Alors, pour toutes valuations propositionnelles* v *et* v',

$$I^*(F)(v, sk) = I^*(F)(v', sk)$$

Démonstration du lemme 4 *La démonstration est immédiate par induction structurelle grâce aux définitions 13, 14 et 20 puisque la QBF étant close, seules les valuations introduites par la sémantique des quantificateurs sont accessibles par la définition 12 dans le cas d'arrêt portant sur le symbole propositionnel de la définition 20.*

Lemme 5 *Soit QM une QBF close prénexe quantifiée uniquement existentiellement. Si la valuation propositionnelle v est un modèle propositionnel de M alors la valuation QBF de composante fonctionnelle $\{(x \mapsto \hat{x})\}_{x \in \mathcal{V}(M)}$, telle que pour tout $x \in \mathcal{V}(M)$, $\hat{x} = v(x)$, est un modèle QBF de QM. Si la valuation QBF de composante fonctionnelle $\{(x \mapsto \hat{x})\}_{x \in \mathcal{V}(M)}$ est un modèle QBF de QM alors la valuation propositionnelle v, telle que pour tout $x \in \mathcal{V}(M)$, $v(x) = \hat{x}$, est un modèle propositionnel de M.*

Démonstration du lemme 5 *La démonstration est immédiate par induction structurelle grâce aux définitions 13, 14 et 15 et à la définition de la* sémantique de la logique propositionnelle[(247)]*.*

Lemme 6 *Soient F et G deux QBF, F librement substituable pour un symbole propositionnel x dans G, v une valuation propositionnelle et sk une valuation fonctionnelle. Alors*

$$I^*([x \leftarrow F](G))(v, sk) = I^*(G)(v[x := I^*(F)(v, sk)], sk)$$

Démonstration du lemme 6 *La démonstration est similaire à celle du lemme 2.*

Lemme 7 *Soit G une QBF prénexe dont l'unique symbole propositionnel libre est x et v une valuation propositionnelle. Alors*

$$I^*((\forall x\ G))(v, \emptyset) = I^*(([x \leftarrow \top](G) \wedge [x \leftarrow \bot](G)))(v, \emptyset)$$

et

$$I^*((\exists x\ G))(v, \emptyset) = I^*(([x \leftarrow \top](G) \vee [x \leftarrow \bot](G)))(v, \emptyset).$$

Démonstration du lemme 7

$$\begin{array}{ll} & I^*((\forall x\ G))(v, \emptyset) \\ = & I^x_\forall(I^*(G))(v, \emptyset) \\ = & i_\wedge(I^*(G)(v[x := \mathbf{vrai}], \emptyset), I^*(G)(v[x := \mathbf{faux}], \emptyset)) \\ & \textit{par lemme 6} \\ = & i_\wedge(I^*([x \leftarrow \top](G))(v, \emptyset), I^*([x \leftarrow \bot](G))(v, \emptyset)) \end{array}$$

La preuve est similaire pour le quantificateur existentiel.

Lemme 8 (validité et modèle) *Une QBF prénexe close est valide si et seulement si elle admet un modèle QBF.*

Démonstration du lemme 8 *Le lemme est une conséquence du lemme suivant : «soit sk une valuation fonctionnelle et v une valuation propositionnelle quelconque. $I^*(F)(v, sk) =$ **vrai** si et seulement s'il existe une valuation fonctionnelle sk' telle que $sk \cup sk'$ est une valuation fonctionnelle pour F et $I^*(F)(v, sk \cup sk') =$ **vrai**».*

Preuve par récurrence sur le nombre n de quantificateurs de la QBF F.

$n = 0$. Le lemme est vérifié avec $sk' = \emptyset$.

$n > 0$ *et* $F = \exists x G$. *S'il existe* $(x \mapsto \hat{x})$ *dans* sk *alors le lemme est trivialement vérifié par induction sinon*

$$I^*(\exists x G)(v, sk) = \mathbf{vrai}$$

si et seulement si

$$i_\vee(I^*(G)(v[x := \mathbf{vrai}], sk), I^*(G)(v[x := \mathbf{faux}], sk)) = \mathbf{vrai}$$

Sans perte de généralité, supposons que $I^*(G)(v[x := \mathbf{vrai}], sk) = \mathbf{vrai}$. *Par hypothèse d'induction* $I^*(G)(v[x := \mathbf{vrai}], sk) = \mathbf{vrai}$ *si et seulement s'il existe une valuation fonctionnelle* sk' *telle que* $sk \cup sk'$ *est une valuation fonctionnelle pour* G *et* $I^*(G)(v[x := \mathbf{vrai}], sk \cup sk') = \mathbf{vrai}$ *si et seulement si*

$$I^*(\exists x G)(v, sk \cup (sk' \cup \{(x \mapsto \mathbf{vrai})\})) = \mathbf{vrai}.$$

$n > 0$ *et* $F = \forall x G$. *Soit* sk *une valuation fonctionnelle et* v *une valuation quelconque. Soit* $\{x_i\}_{i \in I}$ *l'ensemble des symboles propositionnels existentiellement quantifiés de la QBF* G.

$$I^*(\forall x G)(v, sk) = \mathbf{vrai}$$

si et seulement si

$$i_\wedge(I^*(G)(v[x := \mathbf{vrai}], sk(\mathbf{vrai})), I^*(G)(v[x := \mathbf{faux}], sk(\mathbf{faux}))) = \mathbf{vrai}$$

si et seulement si

$$I^*(G)(v[x := \mathbf{vrai}], sk(\mathbf{vrai})) = \mathbf{vrai}$$

et

$$I^*(G)(v[x := \mathbf{faux}], sk(\mathbf{faux})) = \mathbf{vrai}$$

si et seulement si, par hypothèse de récurrence, il existe une valuation fonctionnelle $sk_{\mathbf{vrai}} = \{\{(x_i \mapsto f_i^{\mathbf{vrai}})\}_{i \in I}\}$ *telle que* $sk(\mathbf{vrai}) \cup sk_{\mathbf{vrai}}$ *est une valuation fonctionnelle de* G *et*

$$I^*(G)(v[x := \mathbf{vrai}], sk(\mathbf{vrai}) \cup sk_{\mathbf{vrai}}) = \mathbf{vrai}$$

et il existe une valuation fonctionnelle $sk_{\mathbf{faux}} = \{\{(x_i \mapsto f_i^{\mathbf{faux}})\}_{i \in I}\}$ *telle que* $sk(\mathbf{faux}) \cup sk_{\mathbf{faux}}$ *est une valuation fonctionnelle de* G *et*

$$I^*(G)(v[x := \mathbf{faux}], sk(\mathbf{faux}) \cup sk_{\mathbf{faux}}) = \mathbf{vrai}.$$

Maintenant, soit, pour tout i, $i \in I$, *la fonction* f_i *telle que* $f_i(\mathbf{vrai}) = f_i^{\mathbf{vrai}}$ *et* $f_i(\mathbf{faux}) = f_i^{\mathbf{faux}}$ *et* $sk' = \{(x_i \mapsto f_i)\}_{i \in I}$ *et en posant la valuation fonctionnelle* $sk'' = sk \cup sk'$,

$$\begin{array}{ll} & I^*(\forall x G)(v, sk'') \\ = & i_\wedge(I^*(G)(v[x := \mathbf{vrai}], sk''(\mathbf{vrai})), I^*(G)(v[x := \mathbf{faux}], sk''(\mathbf{faux}))) \\ = & \mathbf{vrai} \end{array}$$

La réciproque est triviale.

Lemme 9 (modèle QBF et tautologie) *Soit QM une QBF prénexe close dont les variables existentielles sont $\{x_1, \ldots, x_n\}$, $sk = \{(x_i \mapsto \hat{x_i})\}$ une valuation fonctionnelle pour la QBF QM, $\phi_{\hat{x_1}}, \ldots, \phi_{\hat{x_n}}$ les formules associées positivement aux fonctions $\hat{x_1}, \ldots, \hat{x_n}$ et la substitution $\sigma = [x_1 \leftarrow \phi_{\hat{x_1}}] \ldots [x_n \leftarrow \phi_{\hat{x_n}}]$ alors la valuation QBF (v, sk), v une valuation quelconque, est un modèle de la QBF prénexe QM si et seulement si la formule propositionnelle $\sigma(M)$ est une tautologie.*

Démonstration du lemme 9 *Par récurrence sur le nombre n de quantificateurs.*

$n = 0$. *Immédiat.*

$n > 0$ *et* $F = \exists x G$. *Soit $sk = (x \mapsto \hat{x}) \cup sk'$. Pour toute valuation v, la valuation QBF (v, sk) est un modèle QBF si et seulement si*

$$I^*(\exists x Q G)(v, sk) = \mathbf{vrai}$$

si et seulement si

$$I^*(QG)(v[x := \hat{x}], sk') = \mathbf{vrai}$$

si et seulement si

$$I^*(QG)(v[x := I^*(\phi_{\hat{x}})(v, sk')], sk') = \mathbf{vrai}$$

si et seulement si par le lemme 6

$$I^*([x \leftarrow \phi_{\hat{x}}](QG))(v, sk') = \mathbf{vrai}$$

si et seulement si

$$I^*(Q[x \leftarrow \phi_{\hat{x}}](G))(v, sk') = \mathbf{vrai}$$

si et seulement si, par récurrence,

$$v^*([x \leftarrow \phi_{\hat{x}}]\sigma(G)) = \mathbf{vrai}.$$

$n > 0$ *et* $F = \forall x G$. *Soit $\{x_1, \ldots, x_n\}$ les symboles propositionnels existentiellement quantifiés de G. Soit la valuation fonctionnelle $sk = \{(x_i \mapsto \hat{x_i})\}_{1 \leq i \leq n}$; pour tout $1 \leq i \leq n$, $\phi_{\hat{x_i}}$ est la formule propositionnelle associée positivement à la fonction booléenne $\hat{x_i}$ et $[x \leftarrow \top](\phi_{\hat{x_i}})$ (resp. $[x \leftarrow \bot](\phi_{\hat{x_i}})$) est la formule propositionnelle associée positivement à la fonction booléenne $\hat{x_i}(\mathbf{vrai})$ (resp. $\hat{x_i}(\mathbf{faux})$). Pour toute valuation v, la valuation QBF (v, sk) est un modèle QBF si et seulement si*

$$I^*(\forall x G)(v, sk) = \mathbf{vrai}$$

si et seulement si

$$i_\wedge(I^*(G)(v[x := \mathbf{vrai}], sk(\mathbf{vrai})), I^*(G)(v[x := \mathbf{faux}], sk(\mathbf{faux}))) = \mathbf{vrai}$$

si et seulement si

$$I^*(G)(v[x := \mathbf{vrai}], sk(\mathbf{vrai})) = \mathbf{vrai} \text{ et } I^*(G)(v[x := \mathbf{faux}], sk(\mathbf{faux})) = \mathbf{vrai}$$

si et seulement si, par hypothèse de récurrence,

$$v[x := \mathbf{vrai}]^*([x_1 \leftarrow [x \leftarrow \top](\phi_{\hat{x_1}})] \dots [x_n \leftarrow [x \leftarrow \top](\phi_{\hat{x_n}})](G)) = \mathbf{vrai}$$

et

$$v[x := \mathbf{faux}]^*([x_1 \leftarrow [x \leftarrow \bot](\phi_{\hat{x_1}})] \dots [x_n \leftarrow [x \leftarrow \bot](\phi_{\hat{x_n}})](G)) = \mathbf{vrai}$$

si et seulement si

$$v^*([x_1 \leftarrow [x \leftarrow \top](\phi_{\hat{x_1}})] \dots [x_n \leftarrow [x \leftarrow \top](\phi_{\hat{x_n}})]([x \leftarrow \top](G))) = \mathbf{vrai}$$

et

$$v^*([x_1 \leftarrow [x \leftarrow \bot](\phi_{\hat{x_1}})] \dots [x_n \leftarrow [x \leftarrow \bot](\phi_{\hat{x_n}})]([x \leftarrow \bot](G))) = \mathbf{vrai}$$

si et seulement si $v[x := b]^*([x_1 \leftarrow \phi_{\hat{x_1}}] \dots [x_n \leftarrow \phi_{\hat{x_n}}](G)) = \mathbf{vrai}$ *pour tout booléen* b.

Lemme 10 *Soit la QBF FNC prénexe (close)* QM *dont les variables existentielles sont* $\{x_1, \dots, x_n\}$. *Si la valuation QBF* $(v, \{(x_i \mapsto \hat{x_i})\}_{1 \leq i \leq n})$, v *une valuation propositionnelle quelconque, est un modèle QBF pour la QBF* QM *alors la substitution*

$$\sigma = [x_1 \leftarrow \phi_{\hat{x_1}}][\neg x_1 \leftarrow \nu_{\hat{x_1}}] \dots [x_n \leftarrow \phi_{\hat{x_n}}][\neg x_n \leftarrow \nu_{\hat{x_n}}]$$

est telle que $\sigma(M)$ *est une tautologie.*

Démonstration du lemme 10 *Soit* QM *une QBF prénexe dont les variables existentielles sont* $\{x_1, \dots, x_n\}$, *un valuation QBF* V *pour la QBF* QM *de valuation fonctionnelle* $\{(x_i \mapsto \hat{x_i})\}$, $\phi_{\hat{x_i}}, \dots, \phi_{\hat{x_n}}$ *(resp.* $\nu_{\hat{x_i}}, \dots, \nu_{\hat{x_n}}$*) les formules associées positivement (resp. négativement) aux fonctions* $\hat{x_i}, \dots, \hat{x_n}$ *et la substitution* $\sigma = [x_1 \leftarrow \phi_{\hat{x_1}}] \dots [x_n \leftarrow \phi_{\hat{x_n}}]$ *alors, par le lemme 9, la valuation QBF* V *est un modèle de la QBF* QM *si et seulement si la formule propositionnelle* $\sigma(M)$ *est une tautologie. Soit la substitution* $\sigma' = [x_1 \leftarrow \phi_{\hat{x_1}}][\neg x_1 \leftarrow \nu_{\hat{x_1}}] \dots [x_n \leftarrow \phi_{\hat{x_n}}][\neg x_n \leftarrow \nu_{\hat{x_n}}]$ *et* fnc *la forme normale conjonctive de* M. *Maintenant, pour tout* $i, 1 \leq i \leq n$, $\neg\phi_{\hat{x_i}} \equiv \nu_{\hat{x_i}}$ *donc* $\sigma'(fnc) \equiv \sigma(M)$ *d'où le résultat.*

Lemme technique 1 *Soient* F *une formule propositionnelle,* Q *un lieur,* x *un symbole propositionnel n'apparaissant pas dans* Q, v *une valuation propositionnelle,* sk *une valuation fonctionnelle et* $\circ = \wedge$ *ou* $\circ = \vee$ *alors*

$$I^*(Q(x \circ F))(v) = i_\circ(v(x), I^*(QF)(v)).$$

et

$$I^*(Q(x \circ F))(v, sk) = i_\circ(v(x), I^*(QF)(v, sk)).$$

Démonstration du lemme technique 1 *Par récurrence sur le nombre* n *de quantificateurs de* Q.

$n = 0$. $I^*((x \circ F))(v, sk) = i_\circ(v(x), I^*(F)(v, sk))$.

$n > 0$. *S'il n'existe pas* $(y, \hat{y}) \in sk$

$$
\begin{array}{ll}
& I^*(\exists y Q(x \circ F))(v, sk) \\
= & i_\vee(I^*(Q(x \circ F))(v[y := \mathbf{vrai}], sk), I^*(Q(x \circ F))(v[y := \mathbf{faux}], sk)) \\
= & i_\vee(i_\circ(v[y := \mathbf{vrai}](x), I^*(QF)(v[y := \mathbf{vrai}], sk)), \\
& i_\circ(v[y := \mathbf{faux}](x), I^*(QF)(v[y := \mathbf{faux}], sk))) \\
= & i_\vee(i_\circ(v(x), I^*(QF)(v[y := \mathbf{vrai}], sk)), i_\circ(v(x), I^*(QF)(v[y := \mathbf{faux}], sk))) \\
= & i_\circ(v(x), i_\vee(I^*(QF)(v[y := \mathbf{vrai}], sk), I^*(QF)(v[y := \mathbf{faux}], sk))) \\
= & i_\circ(v(x), I^*(QF)(v, sk))
\end{array}
$$

Pour le quantificateur universel, la démonstration est similaire.
Si $sk = \{(y, \hat{y})\} \cup sk'$

$$
\begin{array}{ll}
& I^*(\exists y Q(x \circ F))(v, sk) \\
= & I^*(Q(x \circ F))(v[y := \hat{y}], sk') \\
= & i_\circ(v(x), I^*(QF)(v[y := \hat{y}], sk')) \\
= & i_\circ(v(x), I^*(\exists y QF)(v, sk))
\end{array}
$$

La preuve est similaire pour la sémantique décisionnelle en omettant la valuation fonctionnelle et le dernier item.

Lemme 11 *La relation* $\equiv$ *est une relation d'équivalence pour les QBF.*

Démonstration du lemme 11 *Soient* F, G *et* H *trois QBF.*

réflexivité : $F \equiv F$ *si et seulement si* $I^*((F \leftrightarrow F))(v) =$ **vrai** *ce qui est immédiat par les définition 13, 14 et 15.*

symétrie : si $F \equiv G$ *alors* $I^*((F \leftrightarrow G))(v) =$ **vrai** *donc par les définition 13, 14 et 15* $I^*((G \leftrightarrow F))(v) =$ **vrai** *donc* $G \equiv F$

transitivité : si $F \equiv G$ *et* $G \equiv H$ *alors* $I^*((F \leftrightarrow G))(v) =$ **vrai** *et* $I^*((G \leftrightarrow H))(v) =$ **vrai** *donc, par la définition 13,* $I^*(F)(v) = I^*(G)(v)$ *et* $I^*(G)(v) = I^*(H)(v)$ *donc* $I^*(F)(v) = I^*(H)(v)$ *donc, par la définition 13,* $I^*((F \leftrightarrow H))(v) =$ **vrai** *donc* $F \equiv H$.

Lemme 12 *Soit* F *une QBF,* y *un symbole propositionnel n'apparaissant pas dans* F *et* q *un quantificateur. Alors* $(qx\ F) \equiv (qy\ [x \leftarrow y](F))$.

Démonstration du lemme 12 *Soit* F *une QBF,* y *un symbole propositionnel n'apparaissant pas dans* F.

$$
\begin{array}{ll}
& I^*((\exists y\ [x \leftarrow y](F)))(v) \\
= & i_\vee(I^*([x \leftarrow y](F))(v[y := \mathbf{vrai}]), I^*([x \leftarrow y](F))(v[y := \mathbf{faux}])) \\
& \textit{par le lemme 2} \\
= & i_\vee(I^*(F)(v[y := \mathbf{vrai}][x := I^*(y)(v[y := \mathbf{vrai}])]), \\
& I^*(F)(v[y := \mathbf{faux}][x := I^*(y)(v[y := \mathbf{faux}])])) \\
= & i_\vee(I^*(F)(v[y := \mathbf{vrai}][x := v[y := \mathbf{vrai}](y)]), I^*(F)(v[y := \mathbf{faux}][x := v[y := \mathbf{faux}](y)])) \\
= & i_\vee(I^*(F)(v[y := \mathbf{vrai}][x := \mathbf{vrai}]), I^*(F)(v[y := \mathbf{faux}][x := \mathbf{faux}])) \\
= & i_\vee(I^*(F)(v[x := \mathbf{vrai}]), I^*(F)(v[x := \mathbf{faux}]))
\end{array}
$$

Donc $(\exists x\ F) \equiv (\exists y\ [x \leftarrow y](F))$. *La démonstration est similaire pour le quantificateur universel.*

Lemme 13 *Soient F et G deux QBF, x et y deux symboles propositionnels, Q un lieur et q un quantificateur. Alors*

($Q^{\equiv}$.1) si $F \equiv G$ alors $qxF \equiv qxG$;
($Q^{\equiv}$.2) $\exists x Q(x \wedge G) \equiv Q[x \leftarrow \top](G)$;
($Q^{\equiv}$.3) $\exists x Q(\neg x \wedge G) \equiv Q[x \leftarrow \bot](G)$;
($Q^{\equiv}$.4) $\forall x Q(x \vee G) \equiv Q[x \leftarrow \bot](G)$;
($Q^{\equiv}$.5) $\forall x Q(\neg x \vee G) \equiv Q[x \leftarrow \top](G)$;
($Q^{\equiv}$.6) $\forall x x \equiv \bot$;
($Q^{\equiv}$.7) $\forall x \neg x \equiv \bot$;
($Q^{\equiv}$.8) $\exists x x \equiv \top$;
($Q^{\equiv}$.9) $\exists x \neg x \equiv \top$.

Les équivalences propositionnelles 0.1 *à* 0.30 *et les équivalences du premier ordre* 1.1 *à* 1.14 *de l'*équivalence logique[(84)] *sont vérifiées pour les QBF.*

Démonstration du lemme 13 *Soient F et G deux QBF, x et y deux symboles propositionnels, Q un lieur et q un quantificateur.*

$Q^{\equiv}$.1 *Nous prouvons que si $I^*(F)(v) = I^*(G)(v)$ alors $I^*(qxF)(v) = I^*(qxG)(v)$ pour toute valuation propositionnelle v. Par hypothèse $I^*(F)(v) = I^*(G)(v)$ donc*

$$I^*(F)(v[x := \mathbf{vrai}]) = I^*(G)(v[x := \mathbf{vrai}])$$

et

$$I^*(F)(v[x := \mathbf{faux}]) = I^*(G)(v[x := \mathbf{faux}])$$

or

$$I^*(qxF)(v) = i_\circ(I^*(F)(v[x := \mathbf{vrai}]), I^*(F)(v[x := \mathbf{faux}]))$$

et

$$I^*(qxG)(v) = i_\circ(I^*(G)(v[x := \mathbf{vrai}]), I^*(G)(v[x := \mathbf{faux}]))$$

avec $\circ = \vee$ si $q = \exists$ et $\circ = \wedge$ si $q = \forall$ donc $I^*(qxF)(v) = I^*(qxG)(v)$. Donc $qxF \equiv qxG$.

$Q^{\equiv}$.2

$$\begin{array}{l} I^*(\exists x Q(x \wedge G))(v) \\ = \quad i_\vee(I^*(Q(x \wedge G))(v[x := \mathbf{vrai}]), I^*(Q(x \wedge G))(v[x := \mathbf{faux}])) \end{array}$$

Par le lemme technique 1

$$\begin{array}{l} I^*(Q(x \wedge G))(v[x := \mathbf{vrai}]) \\ = \quad i_\wedge(v[x := \mathbf{vrai}](x), I^*(QG)(v[x := \mathbf{vrai}])) \\ = \quad i_\wedge(\mathbf{vrai}, I^*([x \leftarrow \top](QG))(v)) \text{ par le lemme 2} \\ = \quad I^*(Q[x \leftarrow \top](G))(v) \end{array}$$

et $I^*(Q(x \wedge G))(v[x := \mathbf{faux}]) = i_\wedge(\mathbf{faux}, I^*(QG)(v[x := \mathbf{faux}])) = \mathbf{faux}$.
Donc $I^*(\exists x Q(x \wedge G))(v) = I^*(Q[x \leftarrow \top](G))(v)$.

$Q^{\equiv}$.3 *Idem* $Q^{\equiv}$.2.

$Q^{\equiv}.4$ *Idem* $Q^{\equiv}.2$.
$Q^{\equiv}.5$ *Idem* $Q^{\equiv}.2$.
$Q^{\equiv}.6$ *Directement de* $Q^{\equiv}.4$ *avec* $G = \bot$.
$Q^{\equiv}.7$ *Directement de* $Q^{\equiv}.5$ *avec* $G = \bot$.
$Q^{\equiv}.8$ *Directement de* $Q^{\equiv}.2$ *avec* $G = \top$.
$Q^{\equiv}.9$ *Directement de* $Q^{\equiv}.3$ *avec* $G = \top$.

Lemme 14 *Soient F, F' et G trois QBF, x un symbole propositionnel. Si $F \equiv F'$ alors $[x \leftarrow F](G) \equiv [x \leftarrow F'](G)$.*

Démonstration du lemme 14 *Par hypothèse, $F \equiv F'$, donc $I^*(F)(v) = I^*(F')(v)$. Le lemme se démontre par induction structurelle sur G.*

$G = x$

$$\begin{array}{ll} & I^*([x \leftarrow F](x))(v) \\ = & I^*(F)(v) \\ = & I^*(x)(v[x := I^*(F)(v)]) \\ = & I^*(x)(v[x := I^*(F')(v)]) \\ = & I^*(F')(v) \\ = & I^*([x \leftarrow F'](x))(v) \end{array}$$

$G = y \neq x$

$$\begin{array}{ll} & I^*([y \leftarrow F](x))(v) \\ = & I^*(y)(v) \\ = & I^*([y \leftarrow F'](x))(v) \end{array}$$

$G = \top$ *(Preuve similaire pour $\bot$)*

$$\begin{array}{ll} & I^*([\top \leftarrow F](x))(v) \\ = & I^*(\top)(v) \\ = & I^*([\top \leftarrow F'](x))(v) \end{array}$$

$G = \neg H$ *(Preuves similaires pour $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$ et $\oplus$)*

$$\begin{array}{ll} & I^*([x \leftarrow F](\neg H))(v) \\ = & I^*(\neg[x \leftarrow F](H))(v) \\ = & i_\neg(I^*([x \leftarrow F](H))(v)) \\ & \textit{par hypothèse d'induction} \\ = & i_\neg(I^*([x \leftarrow F'](H))(v)) \\ = & I^*(\neg[x \leftarrow F'](H))(v) \\ = & I^*([x \leftarrow F'](\neg H))(v) \end{array}$$

$G = (\exists y\ H)$, $x \neq y$

$$\begin{array}{ll} & I^*([x \leftarrow F]((\exists y\ H)))(v) \\ = & I^*((\exists y\ [x \leftarrow F](H)))(v) \\ = & i_\vee(I^*([x \leftarrow F](H))(v[y := \mathbf{vrai}]), I^*([x \leftarrow F](H))(v[y := \mathbf{faux}])) \\ & \textit{par hypothèse d'induction} \\ = & i_\vee(I^*([x \leftarrow F'](H))(v[y := \mathbf{vrai}]), I^*([x \leftarrow F'](H))(v[y := \mathbf{faux}])) \\ = & I^*((\exists y\ [x \leftarrow F'](H)))(v) \\ = & I^*([x \leftarrow F'](\exists y\ H)))(v) \end{array}$$

$G = (\exists x\ H)$

$$\begin{array}{ll} & I^*([x \leftarrow F](\exists x\ H)))(v) \\ = & I^*((\exists x\ H))(v) \\ = & I^*([x \leftarrow F'](\exists x\ H)))(v) \end{array}$$

$G = (\forall y\ H)$. *La preuve est similaire au cas existentiel.*

Lemme 15 *La relation $\cong$ est une relation d'équivalence pour les QBF prénexes.*

Démonstration du lemme 15 *La démonstration est similaire à celle du lemme 11.*

Lemme 16 *Soient F et F' deux QBF prénexes.*
- *Si $F \cong F'$ alors $F \equiv F'$.*
- *De plus si F et F' sont deux QBF sans quantificateur alors $F \equiv F'$ si et seulement si $F \cong F'$.*

Démonstration du lemme 16 *Soient F et F' deux QBF prénexes telles que $F \cong F'$. Donc, par la définition 24, pour toute valuation propositionnelle v et pour toute valuation fonctionnelle sk, $i_\leftrightarrow(I^*(F)(v, sk), I^*(F')(v, sk)) = \mathbf{vrai}$. Donc, par la définition 13, $I^*(F)(v, sk) = I^*(F')(v, sk)$. Donc, pour $sk = \emptyset$, $I^*(F)(v, \emptyset) = I^*(F')(v, \emptyset)$. Donc, par le lemme 3, $I^*(F)(v) = I^*(F')(v)$. Donc, par la définition 13, $i_\leftrightarrow(I^*(F)(v), I^*(F')(v)) = \mathbf{vrai}$. Donc, par la définition 23,$F \equiv F'$.*

Maintenant, $F \equiv F'$ donc, par la définition 23, $i_\leftrightarrow(I^*(F)(v), I^*(F')(v)) = \mathbf{vrai}$, pour toute valuation v. Donc, par la définition 13, $I^*(F)(v) = I^*(F')(v)$. Donc, par le lemme 3, $I^*(F)(v, \emptyset) = I^*(F')(v, \emptyset)$. Donc, comme F et F' sont sans quantificateur, pour toute valuation fonctionnelle sk, $I^*(F)(v, sk) = I^*(F')(v, sk)$. Donc, par la définition 13, $i_\leftrightarrow(I^*(F)(v, sk), I^*(F')(v, sk)) = \mathbf{vrai}$. Donc, par la définition 24, $F \cong F'$.

Lemme 17 *Soient F et F' deux QBF prénexes. Si $F \cong F'$ et x un symbole propositionnel lié ni dans F ni dans F' alors $\exists xF \cong \exists xF'$ et $\forall xF \cong \forall xF'$.*

De plus, si F et F' sont sans quantificateur, $F \equiv F'$ et x un symbole propositionnel alors $\exists xF \cong \exists xF'$ et $\forall xF \cong \forall xF'$.

Démonstration du lemme 17 *Par hypothèse $F \cong F'$ donc par la définition 24 ceci est équivalent à $I^*(F)(v, sk) = I^*(F')(v, sk)$ pour toute valuation propositionnelle v, pour*

toute valuation fonctionnelle sk pour F et F′. Donc pour une valuation propositionnelle quelconque v′ telle que $v = v'[x := \hat{x}]$, $\hat{x} \in \mathbf{BOOL}$,

$$I^*(F)(v'[x := \hat{x}], sk) = I^*(F')(v'[x := \hat{x}], sk)$$

- *Si* $sk' = \{(x \mapsto \hat{x})\} \cup sk$ *donc* $I^*(\exists xF)(v', sk') = I^*(\exists xF')(v', sk')$.
- *Sinon*

$$\begin{array}{rl} & I^*(\exists xF)(v', sk) \\ = & i_\vee(I^*(F)(v'[x := \mathbf{vrai}], sk), I^*(F)(v'[x := \mathbf{faux}], sk)) \\ = & i_\vee(I^*(F')(v'[x := \mathbf{vrai}], sk), I^*(F')(v'[x := \mathbf{faux}], sk)) \\ = & I^*(\exists xF')(v', sk) \end{array}$$

- *La preuve pour la quantification universelle est similaire au second item de la preuve pour la quantification existentielle.*

Le corollaire est obtenue en appliquant le lemme 16 et ce qui précède.

Lemme 18 *Soient F et G deux QBF prénexes, H une formule propositionnelle, x et y deux symboles propositionnels et Q un lieur. Alors*

($Q^{\cong}$*.1)* $\exists x \exists y F \cong \exists y \exists x F$ *;*
($Q^{\cong}$*.2)* $I^*(\forall x \forall y F)(v, sk) = I^*(\forall y \forall x F)(v, sk')$ *avec sk et sk′ identiques à ceci près que les deux premières colonnes sont permutées ;*
($Q^{\cong}$*.3)* $\exists x Q(x \wedge H) \cong \exists x Q(x \wedge [x \leftarrow \top](H))$ *;*
($Q^{\cong}$*.4)* $\exists x Q(\neg x \wedge H) \cong \exists x Q(\neg x \wedge [x \leftarrow \bot](H))$ *;*
($Q^{\cong}$*.5)* $I^*(\forall x Q(x \vee H))(v, sk) = I^*(Q[x \leftarrow \bot](H))(v, sk(\mathbf{faux}))$ *;*
($Q^{\cong}$*.6)* $I^*(\forall x Q(\neg x \vee H))(v, sk) = I^*(Q[x \leftarrow \top](H))(v, sk(\mathbf{vrai}))$ *;*
($Q^{\cong}$*.7)* $\forall x x \cong \bot$ *;*
($Q^{\cong}$*.8)* $\forall x \neg x \cong \bot$.

Démonstration du lemme 18 $Q^{\cong}$*.1 Nous démontrons que, pour toute valuation propositionnelle v et valuation fonctionnelle sk,* $I^*(\exists x \exists y F)(v, sk) = I^*(\exists y \exists x F)(v, sk)$. *Nous n'examinons que le cas où la valuation fonctionnelle sk est telle que* $sk = \{(x \mapsto \hat{x}), (y \mapsto \hat{y})\} \cup sk'$ *; le cas où il n'existe ni* $(x \mapsto \hat{x})$ *ni* $(y \mapsto \hat{y})$ *dans sk est similaire à la démonstration de* $Q^{\equiv}$*.2 du lemme 13.*

$$\begin{array}{rl} & I^*(\exists x \exists y F)(v, sk) \\ = & I^*(F)(v[x := \hat{x}][y := \hat{y}], sk') \\ = & I^*(F)(v[y := \hat{y}][x := \hat{x}], sk') \\ = & I^*(\exists y \exists x F)(v, sk) \end{array}$$

$Q^{\cong}$*.2 Nous démontrons que* $I^*(\forall x \forall y F)(v, sk) = I^*(\forall y \forall x F)(v, sk)$ *pour toute valuation propositionnelle v et deux valuations fonctionnelles, sk et sk′, avec sk et sk′ identiques à ceci près que les deux premières colonnes sont permutées alors (en posant* $v_{xy} = v[x := \mathbf{vrai}][y := \mathbf{vrai}]$, $v_{x\overline{y}} = v[x := \mathbf{vrai}][y := \mathbf{faux}]$, $v_{\overline{x}y} = v[x := \mathbf{faux}][y := \mathbf{vrai}]$, $v_{\overline{xy}} = v[x := \mathbf{faux}][y := \mathbf{faux}]$, $v_{yx} = v[y := \mathbf{vrai}][x := \mathbf{vrai}]$, $v_{y\overline{x}} = v[y := \mathbf{vrai}][x := \mathbf{faux}]$, $v_{\overline{y}x} = v[y := \mathbf{faux}][x := \mathbf{vrai}]$, $v_{\overline{yx}} = v[y := \mathbf{faux}][x := \mathbf{faux}]$*)*

$$
\begin{array}{ll}
& I^*(\forall x \forall y F)(v, sk) \\
= & i_\wedge(i_\wedge(I^*(F)(v_{xy}, sk(\mathbf{vrai})(\mathbf{vrai})), I^*(F)(v_{x\overline{y}}, sk(\mathbf{vrai})(\mathbf{faux}))), \\
& i_\wedge(I^*(F)(v_{\overline{x}y}, sk(\mathbf{faux})(\mathbf{vrai})), I^*(F)(v_{\overline{xy}}, sk(\mathbf{faux})(\mathbf{faux})))) \\
= & i_\wedge(i_\wedge(I^*(F)(v_{yx}, sk(\mathbf{vrai})(\mathbf{vrai})), I^*(F)(v_{\overline{y}x}, sk(\mathbf{vrai})(\mathbf{faux}))), \\
& i_\wedge(I^*(F)(v_{y\overline{x}}, sk(\mathbf{faux})(\mathbf{vrai})), I^*(F)(v_{\overline{yx}}, sk(\mathbf{faux})(\mathbf{faux})))) \\
= & i_\wedge(i_\wedge(I^*(F)(v_{yx}, sk(\mathbf{vrai})(\mathbf{vrai})), I^*(F)(v_{y\overline{x}}, sk(\mathbf{faux})(\mathbf{vrai}))), \\
& i_\wedge(I^*(F)(v_{\overline{y}x}, sk(\mathbf{vrai})(\mathbf{faux})), I^*(F)(v_{\overline{yx}}, sk(\mathbf{faux})(\mathbf{faux})))) \\
= & i_\wedge(i_\wedge(I^*(F)(v_{yx}, sk'(\mathbf{vrai})(\mathbf{vrai})), I^*(F)(v_{y\overline{x}}, sk'(\mathbf{vrai})(\mathbf{faux}))), \\
& i_\wedge(I^*(F)(v_{\overline{y}x}, sk'(\mathbf{faux})(\mathbf{vrai})), I^*(F)(v_{\overline{yx}}, sk'(\mathbf{faux})(\mathbf{faux})))) \\
= & I^*(\forall y \forall x F)(v, sk')
\end{array}
$$

$Q^{\cong}.3$ *S'il n'existe pas de* $(x \mapsto \hat{x}) \in sk$ *alors*

$$
\begin{array}{ll}
& I^*(\exists x Q(x \wedge H))(v, sk) \\
= & i_\vee(I^*(Q(x \wedge H))(v[x := \mathbf{vrai}], sk), I^*(Q(x \wedge H))(v[x := \mathbf{faux}], sk))
\end{array}
$$

Par le lemme technique 1 :

$$
\begin{array}{ll}
& I^*(Q(x \wedge H))(v[x := \mathbf{vrai}], sk) \\
= & i_\wedge(I^*(x)(v[x := \mathbf{vrai}], sk), I^*(QH)(v[x := \mathbf{vrai}], sk)) \\
= & i_\wedge(I^*(x)(v[x := \mathbf{vrai}], sk), I^*([x \leftarrow \top](QH))(v, sk)) \textit{ par le lemme 6} \\
= & i_\wedge(I^*(x)(v[x := \mathbf{vrai}], sk), I^*(Q[x \leftarrow \top](H))(v, sk))
\end{array}
$$

et $I^*(Q(x \wedge H))(v[x := \mathbf{faux}], sk) = i_\wedge(\mathbf{faux}, I^*(QH)(v[x := \mathbf{faux}], sk)) = \mathbf{faux}$. *Donc*

$$
\begin{array}{ll}
& I^*(\exists x Q(x \wedge H))(v, sk) \\
= & i_\wedge(I^*(x)(v[x := \mathbf{vrai}], sk), I^*(Q[x \leftarrow \top](H))(v, sk)) \\
= & i_\wedge(I^*(x)(v[x := \mathbf{vrai}], sk), I^*(Q[x \leftarrow \top](H))(v[x := \mathbf{vrai}], sk)).
\end{array}
$$

De plus

$$
\begin{array}{ll}
& I^*(\exists x Q(x \wedge [x \leftarrow \top](H)))(v, sk) \\
= & i_\vee(I^*(Q(x \wedge [x \leftarrow \top](H)))(v[x := \mathbf{vrai}], sk), \\
& I^*(Q(x \wedge [x \leftarrow \top](H)))(v[x := \mathbf{faux}], sk))
\end{array}
$$

or par le lemme technique 1

$$
I^*(Q(x \wedge [x \leftarrow \top](H)))(v[x := \mathbf{faux}], sk) = \mathbf{faux}
$$

donc

$$
\begin{array}{ll}
& I^*(\exists x Q(x \wedge [x \leftarrow \top](H)))(v, sk) \\
= & I^*(Q(x \wedge [x \leftarrow \top](H)))(v[x := \mathbf{vrai}], sk) \\
= & i_\wedge(I^*(x)(v[x := \mathbf{vrai}], sk), I^*(Q[x \leftarrow \top](H))(v, sk)) \textit{ par le lemme technique 1}
\end{array}
$$

donc $I^*(\exists x Q(x \wedge H))(v, sk) = I^*(\exists x Q(x \wedge [x \leftarrow \top](H)))(v, sk)$.

S'il existe $sk = \{(x \mapsto \hat{x})\} \cup sk'$ *alors*

$$\begin{array}{ll} & I^*(\exists x Q(x \wedge H))(v, sk) \\ = & I^*(Q(x \wedge H))(v[x := \hat{x}], sk') \\ = & i_\wedge(v[x := \hat{x}](x), I^*(QH)(v[x := \hat{x}], sk')) \text{ par le lemme technique 1} \end{array}$$

De même

$$\begin{array}{ll} & I^*(\exists x Q(x \wedge [x \leftarrow \top](H)))(v, sk) \\ = & I^*(Q(x \wedge [x \leftarrow \top](H)))(v[x := \hat{x}], sk') \\ = & i_\wedge(v[x := \hat{x}](x), I^*(Q[x \leftarrow \top](H))(v[x := \hat{x}], sk')) \text{ par le lemme technique 1} \\ = & i_\wedge(v[x := \hat{x}](x), I^*(Q[x \leftarrow \top](H))(v, sk')) \end{array}$$

Maintenant, si $\hat{x} = \mathbf{faux}$ *alors*

$$I^*(\exists x Q(x \wedge H))(v, sk) = \mathbf{faux} = I^*(\exists x Q(x \wedge [x \leftarrow \top](H)))(v, sk).$$

sinon

$$I^*(\exists x Q(x \wedge H))(v, sk) = I^*(QH)(v[x := \mathbf{vrai}], sk')$$

et

$$I^*(\exists x Q(x \wedge [x \leftarrow \top](H)))(v, sk) = I^*(Q[x \leftarrow \top](H))(v, sk')$$

et par le lemme 6

$$I^*(\exists x Q(x \wedge H))(v, sk) = I^*(\exists x Q(x \wedge [x \leftarrow \top](H)))(v, sk).$$

$Q^{\cong}.4$ *La démonstration est similaire à celle de* $Q^{\cong}.3$.
$Q^{\cong}.5$

$$\begin{array}{ll} & I^*(\forall x Q(x \vee H))(v, sk) \\ = & i_\wedge(I^*(Q(x \vee H))(v[x := \mathbf{vrai}], sk(\mathbf{vrai})), I^*(Q(x \vee H))(v[x := \mathbf{faux}], sk(\mathbf{faux}))) \end{array}$$

Par le lemme technique 1

$$\begin{array}{ll} & I^*(Q(x \vee H))(v[x := \mathbf{faux}], sk(\mathbf{faux})) \\ = & i_\vee(v[x := \mathbf{faux}](x), I^*(QH)(v[x := \mathbf{faux}], sk(\mathbf{faux}))) \\ = & i_\vee(\mathbf{faux}, I^*([x \leftarrow \bot](QH))(v, sk(\mathbf{faux}))) \text{ par le lemme 6} \\ = & I^*(Q[x \leftarrow \bot](H))(v, sk(\mathbf{faux})) \end{array}$$

et $I^*(Q(x \vee H))(v[x := \mathbf{vrai}], sk(\mathbf{vrai})) = i_\wedge(\mathbf{vrai}, I^*(QH)(v[x := \mathbf{vrai}], sk(\mathbf{vrai}))) = \mathbf{vrai}$.
Donc $I^*(\forall x Q(x \vee H))(v, sk) = I^*(Q[x \leftarrow \bot](H))(v, sk(\mathbf{faux}))$.
$Q^{\cong}.6$ *La démonstration est similaire à celle de* $Q^{\cong}.5$.
$Q^{\cong}.7$ *&* $Q^{\cong}.8$ *Les résultats* $Q^{\cong}.7$ *et* $Q^{\cong}.8$ *sont des conséquences directes de* $Q^{\cong}.5$ *et* $Q^{\cong}.6$.

Lemme 19 *Soient* F *et* F' *deux QBF sans quantificateur,* G *une QBF prénexe,* x *un symbole propositionnel. Si* $F \equiv F'$ *alors* $[x \leftarrow F](G) \cong [x \leftarrow F'](G)$.

Démonstration du lemme 19 *Par récurrence sur les quantificateurs par le lemme 17 pour le cas de récurrence puis pour le cas d'arrêt par la congruence des connecteurs par rapport à l'équivalence logique.*

Lemme 20 *Soient G une QBF prénexe, x un symbole propositionnel, F une QBF librement substituable pour x dans G. Alors $(\exists x\ ((x{\leftrightarrow}F){\wedge}G)) \equiv [x \leftarrow F](G)$.*

Démonstration du lemme 20

$$
\begin{array}{ll}
& I^*(\exists x((x{\leftrightarrow}F){\wedge}G))(v) \\
= & i_\vee(I^*(((x{\leftrightarrow}F){\wedge}G))(v[x := \mathbf{vrai}]), I^*(((x{\leftrightarrow}F){\wedge}G))(v[x := \mathbf{faux}]))
\end{array}
$$

Si $I^*(F)(v) = \mathbf{vrai}$ *alors*

$$
\begin{array}{ll}
& I^*(\exists x((x{\leftrightarrow}F){\wedge}G))(v) \\
= & i_\vee(I^*(((x{\leftrightarrow}F){\wedge}G))(v[x := \mathbf{vrai}]), I^*(((x{\leftrightarrow}F){\wedge}G))(v[x := \mathbf{faux}])) \\
= & i_\vee(i_\wedge(I^*((x{\leftrightarrow}F))(v[x := \mathbf{vrai}]), I^*(G)(v[x := \mathbf{vrai}])), \\
& i_\wedge(I^*((x{\leftrightarrow}F))(v[x := \mathbf{faux}]), I^*(G)(v[x := \mathbf{faux}]))) \\
= & i_\vee(i_\wedge(i_\leftrightarrow(I^*(x)(v[x := \mathbf{vrai}]), I^*(F)(v[x := \mathbf{vrai}])), I^*(G)(v[x := \mathbf{vrai}])), \\
& i_\wedge(i_\leftrightarrow(I^*(x)(v[x := \mathbf{faux}]), I^*(F)(v[x := \mathbf{faux}])), I^*(G)(v[x := \mathbf{faux}]))) \\
= & i_\vee(i_\wedge(i_\leftrightarrow(\mathbf{vrai}, \mathbf{vrai}), I^*(G)(v[x := \mathbf{vrai}])), \\
& i_\wedge(i_\leftrightarrow(\mathbf{faux}, \mathbf{vrai}), I^*(G)(v[x := \mathbf{faux}]))) \\
= & i_\vee(i_\wedge(\mathbf{vrai}, I^*(G)(v[x := \mathbf{vrai}])), i_\wedge(\mathbf{faux}, I^*(G)(v[x := \mathbf{faux}]))) \\
= & i_\vee(I^*(G)(v[x := \mathbf{vrai}]), \mathbf{faux}) \\
= & I^*(G)(v[x := \mathbf{vrai}]) \\
= & I^*(G)(v[x := I^*(F)(v)]) \\
& \textit{par le lemme 2} \\
= & I^*([x \leftarrow F](G))(v)
\end{array}
$$

Si $I^*(F)(v) = \mathbf{faux}$ *alors*

$$
\begin{array}{ll}
& I^*(\exists x((x{\leftrightarrow}F){\wedge}G))(v) \\
= & i_\vee(I^*(((x{\leftrightarrow}F){\wedge}G))(v[x := \mathbf{vrai}]), I^*(((x{\leftrightarrow}F){\wedge}G))(v[x := \mathbf{faux}])) \\
= & i_\vee(i_\wedge(I^*((x{\leftrightarrow}F))(v[x := \mathbf{vrai}]), I^*(G)(v[x := \mathbf{vrai}])), \\
& i_\wedge(I^*((x{\leftrightarrow}F))(v[x := \mathbf{faux}]), I^*(G)(v[x := \mathbf{faux}]))) \\
= & i_\vee(i_\wedge(i_\leftrightarrow(I^*(x)(v[x := \mathbf{vrai}]), I^*(F)(v[x := \mathbf{vrai}])), I^*(G)(v[x := \mathbf{vrai}])), \\
& i_\wedge(i_\leftrightarrow(I^*(x)(v[x := \mathbf{faux}]), I^*(F)(v[x := \mathbf{faux}])), I^*(G)(v[x := \mathbf{faux}]))) \\
= & i_\vee(i_\wedge(i_\leftrightarrow(\mathbf{vrai}, \mathbf{faux})I^*(G)(v[x := \mathbf{vrai}]), ,) \\
& i_\wedge(i_\leftrightarrow(\mathbf{faux}, \mathbf{faux}), I^*(G)(v[x := \mathbf{faux}]))) \\
= & i_\vee(i_\wedge(\mathbf{faux}, I^*(G)(v[x := \mathbf{vrai}])), i_\wedge(\mathbf{vrai}, I^*(G)(v[x := \mathbf{faux}]))) \\
= & i_\vee(\mathbf{faux}, I^*(G)(v[x := \mathbf{faux}])) \\
= & I^*(G)(v[x := \mathbf{faux}]) \\
= & I^*(G)(v[x := I^*(F)(v)]) \\
& \textit{par le lemme 2} \\
= & I^*([x \leftarrow F](G))(v)
\end{array}
$$

Lemme 21 (complétude de la forme prénexe et des formes normales pour l'équivalence qui préserve les modèles propositionnels) *Pour toute QBF et selon l'équivalence qui préserve les modèles propositionnels, il existe une QBF prénexe qui lui est équivalente, il existe une QBF sous FNN qui lui est équivalente, une QBF sous FND qui lui est équivalente et une QBF sous FNC qui lui est équivalente. De plus, la linéarisation termine et calcule une QBF équivalente sans bi-implication, ni ou exclusif ; la mise sous forme prénexe termine et calcule une QBF équivalente sous forme prénexe ; la mise sous FNN termine et calcule une QBF sous FNN équivalente ; la mise sous FND termine et calcule une QBF sous FND équivalente ; la mise sous FNC par la méthode « académique » termine et calcule une QBF sous FNC équivalente ; la mise sous FNC par la méthode par « renommage de formules » termine et calcule une QBF sous FNC équivalente.*

Démonstration du lemme 21 *La terminaison de la linéarisation, qui est obtenue grâce* équivalences logiques[(84)] *0.3 et 0.30, est démontrée par récurrence sur le nombre de bi-implications et de ou-exclusifs. La forme prénexe est obtenue par application de gauche à droite des équivalences logiques 1.3, 1.4, 1.5, 1.6, 1.9, 1.10, 1.11, 1.12 et 1.13 ; la terminaison de cette mise sous forme prénexe est démontrée par récurrence sur le nombre de connecteurs qui sépare la QBF d'une forme prénexe. La FNN est obtenue par application de gauche à droite des équivalences logiques 1.3, 1.4, 0.1, 0.4, 0.5 et 0.6 ; la terminaison de cette mise sous FNN est démontrée par récurrence sur le nombre de négations qui ne portent pas sur un symbole propositionnel. La FND est obtenue à partir d'une forme prénexe mise sous FNN par application de gauche à droite de l'équivalence logique 0.23 ; la terminaison de cette mise sous FND est démontrée par induction sur la structure. La FNC est obtenue à partir d'une forme prénexe mise sous FNN par application de gauche à droite de l'équivalence logique 0.24 ; la terminaison de cette mise sous FNC « académique » est démontrée par induction sur la structure.*

La correction de la mise sous FNC par « renommage de formules » par l'algorithme 1 est démontrée par récurrence sur le nombre k de connecteurs binaires de la FNN prénexe pour la propriété « si $(Q_\exists, D) = msfnc\_renommage\_formules(F, n)$, n entier quelconque alors $Q_\exists D \equiv F$ et est sous FNC ».

- $k = 0$ $msfnc\_renommage\_formules(M, n) = (\varepsilon, M)$ *donc* $\equiv \varepsilon M$ *et* M *est bien sous FNC.*
- $k > 0$ *Soient* G, x *et* y, *tels que* $|x|, |y| \in \mathcal{SP}$ *et* $F = [z_n \leftarrow (x \circ y)](G)$. *La QBF* G *est sous FNN prénexe donc, par hypothèse de récurrence alors* $Q_\exists D \equiv G$ *et est sous FNC, pour* $(Q_\exists, D) = msfnc\_renommage\_formules(G, n)$. *Si* $\circ = \wedge$ *alors* $cl = (z_n \vee \overline{x} \vee \overline{y}) \wedge (\neg z_n \vee x) \wedge (\neg z_n \vee y)$ *(la preuve est similaire pour* $\circ = \vee$*) donc* $Q_\exists \exists z_n (cl \wedge D)$ *est sous FNC. De plus* $Q_\exists \exists z_n (cl \wedge D) \equiv Q_\exists \exists z_n ((z_n \leftrightarrow (x \wedge y)) \wedge D)$ *par le lemme 13 car* $(z_n \vee \overline{x} \vee \overline{y}) \wedge (\neg z_n \vee x) \wedge (\neg z_n \vee y) \equiv (z_n \leftrightarrow (x \wedge y))$. *Enfin par le lemme 20,* $(z_n \leftrightarrow (x \wedge y))$ *librement substituable dans* D *pour* z_n, *donc* $\exists z_n ((z_n \leftrightarrow (x \wedge y)) \wedge D) \equiv [z_n \leftarrow (x \wedge y)](D)$ *et enfin par le lemme 13*

$$Q_\exists \exists z_n (cl \wedge D) \equiv Q_\exists \exists z_n ((z_n \leftrightarrow (x \wedge y)) \wedge D) \equiv Q_\exists [z_n \leftarrow (x \wedge y)](D) \equiv [z_n \leftarrow (x \wedge y)](G).$$

Lemme 22 (complétude des formes normales pour l'équivalence qui préserve les modèles QBF) *Pour toute QBF prénexe et selon l'équivalence qui préserve les*

modèles QBF, il existe une QBF sous FNN qui lui est équivalente, une QBF sous FND qui lui est équivalente et une QBF sous FNC qui lui est équivalente. De plus, la linéarisation termine et calcule une QBF équivalente sans bi-implication, ni ou exclusif ; la mise sous FNN termine et calcule une QBF sous FNN équivalente ; la mise sous FND termine et calcule une QBF sous FND équivalente ; la mise sous FNC par la méthode « académique » termine et calcule une QBF sous FNC équivalente.

Démonstration du lemme 22 *La terminaison de la linéarisation, qui est obtenue grâce* équivalences logiques(84) *0.3 et 0.30, est démontrée par récurrence sur le nombre de bi-implications et de ou-exclusifs. La FNN est obtenue par application de gauche à droite des équivalences logiques 0.1, 0.4, 0.5 et 0.6 ; la terminaison de cette mise sous FNN est démontrée par récurrence sur le nombre de négations qui ne portent pas sur un symbole propositionnel. La FND est obtenue à partir d'une FNN par application de gauche à droite de l'équivalence logique 0.23 ; la terminaison de cette mise sous FND est démontrée par induction sur la structure. La FNC est obtenue à partir d'une FNN par application de gauche à droite de l'équivalence logique 0.24 ; la terminaison de cette mise sous FNC « académique » est démontrée par induction sur la structure.*

Lemme 23 (complétude de la mise sous FNC par la méthode « par renommage de formules » pour l'équivalence qui préserve les modèles QBF) *Pour toute QBF prénexe et selon l'équivalence qui préserve les modèles QBF, la mise sous FNC par la méthode par « renommage des formules » termine et calcule une QBF sous FNC. Si cette QBF sous FNC admet un modèle QBF $(v, sk \cup sk')$ tel que sk' contient l'ensemble des couples $(e_i \mapsto \hat{e_i})$ des symboles propositionnels existentiellement quantifiés introduit par la méthode alors (v, sk) est un modèle QBF de la QBF initiale.*

Démonstration du lemme 23 *La démonstration est similaire à celle du lemme 21 en remplaçant le lemme 13 par le lemme 18 et le lemme 20 par le lemme 24.*

Lemme 24 *Soient G une QBF prénexe, x un symbole propositionnel, F une QBF sans quantificateur librement substituable pour x dans G, v une valuation propositionnelle et sk une valuation fonctionnelle qui ne contient pas de couple $(x \mapsto \hat{x})$. Si $(\exists x\ ((x \leftrightarrow F) \wedge G))$ admet pour modèle QBF $(v, sk \cup \{(x \mapsto \hat{x})\})$ alors $[x \leftarrow F](G)$ admet pour modèle QBF (v, sk). Si $[x \leftarrow F](G)$ admet pour modèle QBF (v, sk) alors il existe un booléen $\hat{x}$ telle que $(\exists x\ ((x \leftrightarrow F) \wedge G))$ admet pour modèle QBF $(v, sk \cup \{(x \mapsto \hat{x})\})$.*

Démonstration du lemme 24 *Si $(v, sk \cup \{(x \mapsto \hat{x})\})$ est un modèle.*

$$
\begin{aligned}
& I^*(\exists x((x \leftrightarrow F) \wedge G))(v, sk \cup \{(x \mapsto \hat{x})\}) \\
= \;& I^*(((x \leftrightarrow F) \wedge G))(v[x := \hat{x}], sk) \\
= \;& i_\wedge(I^*((x \leftrightarrow F))(v[x := \hat{x}], sk), I^*(G)(v[x := \hat{x}], sk)) \\
= \;& i_\wedge(i_\leftrightarrow(I^*(x)(v[x := \hat{x}], sk), I^*(F)(v[x := \hat{x}], sk)), I^*(G)(v[x := \hat{x}], sk))
\end{aligned}
$$

Si $I^*(F)(v[x := \hat{x}], sk) = \mathbf{vrai}$ *alors*

$$
\begin{array}{ll}
& i_\wedge(i_\leftrightarrow(I^*(x)(v[x := \hat{x}], sk), I^*(F)(v[x := \hat{x}], sk)), I^*(G)(v[x := \hat{x}], sk)) \\
= & i_\wedge(i_\leftrightarrow(\hat{x}, \mathbf{vrai}), I^*(G)(v[x := \hat{x}], sk)) \\
= & i_\wedge(\hat{x}, I^*(G)(v[x := \hat{x}], sk)) = \mathbf{vrai} \\
& \quad \textit{car } sk \cup \{(x \mapsto \hat{x})\} \textit{ est un modèle QBF donc } \hat{x} = \mathbf{vrai} \textit{ et} \\
= & I^*(G)(v[x := \mathbf{vrai}], sk) \\
= & I^*(G)(v[x := I^*(F)(v, sk)], sk) \\
& \quad \textit{par le lemme 6} \\
= & I^*([x \leftarrow F](G))(v, sk)
\end{array}
$$

Si $I^*(F)(v[x := \hat{x}], sk) = \mathbf{faux}$ *alors*

$$
\begin{array}{ll}
& i_\wedge(i_\leftrightarrow(I^*(x)(v[x := \hat{x}], sk), I^*(F)(v[x := \hat{x}], sk)), I^*(G)(v[x := \hat{x}], sk)) \\
= & i_\wedge(i_\leftrightarrow(\hat{x}, \mathbf{faux}), I^*(G)(v[x := \hat{x}], sk)) = \mathbf{vrai} \\
& \quad \textit{car } sk \cup \{(x \mapsto \hat{x})\} \textit{ est un modèle QBF donc } \hat{x} = \mathbf{faux} \textit{ et} \\
= & I^*(G)(v[x := \mathbf{faux}], sk) \\
= & I^*(G)(v[x := I^*(F)(v, sk)], sk) \\
& \quad \textit{par le lemme 6} \\
= & I^*([x \leftarrow F](G))(v, sk)
\end{array}
$$

Si (v, sk) *est un modèle. La preuve est similaire à celle du lemme 20.*

Lemme 25 (axiome et modèle propositionnel) *A tout axiome d'une QBF prénexe* QM *est associé un modèle (propositionnel) pour la matrice* M*.*

Démonstration du lemme 25 *Un axiome d'une QBF prénexe close* QM *est une relation de formules* R *non explicitement contradictoire contenant deux classes donc tout symbole propositionnel* x *du lieur est soit dans la classe de* $\top$*, soit dans celle de* $\overline{\top}$ *; ainsi, une valuation* v *peut être associée à tout axiome de la manière suivante* $v(x) = \mathbf{vrai}$ *si* $[x]_R = [\top]_R$ *et* $v(x) = \mathbf{faux}$ *si* $[x]_R = [\overline{\top}]_R$*.*

Pour tout élément $(F_X \rightarrow F_Y)$ *de* $sfe(M)$ *tel que* $[(F_X \rightarrow F_Y)]_R = [\top]_R$*,* $v^*((F_X \rightarrow F_Y)) \neq \mathbf{vrai}$ *que si* $[F_X] = [\overline{F_Y}] = [\top]$ *mais alors la règle 1 s'applique ce qui est contradictoire avec le fait que c'est un axiome.*

Pour tout élément $\overline{(F_X \rightarrow F_Y)}$ *de* $sfe(M)$ *tel que* $[\overline{(F_X \rightarrow F_Y)}]_R = [\top]_R$*,* $v^*(\overline{(F_X \rightarrow F_Y)}) \neq \mathbf{vrai}$ *que si* $[F_X] = [F_Y] = [\top]$ *ou* $[\overline{F_X}] = [F_Y] = [\top]$ *ou* $[\overline{F_X}] = [\overline{F_Y}] = [\top]$ *mais alors les règles, respectivement, 3, 1 et 2 s'appliquent ce qui est contradictoire avec le fait que c'est un axiome.*

Donc par induction sur la structure, pour tout F *de* $sfe(M)$ *si* $[F]_R = [\top]_R$ *alors* $v^*(F) = \mathbf{vrai}$ *sinon* $[F]_R = [\overline{\top}]_R$ *et* $v^*(F) = \mathbf{faux}$*. Donc* v *est un modèle de* M*.*

Lemme 26 (racine et QBF) *Pour toute QBF prénexe close* QM*,* $(Q\ ;\ Id_{QM}([M] = [\top]))^* \cong QM$*.*

Démonstration du lemme 26 *Pour toute QBF prénexe close QM,*

$$\begin{array}{ll}
& (Q\ ;\ Id_{QM}([M]=[\top]))^* \\
= & Q\bigwedge_{C\in Id_{QM}([M]=[\top])}(\bigwedge_{F,F'\in C}(F\leftrightarrow F')) \\
\cong & Q(M\leftrightarrow\top) \\
& \textit{par le lemme 19} \\
\cong & QM
\end{array}$$

Lemme technique 2 *Soient C_0 et C_1 deux ensembles de formules propositionnelles, F_0 et F_1 deux formules propositionnelles telles que $F_0 \in C_0$ et $F_1 \in C_1$ alors*

$$(\bigwedge_{F,F'\in C_0}(F\leftrightarrow F'))\wedge(\bigwedge_{F,F'\in C_1}(F\leftrightarrow F'))\wedge(F_0\leftrightarrow F_1) \equiv \bigwedge_{F,F'\in C_0\cup C_1}(F\leftrightarrow F')$$

Démonstration du lemme technique 2 *Ce lemme technique est une reformulation pour l'interprétation d'une relation de formules que si deux éléments de deux classes d'équivalence a priori distinctes sont égaux alors les deux classes n'en forment en réalité qu'une seule.*

Soient C_0 et C_1 deux ensembles de formules propositionnelles, F_0 et F_1 deux formules propositionnelles telles que $F_0 \in C_0$ et $F_1 \in C_1$ alors par récurrence sur la taille de C_0

$$(\bigwedge_{F,F'\in C_0}(F\leftrightarrow F'))\wedge(F_0\leftrightarrow F_1) \equiv \bigwedge_{F,F'\in C_0\cup\{F_0\}}(F\leftrightarrow F')$$

puis par récurrence sur la taille de C_1

$$(\bigwedge_{F,F'\in C_0}(F\leftrightarrow F'))\wedge(\bigwedge_{F,F'\in C_1}(F\leftrightarrow F'))\wedge(F_0\leftrightarrow F_1) \equiv \bigwedge_{F,F'\in C_0\cup C_1}(F\leftrightarrow F')$$

Lemme 27 (correction des règles de fusion) *Soit la relation de formules R' le résultat de l'application d'une règle de fusion sur une relation de formules R alors $R'^* \cong R^*$.*

Démonstration du lemme 27 *Par application du lemme technique 2, du lemme 17 et des équivalences suivantes selon les règles.*

1. $\frac{(Q\ ;\ [\overline{F_X}]=[\top],[\overline{F_Y}]=[\top])}{(Q\ ;\ [(F_X\rightarrow F_Y)]=[\overline{F_Y}])}$: $((X\rightarrow Y)\leftrightarrow\neg Y) \equiv ((X\leftrightarrow Y)\wedge\neg Y) \equiv ((\neg X\leftrightarrow\top)\wedge(\neg Y\leftrightarrow\top))$.
2. $\frac{(Q\ ;\ [F_X]=[\top],[F_Y]=[\top])}{(Q\ ;\ [(F_X\rightarrow F_Y)]=[\overline{F_X}])}$: $((X\rightarrow Y)\leftrightarrow X) \equiv ((X\leftrightarrow Y)\wedge X) \equiv ((X\leftrightarrow\top)\wedge(Y\leftrightarrow\top))$.
3. $\frac{(Q\ ;\ [F_X]=[\top],[\overline{F_Y}]=[\top])}{(Q\ ;\ [\overline{(F_X\rightarrow F_Y)}]=[\top])}$: $(\neg(X\rightarrow Y)\leftrightarrow\top) \equiv (X\wedge\neg Y) \equiv ((X\leftrightarrow\top)\wedge(\neg Y\leftrightarrow\top))$.
4. $\frac{(Q\ ;\ [(F_X\rightarrow F_Y)]=[\overline{F_X}])}{(Q\ ;\ [(F_X\rightarrow F_Y)],[\overline{F_Y}]=[\top])}$:

$$((X\rightarrow Y)\leftrightarrow F)\wedge(\neg Y\leftrightarrow\top) \equiv ((X\rightarrow Y)\leftrightarrow F)\wedge(\neg Y\leftrightarrow\top)\wedge((X\rightarrow Y)\leftrightarrow\neg X)$$

5. $\frac{(Q\ ;\ [(F_X\rightarrow F_Y)]=[\overline{F_X}])}{(Q\ ;\ [(F_X\rightarrow F_Y)],[F_X]=[\overline{F_Y}])}$:

$$((X\rightarrow Y)\leftrightarrow F)\wedge(\neg Y\leftrightarrow X) \equiv ((X\rightarrow Y)\leftrightarrow F)\wedge(\neg Y\leftrightarrow X)\wedge((X\rightarrow Y)\leftrightarrow\neg X)$$

6. $\frac{(Q\ ;\ [(F_X{\rightarrow}F_Y)]{=}[\top])}{(Q\ ;\ [(F_X{\rightarrow}F_Y)],[\overline{F_X}]{=}[\top])}$:

$$((X{\rightarrow}Y){\leftrightarrow}F){\wedge}(\neg X{\leftrightarrow}\top) \equiv ((X{\rightarrow}Y){\leftrightarrow}F){\wedge}(\neg X{\leftrightarrow}\top){\wedge}((X{\rightarrow}Y){\leftrightarrow}\top)$$

7. $\frac{(Q\ ;\ [(F_X{\rightarrow}F_Y)]{=}[\top])}{(Q\ ;\ [(F_X{\rightarrow}F_Y)],[F_Y]{=}[\top])}$:

$$((X{\rightarrow}Y){\leftrightarrow}F){\wedge}(Y{\leftrightarrow}\top) \equiv ((X{\rightarrow}Y){\leftrightarrow}F){\wedge}(Y{\leftrightarrow}\top){\wedge}((X{\rightarrow}Y){\leftrightarrow}\top)$$

8. $\frac{(Q\ ;\ [(F_X{\rightarrow}F_Y)]{=}[\top])}{(Q\ ;\ [(F_X{\rightarrow}F_Y)],[F_X]{=}[F_Y])}$:

$$((X{\rightarrow}Y){\leftrightarrow}F){\wedge}(X{\leftrightarrow}Y) \equiv ((X{\rightarrow}Y){\leftrightarrow}F){\wedge}(X{\leftrightarrow}Y){\wedge}((X{\rightarrow}Y){\leftrightarrow}\top)$$

9. $\frac{(Q\ ;\ [(F_X{\rightarrow}F_Y)]{=}[F_Y])}{(Q\ ;\ [(F_X{\rightarrow}F_Y)],[F_X]{=}[\top])}$:

$$((X{\rightarrow}Y){\leftrightarrow}F){\wedge}(X{\leftrightarrow}\top) \equiv ((X{\rightarrow}Y){\leftrightarrow}F){\wedge}(X{\leftrightarrow}\top){\wedge}((X{\rightarrow}Y){\leftrightarrow}Y)$$

Lemme technique 3 *Si la relation de formules $(\varepsilon\ ;\ P)$ pour une QBF prénexe close QM est un axiome alors il existe un modèle (propositionnel) pour la QBF $(\varepsilon\ ;\ P)^*$.*

Démonstration du lemme technique 3 *Un axiome d'une QBF prénexe close QM est une relation de formules R non explicitement contradictoire contenant deux classes donc tout symbole propositionnel x du lieur est soit dans la classe de $\top$, soit dans celle de $\overline{\top}$; ainsi, une valuation v peut être associée à tout axiome de la manière suivante $v(x) =$ **vrai** si $[x]_R = [\top]_R$ et $v(x) =$ **faux** si $[x]_R = [\overline{\top}]_R$.*

*Pour tout élément $(F_X{\rightarrow}F_Y)$ de $sfe(M)$ tel que $[(F_X{\rightarrow}F_Y)]_R = [\top]_R$, $v^*((F_X{\rightarrow}F_Y)) \neq$ **vrai** que si $[F_X] = [\overline{F_Y}] = [\top]$ mais alors la règle 1 s'applique ce qui est contradictoire avec le fait que c'est un axiome donc $v^*(((F_X{\rightarrow}F_Y){\leftrightarrow}\top)) =$ **vrai**.*

*Pour tout élément $\overline{(F_X{\rightarrow}F_Y)}$ de $sfe(M)$ tel que $[\overline{(F_X{\rightarrow}F_Y)}]_R = [\top]_R$, $v^*(\overline{(F_X{\rightarrow}F_Y)}) \neq$ **vrai** que si $[F_X] = [F_Y] = [\top]$ ou $[\overline{F_X}] = [F_Y] = [\top]$ ou $[\overline{F_X}] = [\overline{F_Y}] = [\top]$ mais alors les règles, respectivement, 3, 1 et 2 s'appliquent ce qui est contradictoire avec le fait que c'est un axiome donc $v^*((\overline{(F_X{\rightarrow}F_Y)}{\leftrightarrow}\top)) =$ **vrai**.*

*Donc par induction sur la structure, pour tout $F, F' \in [\top]$, $v^*((F{\leftrightarrow}F')) =$ **vrai** donc $v^*(\bigwedge_{F,F'\in[\top]}(F{\leftrightarrow}F')) =$ **vrai** et pour tout $F, F' \in [\overline{\top}]$, $v^*((F{\leftrightarrow}F')) =$ **vrai** donc $v^*((\bigwedge_{F,F'\in[\top]}(F{\leftrightarrow}F'){\wedge}\bigwedge_{F,F'\in[\overline{\top}]}(F{\leftrightarrow}F'))) =$ **vrai**. Donc v est un modèle de $(\varepsilon\ ;\ P)^*$.*

Démonstration du théorème 1 – *Correction du système QBF par rapport à la sémantique des QBF : « Si la relation de formules $Id_{QM}([M] = [\top])$ admet une preuve dans le système S_{QBF} alors la QBF prénexe QM est valide. » La démonstration est par induction sur la structure d'une preuve pour le propriété : « Si la relation de formules $(Q\ ;\ P)$ admet un arbre de déduction tel que toutes les feuilles soient des axiomes alors la QBF $(Q\ ;\ P)^*$ est valide. »*

Axiome : Par le lemme technique 3.

Double négation : Soit un arbre de déduction $\dfrac{\dfrac{\nabla}{(Q\ ;\ [F,\neg\neg F])}}{(Q\ ;\ [F],[\neg\neg F])}\neg\neg$ *tel que toutes les feuilles soient des axiomes alors par hypothèse d'induction pour l'arbre de déduction* $\dfrac{\nabla}{(Q\ ;\ [F,\neg\neg F])}$ *,* $(Q\ ;\ [F,\neg\neg F])^*$ *est valide donc, puisque* $F \equiv \neg\neg F$*, par lemme technique 2,* $(Q\ ;\ [F,\neg\neg F])^* \cong (Q\ ;\ [F],[\neg\neg F])^*$ *donc* $(Q\ ;\ [F],[\neg\neg F])^*$ *est valide.*

Règles de fusion : Soit un arbre de déduction $\dfrac{\dfrac{\nabla}{R'}}{R}$ *tel que toutes les feuilles soient des axiomes alors par hypothèse d'induction pour l'arbre de déduction* $\dfrac{\nabla}{R'}$ *,* R'^* *est valide donc, par le lemme 27,* R^* *est valide.*

$\exists\top$ *: Soit un arbre de déduction* $\dfrac{\dfrac{\nabla}{R'}}{R}$ *tel que toutes les feuilles soient des axiomes alors par hypothèse d'induction pour l'arbre de déduction* $\dfrac{\nabla}{R'}$ *,* R'^* *est valide donc, pour*

$$R'^* = Q \bigwedge_{C\in P\setminus[\top,x],[\overline{\top},x]}\left(\bigwedge_{F,F'\in C}(F\leftrightarrow F')\right)\wedge\left(\bigwedge_{F,F'\in[\top,x]}(F\leftrightarrow F')\right)\wedge\left(\bigwedge_{F,F'\in[\overline{\top},\overline{x}]}(F\leftrightarrow F')\right)$$

par le lemme technique 2,

$$R'^* \cong Q\ \ (x\leftrightarrow\top)\wedge\bigwedge_{C\in P\setminus[\top,x],[\overline{\top},\overline{x}]}\left(\bigwedge_{F,F'\in C}(F\leftrightarrow F')\right)\wedge\left(\bigwedge_{F,F'\in[\top]}(F\leftrightarrow F')\right)\wedge \left(\bigwedge_{F,F'\in[\overline{\top}]}(F\leftrightarrow F')\right)\wedge\left(\bigwedge_{F,F'\in[x]}(F\leftrightarrow F')\right)\wedge\left(\bigwedge_{F,F'\in[\overline{x}]}(F\leftrightarrow F')\right)$$

donc

$$R'^* \cong Q[x\leftarrow\top]\Big(\ \bigwedge_{C\in P\setminus[\top,x],[\overline{\top},\overline{x}]}\left(\bigwedge_{F,F'\in C}(F\leftrightarrow F')\right)\wedge\left(\bigwedge_{F,F'\in[\top]}(F\leftrightarrow F')\right)\wedge \left(\bigwedge_{F,F'\in[\overline{\top}]}(F\leftrightarrow F')\right)\wedge\left(\bigwedge_{F,F'\in[x]}(F\leftrightarrow F')\right)\wedge\left(\bigwedge_{F,F'\in[\overline{x}]}(F\leftrightarrow F')\right)\Big)$$

est valide donc R^* *est valide.*
La démonstration est similaire pour $\exists\bot$*.*

$\exists+$ *: La démonstration est similaire au cas* $\exists\top$ *et pour* $\exists-$*.*

$\forall$ *: Soit un arbre de déduction* $\dfrac{\dfrac{\nabla}{R'}\quad\dfrac{\nabla'}{R''}}{R}$ *tel que toutes les feuilles soient des axiomes alors par hypothèse d'induction pour les arbres de déduction* $\dfrac{\nabla}{R'}$ *et* $\dfrac{\nabla'}{R''}$ *,* R'^* *et* R''^* *sont valides donc, pour*

$$R'^* = Q \bigwedge_{C\in P\setminus[\top,x],[\overline{\top},\overline{x}]}\left(\bigwedge_{F,F'\in C}(F\leftrightarrow F')\right)\wedge\left(\bigwedge_{F,F'\in[\top,x]}(F\leftrightarrow F')\right)\wedge\left(\bigwedge_{F,F'\in[\overline{\top},\overline{x}]}(F\leftrightarrow F')\right)$$

et

$$R''^{*} = Q \bigwedge_{C\in P\setminus[\top,\overline{x}],[\overline{\top},x]}(\bigwedge_{F,F'\in C}(F\leftrightarrow F'))\wedge(\bigwedge_{F,F'\in[\top,\overline{x}]}(F\leftrightarrow F'))\wedge(\bigwedge_{F,F'\in[\overline{\top},x]}(F\leftrightarrow F'))$$

par le lemme technique 2,

$$R'^{*} \cong Q\ (x\leftrightarrow\top)\wedge\bigwedge_{C\in P\setminus[\top,x],[\overline{\top},\overline{x}]}(\bigwedge_{F,F'\in C}(F\leftrightarrow F'))\wedge(\bigwedge_{F,F'\in[\top]}(F\leftrightarrow F'))\wedge (\bigwedge_{F,F'\in[\overline{\top}]}(F\leftrightarrow F'))\wedge(\bigwedge_{F,F'\in[x]}(F\leftrightarrow F'))\wedge(\bigwedge_{F,F'\in[\overline{x}]}(F\leftrightarrow F'))$$

et

$$R''^{*} \cong Q\ (x\leftrightarrow\overline{\top})\wedge\bigwedge_{C\in P\setminus[\top,\overline{x}],[\overline{\top},x]}(\bigwedge_{F,F'\in C}(F\leftrightarrow F'))\wedge(\bigwedge_{F,F'\in[\top]}(F\leftrightarrow F'))\wedge (\bigwedge_{F,F'\in[\overline{\top}]}(F\leftrightarrow F'))\wedge(\bigwedge_{F,F'\in[x]}(F\leftrightarrow F'))\wedge(\bigwedge_{F,F'\in[\overline{x}]}(F\leftrightarrow F'))$$

donc

$$R'^{*} \cong Q[x\leftarrow\top](\ \bigwedge_{C\in P\setminus[\top,x],[\overline{\top},\overline{x}]}(\bigwedge_{F,F'\in C}(F\leftrightarrow F'))\wedge(\bigwedge_{F,F'\in[\top]}(F\leftrightarrow F'))\wedge (\bigwedge_{F,F'\in[\overline{\top}]}(F\leftrightarrow F'))\wedge(\bigwedge_{F,F'\in[x]}(F\leftrightarrow F'))\wedge(\bigwedge_{F,F'\in[\overline{x}]}(F\leftrightarrow F')))$$

et

$$R''^{*} \cong Q[x\leftarrow\bot](\ \bigwedge_{C\in P\setminus[\top,x],[\overline{\top},\overline{x}]}(\bigwedge_{F,F'\in C}(F\leftrightarrow F'))\wedge(\bigwedge_{F,F'\in[\top]}(F\leftrightarrow F'))\wedge (\bigwedge_{F,F'\in[\overline{\top}]}(F\leftrightarrow F'))\wedge(\bigwedge_{F,F'\in[x]}(F\leftrightarrow F'))\wedge(\bigwedge_{F,F'\in[\overline{x}]}(F\leftrightarrow F')))$$

donc, par le lemme 7, R est valide.

– *Complétude du système QBF par rapport à la sémantique des QBF : « Si une QBF prénexe QM est valide alors la relation de formules $Id_{QM}([M]=[\top])$ admet une preuve dans le système S_{QBF}. » Si une QBF prénexe est valide alors elle admet un modèle QBF par le lemme 8. Soit $(\emptyset, sk)$ le modèle QBF de QM, la preuve de racine $Id_{QM}([M]=[\top])$ se construit en trois phases : application à partir de la racine de la règle d'élimination des doubles négations jusqu'à saturation ; application des règles d'élimination des quantificateurs existentiel et universel selon l'ordre du lieur, en particulier pour le quantificateur existentiel associé au symbole propositionnel x : la règle $\exists\top$ si $sk = \{(x\mapsto \mathbf{vrai}),\ldots\}$ la règle $\exists\bot$ si $sk = \{(x\mapsto \mathbf{faux}),\ldots\}$;application des règles de fusion 4, 6, 7 et 9 qui sont complètes pour évaluer une expression booléenne.*

Lemme 28 (correction et complétude du système S_{QBF} enrichi par rapport à la sémantique des QBF) *Soit la relation de formules R' le résultat de l'application d'une nouvelle règle de fusion sur une relation de formules R alors $R'^{*} \cong R^{*}$. Une QBF QM est valide si et seulement si la relation de formules $Id_{QM}([M]=[\top])$ admet une preuve dans le système S_{QBF} enrichi.*

Démonstration du lemme 28 *Pour les règles 10, 11, 12, 13, 14 et 15 cela revient à démontrer, respectivement,*

$$(\exists yQ'\forall xQ''\ ;\ [y]=[\top],[(x\rightarrow y)]=[\top])^* \cong (\exists yQ'\forall xQ''\ ;\ [(x\rightarrow y)]=[\top])^*,$$
$$(\exists yQ'\forall xQ''\ ;\ [y]=[\top],[(x\rightarrow y)]=[y])^* \cong (\exists yQ'\forall xQ''\ ;\ [(x\rightarrow y)]=[y])^*,$$
$$(\exists xQ'\forall yQ''\ ;\ [x]=[\top],[(x\rightarrow y)]=[y])^* \cong (\exists xQ'\forall yQ''\ ;\ [(x\rightarrow y)]=[y])^*,$$
$$(\exists yQ'\forall xQ''\ ;\ [\overline{y}]=[\top],[(x\rightarrow y)]=[\overline{x}])^* \cong (\exists yQ'\forall xQ''\ ;\ [(x\rightarrow y)]=[\overline{x}])^*,$$
$$(\exists xQ'\forall yQ''\ ;\ [\overline{x}]=[\top],[(x\rightarrow y)]=[\top])^* \cong (\exists xQ'\forall yQ''\ ;\ [(x\rightarrow y)]=[\top])^* \text{ et}$$
$$(\exists xQ'\forall yQ''\ ;\ [\overline{x}]=[\top],[(x\rightarrow y)]=[\overline{x}])^* \cong (\exists xQ'\forall yQ''\ ;\ [(x\rightarrow y)]=[\overline{x}])^*$$

ce qui est immédiat par, respectivement,

$$\exists y\forall x((x\rightarrow y)\leftrightarrow\top) \cong \exists y\forall x(((x\rightarrow y)\leftrightarrow\top)\wedge(y\leftrightarrow\top)),$$
$$\exists y\forall x((x\rightarrow y)\leftrightarrow y) \cong \exists y\forall x(((x\rightarrow y)\leftrightarrow y)\wedge(y\leftrightarrow\top)),$$
$$\exists x\forall y((x\rightarrow y)\leftrightarrow y) \cong \exists x\forall y(((x\rightarrow y)\leftrightarrow y)\wedge(x\leftrightarrow\top)),$$
$$\exists x\forall y((x\rightarrow y)\leftrightarrow\neg x) \cong \exists x\forall y(((x\rightarrow y)\leftrightarrow y)\wedge(\neg y\leftrightarrow\top)),$$
$$\exists x\forall y((x\rightarrow y)\leftrightarrow\top) \cong \exists x\forall y(((x\rightarrow y)\leftrightarrow y)\wedge(\neg x\leftrightarrow\top)) \text{ et}$$
$$\exists x\forall y((x\rightarrow y)\leftrightarrow\overline{x}) \cong \exists x\forall y(((x\rightarrow y)\leftrightarrow y)\wedge(\neg x\leftrightarrow\top)).$$

Lemme 29 (correction et complétude du système S^a_{QBF} par rapport à la sémantique des QBF) *Une QBF prénexe close, QM, admet un modèle QBF, V, si et seulement si, $V \Rightarrow Id_{QM}([M]=[\top])$ admet une preuve dans le système S^a_{QBF}.*

Démonstration du lemme 29 *La démonstration est un corollaire immédiat du théorème 1 et du lemme 28. Pour les instances de règles qui ne modifient pas la valuation QBF, la prémisse et la conclusion sont équivalentes au sens de la préservation des modèles QBF. Pour les règles $\exists\top^a$, $\exists\bot^a$ et $\forall^a$, ce n'est que l'expression de la sémantique des quantificateurs. Pour les règles $\exists\top^a$, $\exists\bot^a$ ainsi que les règles 10^a, 11^a, 12^a, 13^a, 14^a et 15^a, l'argument est le même que pour le lemme 28.*

Lemme 30 *Soit Q un lieu, M_+ une conjonction des clauses contenant un symbole propositionnel x sous sa forme de littéral positif, M_- une conjonction des clauses contenant x sous sa forme de littéral négatif et M' une conjonction des clauses ne contenant pas x. Alors*

$$Q\exists x((x\vee M_+)\wedge(\neg x\vee M_-)\wedge M') \equiv Q((M_-\vee M_+)\wedge M')$$

et

$$Q\forall x((x\vee M_+)\wedge(\neg x\vee M_-)\wedge M') \equiv Q((M_-\wedge M_+)\wedge M')$$

Démonstration du lemme 30 *Soit Q un lieu, M_+ une conjonction des clauses contenant un symbole propositionnel x sous sa forme de littéral positif, M_- une conjonction des clauses contenant x sous sa forme de littéral négatif et M' une conjonction des clauses ne contenant pas x. Alors*

$$\begin{aligned}
&Q\exists x((x\vee M_+)\wedge(\neg x\vee M_-)\wedge M')\\
\equiv\ & Q([x\leftarrow\top](((x\vee M_+)\wedge(\neg x\vee M_-)\wedge M'))\vee[x\leftarrow\bot](((x\vee M_+)\wedge(\neg x\vee M_-)\wedge M')))\\
\equiv\ & Q((M_-\wedge M')\vee(M_+\wedge M'))\\
\equiv\ & Q((M_-\vee M_+)\wedge M')
\end{aligned}$$

et

$$
\begin{array}{l}
Q\forall x((x\vee M_+)\wedge(\neg x\vee M_-)\wedge M') \\
\equiv\ \ Q([x\leftarrow\top](((x\vee M_+)\wedge(\neg x\vee M_-)\wedge M'))\wedge[x\leftarrow\bot](((x\vee M_+)\wedge(\neg x\vee M_-)\wedge M'))) \\
\equiv\ \ Q((M_-\wedge M')\wedge(M_+\wedge M')) \\
\equiv\ \ Q((M_-\wedge M_+)\wedge M')
\end{array}
$$

Notation :
Par abus de notation nous définissons la fonction d'interprétation sur les gardes ainsi :

$$((P_1,N_1);\ldots;(P_n,N_n))^* = \bigwedge_{1\leq k\leq n}((\neg x_k\vee P_k)\wedge(x_k\vee N_k))$$

et l'opérateur $\otimes$ sur les gardes ainsi :

$$
\begin{array}{l}
((P_1,N_1);\ldots;(P_n,N_n))\otimes((P'_1,N'_1);\ldots;(P'_n,N'_n)) \\
=\ \ (\ ((P_1\vee P'_1),(N_1\vee N'_1)); \\
\qquad (\mathcal{P}_2\wedge(P'_2\vee\mathcal{X})\wedge(P_2\vee\mathcal{X}'),\ \mathcal{N}_2\wedge(N'_2\vee\mathcal{X})\wedge(N_2\vee\mathcal{X}'));\ldots; \\
\qquad (\mathcal{P}_n\wedge(P'_n\vee\mathcal{X})\wedge(P_n\vee\mathcal{X}'),\ \mathcal{N}_n\wedge(N'_n\vee\mathcal{X})\wedge(N_n\vee\mathcal{X}')))
\end{array}
$$

avec $\mathcal{X}=((\neg x_1\vee P_1)\wedge(x_1\vee N_1)), \mathcal{X}'=((\neg x_1\vee P'_1)\wedge(x_1\vee N'_1))$ et

$$((\mathcal{P}_2,\mathcal{N}_2);\ldots;(\mathcal{P}_n,\mathcal{N}_n)) = ((P_2,N_2);\ldots;(P_n,N_n))\otimes((P'_2,N'_2);\ldots;(P'_n,N'_n))$$

De plus Q_i est une notation pour $q_ix_i\ldots q_nx_n$ et Q^i pour $q_1x_1\ldots q_ix_i$

Démonstration du théorème 2 *Si une QBF prénexe, QM est non valide, une base littérale bl de lieur Q et telle que sa garde soit uniquement constituée de $(\bot,\bot)$ est elle que $bl^*\cong QM$.*

Par le lemme 22, pour toute QBF prénexe valide, QM, il existe une QBF sous FNC, QM_{fnc}, telle que $QM\cong QM_{fnc}$. Maintenant par récurrence sur le nombre de quantificateurs :

- *Si $Q=\varepsilon$ alors nécessairement $M_{fnc}=\top$ et $\langle Q\mid\top\rangle^*=\top\cong QM$.*
- *Soit $Q=Q'qx$ alors il existe trois formules propositionnelles P, N et M' ne contenant pas x et telles que $M_{fnc}=(((\neg x\vee P)\wedge(x\vee N))\wedge M')$ (si x n'est pas présent dans M_{fnc} alors $P=\top$ et $N=\top$) ; par récurrence sur la QBF $Q'M'$ qui est sous FNC, il existe une base littérale $bl=\langle Q'\mid G\rangle$ telle que $bl^*\cong Q'M'$ et $G^*\equiv M'$; donc $(G;(P,N))^*=(G^*\wedge((\neg x\vee P)\wedge(x\vee N)))\equiv M_{fnc}$ et, par le lemme 17,*

$$\langle Q'qx\mid G;(P,N)\rangle^* = Q'qx(G^*\wedge((\neg x\vee P)\wedge(x\vee N)))\cong QM_{fnc}\cong QM.$$

Démonstration du théorème 3 *Soit une base littérale bl telle que*

$$bl=(q_1x_1\ldots q_nx_n,(P_1,N_1);\ldots;(P_n,N_n))$$

et $bl^*=q_1x_1\ldots q_nx_nM$. La substitution $[x_1\leftarrow C_1]\ldots[x_i\leftarrow C_i]$ étant obtenue à partir d'un modèle QBF de bl^* et la base bl étant optimale : pour tout k, $1\leq k<i$ si $C_k=\top$ alors

$[x_1 \leftarrow C_1] \ldots [x_{k-1} \leftarrow C_{k-1}](P_k) \equiv \top$ *sinon* $C_k = \bot$ *et* $[x_1 \leftarrow C_1] \ldots [x_{k-1} \leftarrow C_{k-1}](N_k) \equiv \top$. *Mais alors, par la définition 37, si* $C_i = \top$ *(la démonstration est similaire pour* $C_i = \bot$*)*

$$[x_1 \leftarrow C_1] \ldots [x_{i-1} \leftarrow C_{i-1}](N_i) \equiv \top$$
si et seulement s'il existe un modèle pour
$$q_{i+1}x_{i+1} \ldots q_n x_n [x_1 \leftarrow C_1] \ldots [x_{i-1} \leftarrow C_{i-1}][x_i \leftarrow \bot](M).$$

Or N_i*, étant uniquement composé des symboles propositionnels* $x_1, \ldots, x_{i-1}$*, l'évaluer est polynomial en temps par rapport à la taille de la formule* N_i*.*

Lemme 31 (optimalité et minimalité) *Soit bl une base littérale optimale alors* bl^* *est une QBF minimale.*

Démonstration du lemme 31 *Si l'interprétation de la base littérale n'est pas minimale alors il existe un modèle (propositionnel) pour la matrice qui n'est pas dans un modèle QBF donc il existe une séquence de gardes satisfaites par ce modèle (propositionnel) mais au moins une de ces gardes n'aurait pas dû l'être puisque la base littérale est optimale.*

Lemme 32 *Soit* $\forall x QM$ *une QBF. Si* $bl_\top$ *est un* ***sat****-certificat pour* $Q[x \leftarrow \top](M)$ *et* $bl_\bot$ *est* ***sat****-certificat pour* $Q[x \leftarrow \bot](M)$ *alors* $(bl_\top \circ_x bl_\bot)$ *est un* ***sat****-certificat pour* $\forall x QM$*.*

Démonstration du lemme 32 *Soit* $\forall x_1 q_2 x_2 \ldots q_n x_n M$ *une QBF.*
Soit $bl_\top = \langle Q \mid (P_2, N_2); \ldots; (P_n, N_n)\rangle$ *un* ***sat****-certificat pour* $Q[x \leftarrow \top](M)$ *et* $bl_\bot = \langle Q \mid (P'_2, N'_2); \ldots; (P'_n, N'_n)\rangle$ *un* ***sat****-certificat pour* $Q[x \leftarrow \bot](M)$*. Si* x_i*,* $2 \leq i \leq n$*, est un symbole propositionnel universellement quantifié alors* $P_i = N_i = P'_i = N'_i = \top$ *et* $((\neg x_i \vee P_i) \wedge (x_i \vee P'_i)) \equiv \top$ *et* $((\neg x_i \vee N_i) \wedge (x_i \vee N'_i)) \equiv \top$*. Si* x_i*,* $2 \leq i \leq n$*, est un symbole propositionnel existentiellement quantifié avec sa fonction associée* $\hat{x}_i^\top$ *pour* $bl_\top$ *(la séquence des fonctions booléennes* $\hat{x}_i^\top$ *forme un modèle pour* $Q[x \leftarrow \top](M)$ *et l'unique modèle de* $bl_\top^*$*) et avec sa fonction associée* $\hat{x}_i^\bot$ *pour* $bl_\bot$ *(la séquence des fonctions booléennes* $\hat{x}_i^\bot$ *forme un modèle pour* $Q[x \leftarrow \bot](M)$ *et l'unique modèle de* $bl_\bot^*$ *) alors la valuation fonctionnelle constituée des* $\hat{x}_i$ *tels que* $\hat{x}_i(\mathbf{vrai}) = \hat{x}_i^\top$ *et* $\hat{x}_i(\mathbf{faux}) = \hat{x}_i^\bot$ *vérifient* $v^*(((\neg x_i \vee P_i) \wedge (x_i \vee P'_i))) = \mathbf{vrai}$ *et* $v^*(((\neg x_i \vee N_i) \wedge (x_i \vee N'_i))) = \mathbf{faux}$*, constituant un modèle pour* $\forall x_1 q_2 x_2 \ldots q_n x_n M$ *et l'unique modèle de l'interprétation de la base littérale* $(bl_\top \circ_{x_1} bl_\bot)$*.*

Démonstration du théorème 4 *La démonstration est par récurrence sur la longueur du lieur d'une base littérale.*

- Cas de Base : *Soit* $Q = \exists x$*.*
  - *Si* $M \equiv \top$ *alors* $certificat\_par\_recherche(\exists x, M)$ *retourne* $bl = \langle \exists x \mid (\top, \bot)\rangle$ *alors* $bl^* = \exists x((\neg x \vee \top) \wedge (x \vee \bot)) \cong \exists x x \equiv \exists x M$*.*
  - *Si* $M \equiv \bot$ *alors* $\exists x M$ *n'est pas valide et* $certificat\_par\_recherche(\exists x, M)$ *retourne* $bl = non\_valide$*.*
  - *Si* $M \equiv x$ *alors* $certificat\_par\_recherche(\exists x, M)$ *retourne* $bl = \langle \exists x \mid (\top, \bot)\rangle$ *alors* $bl^* = \exists x((\neg x \vee \top) \wedge (x \vee \bot)) \cong \exists x x \equiv \exists x M$*.*
  - *Si* $M \equiv \neg x$ *alors* $certificat\_par\_recherche(\exists x, M)$ *retourne* $bl = \langle \exists x \mid (\bot, \top)\rangle$ *alors* $bl^* = \exists x((\neg x \vee \bot) \wedge (x \vee \top)) \cong \exists x \neg x \equiv \exists x M$*.*

Maintenant soit $Q = \forall x$. *Si* $M \equiv \top$ *alors* $certificat\_par\_recherche(\forall x, M)$ *retourne* $bl = \langle \forall x \mid (\top, \top) \rangle$ *alors* $bl^* = \forall x((\neg x \vee \top) \wedge (x \vee \top)) \cong \forall x \top \cong \forall x M$. *Sinon* $\forall x M$ *n'est pas valide et* $certificat\_par\_recherche(\forall x, M)$ *retourne* $bl = non\_valide$.

– Cas d'induction : *Supposons que* $Q = \exists x Q'$. *Soit*

$$bl^+ = certificat\_par\_recherche(Q', [x \leftarrow \top](M))$$

et

$$bl^- = certificat\_par\_recherche(Q', [x \leftarrow \bot](M)).$$

– *Si* $bl^+ = non\_valide$ *et* $bl^- = non\_valide$ *alors* $certificat\_par\_recherche(\exists x Q', M)$ *retourne non_valide. Par hypothèses d'induction, les deux QBF* $Q'[x \leftarrow \top](M)$ *et* $Q'[x \leftarrow \bot](M)$ *ne sont pas valides et alors* $\exists x Q' M$ *n'est pas valide.*

– *Si* $bl^+ \neq non\_valide$ *(le cas pour* ($bl^+ = non\_valide$ *et* $bl^- \neq non\_valide$*) est similaire) alors par hypothèses d'induction* $bl^+ = certificat\_par\_recherche(Q', [x \leftarrow \top](M))$ *est un* ***sat****-certificat pour* $Q'[x \leftarrow \top](M)$ *et* $\langle \exists x Q' \mid (\top, \bot); grds(bl^+) \rangle$ *est un* ***sat****-certificat pour* $\exists x Q' M$.

Maintenant supposons que $Q = \forall x Q'$. *Si* $certificat\_par\_recherche(Q', [x \leftarrow \top](M))$ *retourne non_valide ou* $certificat\_par\_recherche(Q', [x \leftarrow \top](M))$ *retourne non_valide alors* $certificat\_par\_recherche(\forall x Q', M)$ *retourne non_valide. Par hypothèse d'induction une des deux QBF* $Q'[x \leftarrow \top](M)$ *ou* $Q'[x \leftarrow \bot](M)$ *n'est pas valide et alors* $\forall x Q' M$ *n'est pas valide.*

Sinon, par hypothèse d'induction $bl^+ = certificat\_par\_recherche(Q', [x \leftarrow \top](M))$ *est un* ***sat****-certificat pour* $Q'[x \leftarrow \top](M)$ *et* $bl^- = certificat\_par\_recherche(Q', [x \leftarrow \bot](M))$ *est un* ***sat****-certificat pour* $Q'[x \leftarrow \bot](M)$ *alors par le lemme 32* ($bl^+ \circ_x bl^-$) *est un* ***sat****-certificat pour* $\forall x Q' M$.

Lemme technique 4

$$\begin{aligned} &((A \vee ((P_1, N_1); \ldots; (P_n, N_n))^*) \wedge (B \vee ((P_1, N_1); \ldots; (P_n, N_n))^*)) \\ &\quad \equiv \quad (((P_1 \vee (A \wedge B)), (N_1 \vee (A \wedge B))); \ldots; ((P_n \vee (A \wedge B)), (N_n \vee (A \wedge B))))^* \end{aligned}$$

Démonstration du lemme technique 4

$$\begin{aligned} &((A \vee ((P_1, N_1); \ldots; (P_n, N_n))^*) \wedge (B \vee ((P_1, N_1); \ldots; (P_n, N_n))^*)) \\ &\quad \equiv \quad ((A \vee \textstyle\bigwedge_{1 \leq i \leq n} ((\neg x_i \vee P_i) \wedge (x_i \vee N_i))) \wedge (B \vee \bigwedge_{1 \leq i \leq n} ((\neg x_i \vee P_i) \wedge (x_i \vee N_i)))) \\ &\quad \equiv \quad ((A \wedge B) \vee \textstyle\bigwedge_{1 \leq i \leq n} ((\neg x_i \vee P_i) \wedge (x_i \vee N_i))) \\ &\quad \equiv \quad \textstyle\bigwedge_{1 \leq i \leq n} (((\neg x_i \vee P_i) \vee (A \wedge B)) \wedge ((x_i \vee N_i) \vee (A \wedge B))) \\ &\quad \equiv \quad \textstyle\bigwedge_{1 \leq i \leq n} ((\neg x_i \vee (P_i \vee (A \wedge B))) \wedge (x_i \vee (N_i \vee (A \wedge B)))) \\ &\quad \equiv \quad (((P_1 \vee (A \wedge B)), (N_1 \vee (A \wedge B))); \ldots; ((P_n \vee (A \wedge B)), (N_n \vee (A \wedge B))))^* \end{aligned}$$

Lemme technique 5

$$\begin{aligned} &(((P_1, N_1); \ldots; (P_n, N_n))^* \wedge ((P'_1, N'_1); \ldots; (P'_n, N'_n))^*) \\ &\quad \equiv \quad (((P_1 \wedge P'_1), (N_1 \wedge N'_1)); \ldots; ((P_n \wedge P'_n), (N_n \wedge N'_n)))^* \end{aligned}$$

Démonstration du lemme technique 5

$$\begin{array}{ll}
(((P_1,N_1);\ldots;(P_n,N_n))^*\wedge((P'_1,N'_1);\ldots;(P'_n,N'_n))^*) & \\
\quad\equiv & (\bigwedge_{1\leq i\leq n}((\neg x_i\vee P_i)\wedge(x_i\vee N_i))\wedge\bigwedge)_{1\leq i\leq n}((\neg x_i\vee P'_i)\wedge(x_i\vee N'_i)) \\
\quad\equiv & \bigwedge_{1\leq i\leq n}(((\neg x_i\vee P_i)\wedge(x_i\vee N_i))\wedge((\neg x_i\vee P'_i)\wedge(x_i\vee N'_i))) \\
\quad\equiv & \bigwedge_{1\leq i\leq n}(((\neg x_i\vee P_i)\wedge(\neg x_i\vee P'_i))\wedge((x_i\vee N_i)\wedge(x_i\vee N'_i))) \\
\quad\equiv & \bigwedge_{1\leq i\leq n}((\neg x_i\vee(P_i\wedge P'_i))\wedge(x_i\vee(N_i\wedge N'_i))) \\
\quad\equiv & (((P_1\wedge P'_1),(N_1\wedge N'_1));\ldots;((P_n\wedge P'_n),(N_n\wedge N'_n)))^*
\end{array}$$

Lemme 33 *Soient Q un lieur et $bl, bl' \in \mathcal{BL}_Q$ tels que $bl^* = QM$ et $bl'^* = QM'$ alors $(bl \otimes bl')^* = QM_\otimes$ avec $M_\otimes \equiv (M \vee M')$.*

Démonstration du lemme 33 *Le théorème est vérifié par définition lorsque le lieur est vide. Le théorème est une conséquence directe du lemme suivant : Soient Q un lieur non vide et $B, B' \in \mathcal{BL}_Q$ tels que $B = (Q, G)$ et $B' = (Q, G')$ alors $(G \otimes G')^* \equiv (G^* \vee G'^*)$.*

La démonstration de ce lemme est par induction.

– Cas de base : $Q = q_1x_1$.
Soient $B = (q_1x_1, (P_1, N_1))$ et $B' = (q_1x_1, (P'_1, N'_1))$. Alors

$$\begin{array}{ll}
((P_1,N_1)^*\vee(P'_1,N'_1)^*) & \\
\quad= & (((\neg x_1\vee P_1)\wedge(x_1\vee N_1))\vee((\neg x_1\vee P'_1)\wedge(x_1\vee N'_1))) \\
\quad\equiv & ((\neg x_1\vee(P_1\vee P'_1))\wedge(x_1\vee(N_1\vee N'_1)))
\end{array} \quad (1)$$

Par définition $((P_1, N_1) \otimes (P'_1, N'_1)) = ((P_1 \vee P'_1), (N_1 \vee N'_1))$ alors

$$\begin{array}{ll}
((P_1,N_1)\otimes(P'_1,N'_1))^* & \\
\quad= & ((\neg x_1\vee(P_1\vee P'_1))\wedge(x_1\vee(N_1\vee N'_1))) \\
\quad= & ((P_1,N_1)^*\vee(P'_1,N'_1)^*)
\end{array}$$

– Cas d'induction : $Q = q_1x_1 \ldots q_nx_n$.
Soient

$$B = (q_1x_1 \ldots q_nx_n, (P_1, N_1); \ldots; (P_n, N_n))$$

et

$$B' = (q_1x_1 \ldots q_nx_n, (P'_1, N'_1); \ldots; (P'_n, N'_n))$$

Alors par la définition 35

$$\begin{array}{ll}
((P_1,N_1)\ldots(P_n,N_n))^* & \\
\quad= & \bigwedge_{1\leq k\leq n}((\neg x_k\vee P_k)\wedge(x_k\vee N_k)) \\
\quad= & ((\neg x_1\vee P_1)\wedge(x_1\vee N_1))\wedge[(P_2,N_2);\ldots;(P_n,N_n)]^*
\end{array} \quad (2)$$

et

$$\begin{array}{ll}
((P'_1,N'_1)\ldots(P'_n,N'_n))^* & \\
\quad= & \bigwedge_{1\leq k\leq n}((\neg x_k\vee P'_k)\wedge(x_k\vee N'_k)) \\
\quad= & ((\neg x_1\vee P'_1)\wedge(x_1\vee N'_1))\wedge[(P'_2,N'_2);\ldots;(P'_n,N'_n)]^*
\end{array} \quad (3)$$

Alors de (2) et (3)

$$\begin{array}{rl} & (((P_1,N_1);\ldots;(P_n,N_n))^*\vee((P'_1,N'_1);\ldots;(P'_n,N'_n))^*) \\ \equiv & ((\neg x_1\vee(P_1\vee P'_1))\wedge(x_1\vee(N_1\vee N'_1)))\wedge \\ & ((\neg x_1\vee P_1)\vee[(P'_2,N'_2);\ldots;(P'_n,N'_n)]^*)\wedge \\ & ((x_1\vee N_1)\vee[(P'_2,N'_2);\ldots;(P'_n,N'_n)]^*)\wedge \\ & ((\neg x_1\vee P'_1)\vee[(P_2,N_2);\ldots;(P_n,N_n)]^*)\wedge \\ & ((x_1\vee N'_1)\vee[(P_2,N_2);\ldots;(P_n,N_n)]^*)\wedge \\ & ([(P_2,N_2);\ldots;(P_n,N_n)]^*\vee[(P'_2,N'_2);\ldots;(P'_n,N'_n)]^*) \end{array} \quad (4)$$

Soient

$$\mathcal{X}=((\neg x_1\vee P_1)\wedge(x_1\vee N_1)) \quad (5)$$

et

$$\mathcal{X}'=((\neg x_1\vee P'_1)\wedge(x_1\vee N'_1)) \quad (6)$$

Alors de (4), (5), (6) et du lemme 4

$$\begin{array}{rl} & (((P_1,N_1);\ldots;(P_n,N_n))^*\vee((P'_1,N'_1);\ldots;(P'_n,N'_n))^*) \\ \equiv & ((\neg x_1\vee(P_1\vee P'_1))\wedge(x_1\vee(N_1\vee N'_1)))\wedge \\ & [((P'_2\vee\mathcal{X}),(N'_2\vee\mathcal{X}));\ldots;((P'_n\vee\mathcal{X}),(N'_n\vee\mathcal{X}))]^*\wedge \\ & [((P_2\vee\mathcal{X}'),(N_2\vee\mathcal{X}'));\ldots;((P_n\vee\mathcal{X}'),(N_n\vee\mathcal{X}'))]^*\wedge \\ & ([(P_2,N_2);\ldots;(P_n,N_n)]^*\vee[(P'_2,N'_2);\ldots;(P'_n,N'_n)]^*) \end{array} \quad (7)$$

Soit

$$\begin{array}{rl} & (\mathcal{P}_2,\mathcal{N}_2);\ldots;(\mathcal{P}_n,\mathcal{N}_n) \\ = & ((P_2,N_2);\ldots;(P_n,N_n))\otimes((P'_2,N'_2);\ldots;(P'_n,N'_n)) \end{array}$$

Par hypothèse d'induction

$$\begin{array}{rl} & [(\mathcal{P}_2,\mathcal{N}_2);\ldots;(\mathcal{P}_n,\mathcal{N}_n)]^* \\ = & ([(P_2,N_2);\ldots;(P_n,N_n)]^*\vee[(P'_2,N'_2);\ldots;(P'_n,N'_n)]^*) \end{array} \quad (8)$$

Alors par (7), (8) et le lemme 5

$$\begin{array}{rl} & (((P_1,N_1);\ldots;(P_n,N_n))^*\vee((P'_1,N'_1);\ldots;(P'_n,N'_n))^*) \\ \equiv & ((\neg x_1\vee(P_1\vee P'_1))\wedge(x_1\vee(N_1\vee N'_1)))\wedge \\ & [((P'_2\vee\mathcal{X}),(N'_2\vee\mathcal{X}));\ldots;((P'_n\vee\mathcal{X}),(N'_n\vee\mathcal{X}))]^*\wedge \\ & [((P_2\vee\mathcal{X}'),(N_2\vee\mathcal{X}'));\ldots;((P_n\vee\mathcal{X}'),(N_n\vee\mathcal{X}'))]^*\wedge \\ & [(\mathcal{P}_2,\mathcal{N}_2);\ldots;(\mathcal{P}_n,\mathcal{N}_n)]^* \\ \equiv & ((\neg x_1\vee(P_1\vee P'_1))\wedge(x_1\vee(N_1\vee N'_1)))\wedge \\ & [(((((P'_2\vee\mathcal{X})\wedge(P_2\vee\mathcal{X}'))\wedge\mathcal{P}_2),(((N'_2\vee\mathcal{X})\wedge(N_2\vee\mathcal{X}'))\wedge\mathcal{N}_2));\ldots; \\ & ((((P'_n\vee\mathcal{X})\wedge(P_n\vee\mathcal{X}'))\wedge\mathcal{P}_n),(((N'_n\vee\mathcal{X})\wedge(N_n\vee\mathcal{X}'))\wedge\mathcal{N}_n))]^* \\ \equiv & (((P_1,N_1);\ldots;(P_n,N_n))\otimes((P'_1,N'_1);\ldots;(P'_n,N'_n)))^* \end{array}$$

Lemme 34 *Soient $Q=q_1x_1\ldots q_nx_n$ un lieur non vide et $bl=(Q,(P_1,N_1);\ldots;(P_n,N_n))$ et $bl'=(Q,(P'_1,N'_1);\ldots;(P'_n,N'_n))$ deux bases littérales.*
Si $(bl\otimes bl')=(Q,(P_1^\otimes,N_1^\otimes);\ldots;(P_n^\otimes,N_n^\otimes))$ alors pour tout i, $1\leq i\leq n$:

$$P_i^\otimes\equiv(P_i\vee P'_i)\wedge(P_i\vee\bigwedge_{1\leq j<i}((\neg x_j\vee P'_j)\wedge(x_j\vee N'_j)))\wedge(P'_i\vee\bigwedge_{1\leq j<i}((\neg x_j\vee P_j)\wedge(x_j\vee N_j)))$$

et

$$N_i^{\otimes} \equiv (N_i \vee N_i') \wedge (P_i \vee \bigwedge_{1 \leq j < i} ((\neg x_j \vee P_j') \wedge (x_j \vee N_j'))) \wedge (N_i' \vee \bigwedge_{1 \leq j < i} ((\neg x_j \vee P_j) \wedge (x_j \vee N_j))).$$

Démonstration du lemme 34 *La démonstration est par récurrence sur la taille du lieur.*

Cas de base : Le lemme est vérifié pour $n = 1$ par définition car

$$\bigwedge_{1 \leq j < i} ((\neg x_j \vee P_j') \wedge (x_j \vee N_j')) \equiv \top.$$

Cas de récurrence :

$$\begin{array}{rl} & (Q, (P_1, N_1); \ldots; (P_n, N_n)) \otimes (Q, (P_1', N_1'); \ldots; (P_n', N_n')) \\ = & (Q,\ ((P_1 \vee P_1'), (N_1 \vee N_1')); \\ & \quad (\mathcal{P}_2 \wedge (P_2' \vee \mathcal{X}) \wedge (P_2 \vee \mathcal{X}'),\ \mathcal{N}_2 \wedge (N_2' \vee \mathcal{X}) \wedge (N_2 \vee \mathcal{X}')); \ldots; \\ & \quad (\mathcal{P}_n \wedge (P_n' \vee \mathcal{X}) \wedge (P_n \vee \mathcal{X}'),\ \mathcal{N}_n \wedge (N_n' \vee \mathcal{X}) \wedge (N_n \vee \mathcal{X}'))) \end{array}$$

avec $\mathcal{X} = ((\neg x_1 \vee P_1) \wedge (x_1 \vee N_1)), \mathcal{X}' = ((\neg x_1 \vee P_1') \wedge (x_1 \vee N_1'))$
et $Q' = q_2 x_2 \ldots q_n x_n$

$$\begin{array}{l} (Q', (\mathcal{P}_2, \mathcal{N}_2); \ldots; (\mathcal{P}_n, \mathcal{N}_n)) = \\ \quad (Q', (P_2, N_2); \ldots; (P_n, N_n)) \otimes (Q', (P_2', N_2'); \ldots; (P_n', N_n')) \end{array}$$

Par hypothèse de récurrence pour tout i, $2 \leq i \leq n$:

$$\mathcal{P}_i = (P_i \vee P_i') \wedge (P_i \vee \bigwedge_{2 \leq j < i} ((\neg x_j \vee P_j') \wedge (x_j \vee N_j'))) \wedge (P_i' \vee \bigwedge_{2 \leq j < i} ((\neg x_j \vee P_j) \wedge (x_j \vee N_j)))$$

et

$$\mathcal{N}_i = (N_i \vee N_i') \wedge (P_i \vee \bigwedge_{2 \leq j < i} ((\neg x_j \vee P_j') \wedge (x_j \vee N_j'))) \wedge (N_i' \vee \bigwedge_{2 \leq j < i} ((\neg x_j \vee P_j) \wedge (x_j \vee N_j))).$$

Donc pour tout i, $1 \leq i \leq n$:

$$\begin{array}{ll} & P_i^{\oplus} \\ = & \mathcal{P}_i \wedge (P_i' \vee \mathcal{X}) \wedge (P_i \vee \mathcal{X}') \\ \equiv & (P_i \vee P_i') \wedge (P_i \vee \bigwedge_{1 \leq j < i} ((\neg x_j \vee P_j') \wedge (x_j \vee N_j'))) \wedge (P_i' \vee \bigwedge_{1 \leq j < i} ((\neg x_j \vee P_j) \wedge (x_j \vee N_j))) \end{array}$$

et

$$\begin{array}{ll} & N_i^{\oplus} \\ = & \mathcal{N}_i \wedge (N_i' \vee \mathcal{X}) \wedge (N_i \vee \mathcal{X}') \\ \equiv & (N_i \vee N_i') \wedge (P_i \vee \bigwedge_{1 \leq j < i} ((\neg x_j \vee P_j') \wedge (x_j \vee N_j'))) \wedge (N_i' \vee \bigwedge_{1 \leq j < i} ((\neg x_j \vee P_j) \wedge (x_j \vee N_j))). \end{array}$$

Démonstration du théorème 5 *La démonstration est par récurrence sur le nombre de quantificateurs.*

- Cas de base :
  - *Supposons que* $q = \forall$.
    - *Si* $M \equiv \top$ *alors*

$$compilation\_par\_recherche(\forall x x, M) = \langle \forall x \mid (\top, \top) \rangle$$

alors

$$compilation\_par\_recherche(\forall x, M) = \forall x((\neg x \vee \top) \wedge (x \vee \top)) \cong \forall x \top \cong \forall x M$$

    - *Si* $M \equiv x$ *alors*

$$compilation\_par\_recherche(\forall x, M) = non\_valide$$

alors

$$compilation\_par\_recherche(\forall x, M)^* = \bot \cong \forall x x \cong \forall x M$$

    - *Si* $M \equiv \neg x$ *alors*

$$compilation\_par\_recherche(\forall x, M) = non\_valide$$

alors

$$compilation\_par\_recherche(\forall x, M)^* = \bot \cong \forall x \neg x \cong \forall x M$$

    - *Si* $M \equiv \bot$ *alors*

$$compilation\_par\_recherche(\forall x, M) = non\_valide$$

alors

$$compilation\_par\_recherche(\forall x, M)^* = \bot \cong \forall x \bot \cong \forall x M$$

  - *Supposons que* $q = \exists$.
    - *Si* $M \equiv \top$ *alors*

$$compilation\_par\_recherche(\exists x, M) = \langle \exists x \mid (\top, \top) \rangle$$

alors

$$compilation\_par\_recherche(\exists x, M) = \exists x((\neg x \vee \top) \wedge (x \vee \top)) \cong \exists x \top \cong \exists x M$$

    - *Si* $M \equiv x$ *alors*

$$compilation\_par\_recherche(\exists x, M) = \langle \exists x \mid (\top, \bot) \rangle$$

alors

$$compilation\_par\_recherche(\exists x, M)^* = \exists x((\neg x \vee \top) \wedge (x \vee \bot)) \cong \exists x x \cong \exists x M$$

– *Si* $M \equiv \neg x$ *alors*

$$compilation\_par\_recherche(\exists x, M) = non\_valide$$

alors

$$compilation\_par\_recherche(\exists x, M)^* = \exists x((\neg x \vee \bot) \wedge (x \vee \top)) \cong \exists x \neg x \cong \exists x M$$

– *Si* $M \equiv \bot$ *alors*

$$compilation\_par\_recherche(\exists x, M) = non\_valide$$

alors

$$compilation\_par\_recherche(\exists x, M)^* = \bot \cong \exists x \bot \cong \exists x M$$

– Cas d'induction : *Soit* $Q = qxQ'$ *et*

$$bl^+ = compilation\_par\_recherche(Q', [x \leftarrow \top](M))$$

et

$$bl^- = compilation\_par\_recherche(Q', [x \leftarrow \bot](M))$$

Par hypothèse d'induction,

$$(bl^+)^* \cong Q'[x \leftarrow \top](M) \tag{1}$$

and

$$(bl^-)^* \cong Q'[x \leftarrow \bot](M) \tag{2}$$

– *Nous supposons que* $(bl^+ = non\_valide)$ *et* $(bl^- = non\_valide)$*. Alors avec (1) et (2)*

$$Q'[x \leftarrow \top](M) \cong \bot$$

and

$$Q'[x \leftarrow \bot](M) \cong \bot$$

alors

$$compilation\_par\_recherche(Q, M)^* = \bot \cong QM.$$

– *Nous supposons que* $(bl^+ \neq non\_valide)$ *et* $(bl^- = non\_valide)$*. Si* $q = \forall$ *la démonstration est similaire au cas* $((bl^+ = non\_valide)$ *et* $(bl^- = non\_valide))$*. Sinon* $q = \exists$ *et de (2)*

$$Q'[x \leftarrow \bot](M) \cong \bot \tag{3}$$

Maintenant soit

$$\begin{array}{rcl} QM_+ & & \\ & = & \langle Q \mid (\top, \bot); grds(bl^+) \rangle^* \\ & = & compilation\_par\_recherche(Q, M)^* \end{array} \tag{4}$$

et

$$Q'M^+ = (bl^+)^* \tag{5}$$

Par la définition 35

$$M_+ = (\neg x \vee \top) \wedge (x \vee \bot) \wedge M^+$$

alors

$$[x \leftarrow \top](M_+) \equiv M^+ \text{ et } x \leftarrow \bot M_+ \equiv \bot$$

alors

$$Q'[x \leftarrow \top](M_+) \cong Q'M^+ \tag{6}$$

et

$$Q'[x \leftarrow \bot](M_+) \cong \bot \tag{7}$$

alors avec (6), (5) et (1)

$$Q'[x \leftarrow \top](M_+) \cong Q'[x \leftarrow \top](M) \tag{8}$$

alors avec (7) et (3)

$$QM_+ \cong QM$$

alors avec (4)

$$compilation\_par\_recherche(Q, M)^* \cong QM.$$

– *Nous supposons que* $(bl^+ = non\_valide)$ *et* $(bl^- \neq non\_valide)$.
La cas est similaire au cas précédent.

– *nous supposons que* $(bl^+ \neq non\_valide)$ *et* $(bl^- \neq non\_valide)$.
Soit

$$QM_+ = \langle Q \mid (\top, \bot); grds(bl^+) \rangle^* \tag{9}$$

$$QM_- = \langle Q \mid (\bot, \top); grds(bl^-) \rangle^* \tag{10}$$

et

$$\begin{array}{rl} QM_\otimes & \\ = & \langle Q \mid (\top, \bot); grds(bl^+) \rangle \otimes \langle Q \mid (\bot, \top); grds(bl^-) \rangle^* \\ = & compilation\_par\_recherche(Q, M)^* \end{array} \tag{11}$$

Par le lemme 33,

$$M_\otimes \equiv (M_+ \vee M_-) \tag{12}$$

Soit

$$Q'M^+ = (bl^+)^* \tag{13}$$

et

$$Q'M^- = (bl^-)^* \tag{14}$$

Par la définition 35, (9) et (13)

$$M_+ = (\neg x \vee \top) \wedge (x \vee \bot) \wedge M^+$$

alors

$$[x \leftarrow \top](M_+) \equiv M^+ \tag{15}$$

et

$$[x \leftarrow \bot](M_+) \equiv \bot \tag{16}$$

Par la définition 35, (10) et (14)

$$M_- = (\neg x \vee \bot) \wedge (x \vee \top) \wedge M^-$$

alors

$$[x \leftarrow \top](M_-) \equiv \bot \tag{17}$$

et

$$[x \leftarrow \bot](M_-) \equiv M^- \tag{18}$$

Alors par (12), (15) et (17)

$$[x \leftarrow \top](M_\otimes) \equiv M^+ \tag{19}$$

et par (12), (16) et (18)

$$[x \leftarrow \bot](M_\otimes) \equiv M^- \tag{20}$$

Alors de (19)

$$Q'[x \leftarrow \top](M_\otimes) \cong Q'M^+ \tag{21}$$

et de (20)

$$Q'x \leftarrow \bot M_\otimes \cong Q'M^- \tag{22}$$

Alors de (21), (13) et (1)

$$Q'[x \leftarrow \top](M_\otimes) \cong Q'[x \leftarrow \top](M)$$

et de (22), (14) et (2)

$$Q'x \leftarrow \bot M_\otimes \cong Q'x \leftarrow \bot M$$

alors

$$QM_\otimes \cong QM$$

alors de (11)

$$compilation\_par\_recherche(Q, M)^* \cong QM.$$

Lemme technique 6 *Soit $Q_1 M$ une QBF prénexe valide et*

$$\langle Q_1 \mid (P_1, N_1); \ldots; (P_n, N_n) \rangle = compilation\_par\_recherche(Q_1, M).$$

Soient i, $1 \leq i \leq n$, et $\sigma_{i-1} = [x_1 \leftarrow C_1] \ldots [x_{i-1} \leftarrow C_{i-1}]$ une substitution telle que pour tout k, $1 \leq k < i$, si $C_k = \top$ alors $[x_1 \leftarrow C_1] \ldots [x_{k-1} \leftarrow C_{k-1}](P_k) \equiv \top$ sinon $[x_1 \leftarrow C_1] \ldots [x_{k-1} \leftarrow C_{k-1}](N_k) \equiv \top$.

Alors

$$\begin{aligned} & compilation\_par\_recherche(Q_i, \sigma_{i-1}(M))^* \\ \cong \;& Q_i(\textstyle\bigwedge_{i \leq j \leq n} ((\neg x_j \vee \sigma_{i-1}(P_j)) \wedge (x_j \vee \sigma_{i-1}(N_j)))) \end{aligned}$$

Démonstration du lemme technique 6 *Par le théorème 5,*

$$\begin{array}{rl} & compilation\_par\_recherche(Q_1, M)^* \\ = & Q_1(\bigwedge_{1\leq j\leq n}((\neg x_j \vee P_j)\wedge(x_j \vee N_j))) \\ \cong & Q_1 M \end{array}$$

Alors

$$Q_i\sigma_{i-1}(\bigwedge_{1\leq j\leq n}((\neg x_j \vee P_j)\wedge(x_j \vee N_j))) \cong Q_i\sigma_{i-1}(M) \tag{23}$$

Mais, par la définition de σ_{i-1}

$$\begin{array}{rl} & \sigma_{i-1}(\bigwedge_{1\leq j\leq n}((\neg x_j \vee P_j)\wedge(x_j \vee N_j))) \\ \equiv & \bigwedge_{i\leq j\leq n}((\neg x_j \vee \sigma_{i-1}(P_j))\wedge(x_j \vee \sigma_{i-1}(N_j))) \end{array}$$

Alors

$$\begin{array}{rl} & Q_i\sigma_{i-1}(\bigwedge_{1\leq j\leq n}((\neg x_j \vee P_j)\wedge(x_j \vee N_j))) \\ \cong & Q_i \bigwedge_{i\leq j\leq n}((\neg x_j \vee \sigma_{i-1}(P_j))\wedge(x_j \vee \sigma_{i-1}(N_j))) \end{array} \tag{24}$$

Mais par le théoèrème 5

$$compilation\_par\_recherche(Q_i, \sigma_{i-1}(M))^* \cong Q_i\sigma_{i-1}(M)$$

Donc, par (23) et (24),

$$\begin{array}{rl} & compilation\_par\_recherche(Q_i, \sigma_{i-1}(M))^* \\ \cong & Q_i \bigwedge_{i\leq j\leq n}((\neg x_j \vee \sigma_{i-1}(P_j))\wedge(x_j \vee \sigma_{i-1}(N_j))) \end{array}$$

Lemme technique 7 *Soit* Q_1M *une QBF prénexe valide et*

$$\langle Q_1 \mid (P_1, N_1); \ldots; (P_n, N_n)\rangle = compilation\_par\_recherche(Q_1, M).$$

Soient i, $1 \leq i \leq n$ *et* $\sigma_{i-1} = [x_1 \leftarrow C_1]\ldots[x_{i-1} \leftarrow C_{i-1}]$ *une substitution telle que pour tout* k, $1 \leq k < i$, *si* $C_k = \top$ *alors* $[x_1 \leftarrow C_1]\ldots[x_{k-1} \leftarrow C_{k-1}](P_k) \equiv \top$ *sinon* $[x_1 \leftarrow C_1]\ldots[x_{k-1} \leftarrow C_{k-1}](N_k) \equiv \top$. *Soit*

$$\langle Q_i \mid (P_i^i, N_i^i); \ldots; (P_n^i, N_n^i)\rangle = compilation\_par\_recherche(Q_i, \sigma_{i-1}(M)).$$

Alors pour tout j, $i \leq j \leq n$, $\sigma_{i-1}(P_j) \equiv P_j^i$ *et* $\sigma_{i-1}(N_j) \equiv N_j^i$.

Démonstration du lemme technique 7 *Par le lemme technique 6,*

$$\begin{array}{rl} & compilation\_par\_recherche(Q_i, \sigma_{i-1}(M))^* \\ \cong & Q_i(\bigwedge_{i\leq k\leq n}((\neg x_k \vee \sigma_{i-1}(P_k))\wedge(x_k \vee \sigma_{i-1}(N_k)))) \end{array}$$

Par le théoreme 5,

$$\begin{array}{rl} & compilation\_par\_recherche(Q_i, \sigma_{i-1}(M))^* \\ = & Q_i(\bigwedge_{i\leq k\leq n}((\neg x_k \vee P_k^i)\wedge(x_k \vee N_k^i))) \end{array}$$

Alors, par induction, pour tout j, $i \leq j \leq n$, $\sigma_{i-1}(P_j) \equiv P_j^i$ *et* $\sigma_{i-1}(N_j) \equiv N_j^i$.

Démonstration du théorème 6 *Nous devons démontrer que : Soit Q_1M une QBF prénexe et*

$$\langle Q_1 \mid (P_1,N_1);\ldots;(P_n,N_n)\rangle = compilation\_par\_recherche(Q_1,M).$$

Pour tout i, $1 \leq i \leq n$, soit $\sigma_{i-1} = [x_1 \leftarrow C_1]\ldots[x_{i-1} \leftarrow C_{i-1}]$ une substitution telle que pour tout k, $1 \leq k < i$ si $C_k = \top$ alors $[x_1 \leftarrow C_1]\ldots[x_{k-1} \leftarrow C_{k-1}](P_k) \equiv \top$ sinon $[x_1 \leftarrow C_1]\ldots[x_{k-1} \leftarrow C_{k-1}](N_k) \equiv \top$.

Alors

$\sigma_{i-1}(P_i) \equiv \top$
si et seulement s'il existe un modèle QBF pour
$Q_{i+1}\sigma_{i-1}[x_i \leftarrow \top](\bigwedge_{1\leq k\leq n}((\neg x_k \vee P_k)\wedge(x_k \vee N_k)))$

et

$\sigma_{i-1}(N_i) \equiv \top$
si et seulement s'il existe un modèle QBF pour
$Q_{i+1}\sigma_{i-1}[x_i \leftarrow \bot](\bigwedge_{1\leq k\leq n}((\neg x_k \vee P_k)\wedge(x_k \vee N_k)))$.

Par les propriétés de σ_{i-1}, ces conséquences sont équivalentes à :

$\sigma_{i-1}(P_i) \equiv \top$
si et seulement s'il existe un modèle QBF pour
$Q_{i+1}\bigwedge_{i\leq k\leq n}((\neg x_k \vee \sigma_{i-1}[x_i \leftarrow \top](P_k))\wedge(x_k \vee \sigma_{i-1}[x_i \leftarrow \top](N_k)))$

et

$\sigma_{i-1}(N_i) \equiv \top$
si et seulement s'il existe un modèle QBF pour
$Q_{i+1}\bigwedge_{i\leq k\leq n}((\neg x_k \vee \sigma_{i-1}[x_i \leftarrow \bot](P_k))\wedge(x_k \vee \sigma_{i-1}[x_i \leftarrow \bot](N_k)))$.

Et, par le lemme technique 6, ces conséquences sont aussi équivalentes à

$\sigma_{i-1}(P_i) \equiv \top$
si et seulement s'il existe un modèle QBF pour
$compilation\_par\_recherche(Q_{i+1}, \sigma_{i-1}[x_i \leftarrow \top](M))$

et

$\sigma_{i-1}(N_i) \equiv \top$
si et seulement s'il existe un modèle QBF pour
$compilation\_par\_recherche(Q_{i+1}, \sigma_{i-1}[x_i \leftarrow \bot](M))$.

Nous démontrons pour $[x_i \leftarrow \bot]$, la démonstration pour $\top$ est similaire.
Par le théorème 5,

$$Q_iM \cong Q_1 \bigwedge_{1\leq k\leq n}((\neg x_k \vee P_k)\wedge(x_k \vee N_k))$$

alors

$$Q_i\sigma_{i-1}(M) \cong Q_1\sigma_{i-1}(\bigwedge_{1\leq k\leq n}((\neg x_k \vee P_k)\wedge(x_k \vee N_k)))$$

alors, par la définition de σ_{i-1}*,*

$$\begin{array}{rl} & Q_i\sigma_{i-1}(M) \\ \cong & Q_1((\neg x_i \vee \sigma_{i-1}(P_i)) \wedge (x_i \vee \sigma_{i-1}(N_i))) \\ \wedge & \bigwedge_{i+1 \leq k \leq n} ((\neg x_k \vee \sigma_{i-1}(P_k)) \wedge (x_k \vee \sigma_{i-1}(N_k))) \end{array}$$

alors il existe un modèle QBF pour $Q_1\sigma_{i-1}[x_i \leftarrow \bot](M)$ *si et seulement s'il existe un modèle QBF pour*

$$Q_i(\sigma_{i-1}(N_i) \wedge \bigwedge_{i+1 \leq k \leq n} ((\neg x_k \vee \sigma_{i-1}[x_i \leftarrow \bot](P_k)) \wedge (x_k \vee \sigma_{i-1}[x_i \leftarrow \bot](N_k))))$$

si et seulement si $\sigma_{i-1}(N_i) \equiv \top$ *et il existe un modèle QBF pour*

$$Q_{i+1} \bigwedge_{i+1 \leq k \leq n} ((\neg x_k \vee \sigma_{i-1}[x_i \leftarrow \bot](P_k)) \wedge (x_k \vee \sigma_{i-1}[x_i \leftarrow \bot](N_k)))$$

alors si $\sigma_{i-1}(N_i) \not\equiv \top$ *alors il existe un modèle QBF pour* $Q_{i+1}\sigma_{i-1}[x_i \leftarrow \bot](M)$ *alors il existe un modèle QBF pour* $Q_{i+1}\sigma_{i-1}[x_i \leftarrow \bot](M)$ *alors* $\sigma_{i-1}(N_i) \equiv \top$*.*

Maintenant s'il n'existe pas de modèle QBF $Q_{i+1}\sigma_{i-1}[x_i \leftarrow \bot](M)$ *alors, par le lemme technique 7 et la définition de l'algorithme compilation_par_recherche,* $\sigma_{i-1}(N_i) = \bot$*.*

Démonstration du théorème 7 – Cas de base :

– Cas d'induction : *Par définition,*

$$\begin{array}{rl} & compilation\_par\_elimination(Q^n, (((\neg x_n \vee P_n) \wedge (x_n \vee N_n)) \wedge M')) \\ = & \langle Q^n \mid (P_1, N_1); \ldots; (P_n, N_n) \rangle \end{array}$$

donc

$$\begin{array}{rl} & compilation\_par\_elimination(Q^n, (((\neg x_n \vee P_n) \wedge (x_n \vee N_n)) \wedge M'))^* \\ = & Q^n[\bigwedge_{1 \leq i \leq n-1} ((\neg x_i \vee P_i) \wedge (x_i \vee N_i))] \wedge ((\neg x_n \vee P_n) \wedge (x_n \vee N_n)) \end{array} \quad (1)$$

et

$$\begin{array}{rl} & compilation\_par\_elimination(Q^{n-1}, ((P_n \vee N_n) \wedge M')) \\ = & (Q^{n-1}, (P_1, N_1); \ldots; (P_{n-1}, N_{n-1})) \end{array}$$

et

$$\begin{array}{rl} & compilation\_par\_elimination(Q^{n-1}, ((P_n \vee N_n) \wedge M'))^* \\ = & Q^{n-1} \bigwedge_{1 \leq i \leq n-1} ((\neg x_i \vee P_i) \wedge (x_i \vee N_i)) \end{array} \quad (2)$$

– Supposons que $q_n = \exists$*. Soit,* $sk \cup \{(x_n \mapsto \hat{x_n})\}$*, (la composante fonctionnelle d')un modèle de la QBF* $Q^{n-1}\exists x_n(M' \wedge ((\neg x_n \vee P_n) \wedge (x_n \vee N_n)))$*, donc* $sk \cup \{(x_n \mapsto \hat{x_n})\}$ *est aussi un modèle de* $Q^{n-1}\exists x_n M'$ *et*

$$sk \cup \{(x_n \mapsto \hat{x_n})\} \text{ est aussi un modèle de } Q^{n-1}\exists x_n((\neg x_n \vee P_n) \wedge (x_n \vee N_n)) \quad (3)$$

Donc sk est un modèle pour $Q^{n-1}M'$ *et, par les définitions 19 et 20 de la sémantique du quantificateur existentiel, sk est un modèle pour* $Q^{n-1}(P_n \vee N_n)$*. Donc sk est un modèle pour* $Q^{n-1}(M' \wedge (P_n \vee N_n))$ *or par hypothèse d'induction,*

$$\begin{array}{rl} & compilation\_par\_elimination(Q^{n-1}, (M' \wedge (P_n \vee N_n)))^* \\ \cong & Q^{n-1}(M' \wedge (P_n \vee N_n)) \end{array}$$

Donc sk est un modèle pour $compilation\_par\_elimination(Q^{n-1}, (M' \wedge (P_n \vee N_n)))^*$ *donc, par (2), sk est un modèle pour* $Q^{n-1} \bigwedge_{1 \leq i \leq n-1} ((\neg x_i \vee P_i) \wedge (x_i \vee N_i))$. *Or, par (3),* $sk \cup \{(x_n \mapsto \hat{x_n})\}$ *est un modèle pour*

$$Q^{n-1} \exists x_n [\bigwedge_{1 \leq i \leq n-1} ((\neg x_i \vee P_i) \wedge (x_i \vee N_i))] \wedge ((\neg x_n \vee P_n) \wedge (x_n \vee N_n))$$

Donc par (1), $sk \cup \{(x_n \mapsto \hat{x_n})\}$ *est un modèle pour*

$$compilation\_par\_elimination(Q^{n-1} \exists x_n, (M' \wedge ((\neg x_n \vee P_n) \wedge (x_n \vee N_n))))^*$$

La réciproque est similaire.

– Supposons que $q_n = \forall$. *Ce cas est similaire au cas existentiel : la disjonction introduite par la sémantique du quantificateur existentiel est remplacée par une conjonction pour la sémantique du quantificateur universel ; de même la disjonction introduite par l'élimination de la variable existentiellement quantifiée dans l'algorithme est remplacée par une conjonction pour l'élimination de la variable universellement quantifiée.*

Démonstration du théorème 8 – Cas d'induction :

– Supposons que $q_n = \exists$. *Par hypothèse d'induction*

$$compilation\_par\_elimination(Q^{n-1}, (M' \wedge (P_n \vee N_n))) = \langle Q^{n-1} \mid (P_1, N_1); \ldots; (P_{n-1}, N_{n-1}) \rangle$$

est optimale :
Pour tout i, $1 \leq i \leq n-1$, *soit* $[x_1 \leftarrow C_1] \ldots [x_{i-1} \leftarrow C_{i-1}]$ *une substitution telle que pour tout* k, $1 \leq k < i$ *si* $C_k = \top$ *alors* $[x_1 \leftarrow C_1] \ldots [x_{k-1} \leftarrow C_{k-1}](P_k) \equiv \top$ *sinon* $[x_1 \leftarrow C_1] \ldots [x_{k-1} \leftarrow C_{k-1}](N_k) \equiv \top$.
Alors

$[x_1 \leftarrow C_1] \ldots [x_{i-1} \leftarrow C_{i-1}](P_i) \equiv \top$
si et seulement s'il existe un modèle QBF pour
$q_{i+1} x_{i+1} \ldots q_{n-1} x_{n-1} [x_1 \leftarrow C_1] \ldots [x_{i-1} \leftarrow C_{i-1}][x_i \leftarrow \top]((M' \wedge (P_n \vee N_n)))$

et

$[x_1 \leftarrow C_1] \ldots [x_{i-1} \leftarrow C_{i-1}](N_i) \equiv \top$
si et seulement s'il existe un modèle QBF pour
$q_{i+1} x_{i+1} \ldots q_{n-1} x_{n-1} [x_1 \leftarrow C_1] \ldots [x_{i-1} \leftarrow C_{i-1}][x_i \leftarrow \bot]((M' \wedge (P_n \vee N_n)))$.

Maintenant, par hypothèse sur la substitution $[x_1 \leftarrow C_1] \ldots [x_{n-1} \leftarrow C_{n-1}]$, *pour tout* i, $1 \leq i \leq n-1$ *(la QBF étant valide)*

$$q_{i+1} x_{i+1} \ldots q_{n-1} x_{n-1} [x_1 \leftarrow C_1] \ldots [x_i \leftarrow C_i]((M' \wedge (P_n \vee N_n))) \equiv \top$$

donc

$$q_{i+1} x_{i+1} \ldots q_{n-1} x_{n-1} [x_1 \leftarrow C_1] \ldots [x_i \leftarrow C_i]((M' \wedge \exists x_n ((\neg x_n \vee P_n) \wedge (x_n \vee N_n)))) \equiv \top$$

donc

$$q_{i+1}x_{i+1}\dots q_{n-1}x_{n-1}\exists x_n[x_1 \leftarrow C_1]\dots[x_i \leftarrow C_i]((M'\wedge((\neg x_n\vee P_n)\wedge(x_n\vee N_n)))) \equiv \top$$

De plus pour n, supposons que

$$[x_1 \leftarrow C_1]\dots[x_{n-1} \leftarrow C_{n-1}](P_n) \equiv \top$$

alors

$$[x_1 \leftarrow C_1]\dots[x_{n-1} \leftarrow C_{n-1}]((M'\wedge[x_n \leftarrow \top](((\neg x_n\vee P_n)\wedge(x_n\vee N_n))))) \equiv \top$$

donc, $x_n \notin \mathcal{SP}(M')$*,*

$$[x_1 \leftarrow C_1]\dots[x_{n-1} \leftarrow C_{n-1}][x_n \leftarrow \top]((M'\wedge((\neg x_n\vee P_n)\wedge(x_n\vee N_n)))) \equiv \top$$

La preuve est similaire avec $[x_1 \leftarrow C_1]\dots[x_{n-1} \leftarrow C_{n-1}](N_n) \equiv \top$*.*
Réciproquement pour i*,* $1 \leq i \leq n-1$*, si*

$$q_{i+1}x_{i+1}\dots q_{n-1}x_{n-1}\exists x_n[x_1 \leftarrow C_1]\dots[x_i \leftarrow C_i]((M'\wedge((\neg x_n\vee P_n)\wedge(x_n\vee N_n))))$$

admet un modèle alors, puisque $x_n \notin \mathcal{SP}(M')$*,*

$$q_{i+1}x_{i+1}\dots q_{n-1}x_{n-1}[x_1 \leftarrow C_1]\dots[x_i \leftarrow C_i]((M'\wedge\exists x_n((\neg x_n\vee P_n)\wedge(x_n\vee N_n)))) \equiv \top$$

alors

$$q_{i+1}x_{i+1}\dots q_{n-1}x_{n-1}[x_1 \leftarrow C_1]\dots[x_i \leftarrow C_i]((M'\wedge(P_n\vee N_n))) \equiv \top$$

donc par hypothèse d'induction si $C_i = \top$ *alors* $[x_1 \leftarrow C_1]\dots[x_{i-1} \leftarrow C_{i-1}](P_i) \equiv \top$ *sinon* $C_i = \bot$ *alors* $[x_1 \leftarrow C_1]\dots[x_{i-1} \leftarrow C_{i-1}](N_i) \equiv \top$*.*
Réciproquement pour n*,*

$$[x_1 \leftarrow C_1]\dots[x_{n-1} \leftarrow C_{n-1}][x_n \leftarrow \top]((M'\wedge((\neg x_n\vee P_n)\wedge(x_n\vee N_n)))) \equiv \top$$

donc, puisque $x_n \notin \mathcal{SP}(M') \cup \mathcal{SP}(P_n) \cup \mathcal{SP}(N_n)$*,*

$$[x_1 \leftarrow C_1]\dots[x_{n-1} \leftarrow C_{n-1}][x_n \leftarrow \top]((M'\wedge P_n)) \equiv \top$$

donc

$$[x_1 \leftarrow C_1]\dots[x_{n-1} \leftarrow C_{n-1}][x_n \leftarrow \top](P_n) \equiv \top$$

La preuve est similaire avec

$$[x_1 \leftarrow C_1]\dots[x_{n-1} \leftarrow C_{n-1}][x_n \leftarrow \bot](M'\wedge((\neg x_n\vee P_n)\wedge(x_n\vee N_n))) \equiv \top$$

– Supposons que $q_n = \forall$. *Par hypothèse d'induction*

$$compilation\_par\_elimination(Q^{n-1}, (M' \wedge (P_n \wedge N_n))) = \langle Q^{n-1} \mid (P_1, N_1); \ldots; (P_{n-1}, N_{n-1}) \rangle$$

est optimale :
Pour tout i, $1 \leq i \leq n-1$, *soit* $[x_1 \leftarrow C_1] \ldots [x_{i-1} \leftarrow C_{i-1}]$ *une substitution telle que pour tout* k, $1 \leq k < i$ *si* $C_k = \top$ *alors* $[x_1 \leftarrow C_1] \ldots [x_{k-1} \leftarrow C_{k-1}](P_k) \equiv \top$ *sinon* $[x_1 \leftarrow C_1] \ldots [x_{k-1} \leftarrow C_{k-1}](N_k)$).
Alors

$[x_1 \leftarrow C_1] \ldots [x_{i-1} \leftarrow C_{i-1}](P_i) \equiv \top$
si et seulement s'il existe un modèle QBF pour
$q_{i+1}x_{i+1} \ldots q_{n-1}x_{n-1}[x_1 \leftarrow C_1] \ldots [x_{i-1} \leftarrow C_{i-1}][x_i \leftarrow \top]((M' \wedge (P_n \wedge N_n)))$

et

$[x_1 \leftarrow C_1] \ldots [x_{i-1} \leftarrow C_{i-1}]N_i \equiv \top$
si et seulement si il existe un modèle QBF pour
$q_{i+1}x_{i+1} \ldots q_{n-1}x_{n-1}[x_1 \leftarrow C_1] \ldots [x_{i-1} \leftarrow C_{i-1}][x_i \leftarrow \bot]((M' \wedge (P_n \wedge N_n)))$.

Maintenant, par hypothèse sur $[x_1 \leftarrow C_1] \ldots [x_{n-1} \leftarrow C_{n-1}]$, *pour tout* i, $1 \leq i \leq n-1$ *(la QBF étant valide)*

$$q_{i+1}x_{i+1} \ldots q_{n-1}x_{n-1}[x_1 \leftarrow C_1] \ldots [x_i \leftarrow C_i]((M' \wedge (P_n \wedge N_n))) \equiv \top$$

donc

$$q_{i+1}x_{i+1} \ldots q_{n-1}x_{n-1}[x_1 \leftarrow C_1] \ldots [x_i \leftarrow C_i]((M' \wedge \forall x_n((\neg x_n \vee P_n) \wedge (x_n \wedge N_n)))) \equiv \top$$

donc

$$q_{i+1}x_{i+1} \ldots q_{n-1}x_{n-1}\forall x_n[x_1 \leftarrow C_1] \ldots [x_i \leftarrow C_i]((M' \wedge ((\neg x_n \vee P_n) \wedge (x_n \wedge N_n)))) \equiv \top$$

De plus pour n, *la QBF étant valide, par le théorème 7* $[x_1 \leftarrow C_1] \ldots [x_{n-1} \leftarrow C_{n-1}](P_n) \equiv \top$ *et* $[x_1 \leftarrow C_1] \ldots [x_{n-1} \leftarrow C_{n-1}](N_n) \equiv \top$

Lemme technique 8 *Soient* G *une QBF prénexe,* x *un symbole propositionnel,* F *une formule propositionnelle ne contenant pas* x, v *une valuation propositionnelle et* sk *une valuation fonctionnelle telle qu'il n'existe pas* $(x, \hat{x})$ *dans* sk. *Alors*

$$I^*(\exists x((x \leftrightarrow F) \wedge G))(v, sk) = I^*(G)(v[x := I^*(F)(v, sk)], sk)$$

Démonstration du lemme technique 8

$I^*(\exists x((x \leftrightarrow F) \wedge G))(v, sk)$
$= \quad i_\vee(I^*(((x \leftrightarrow F) \wedge G))(v[x := \mathbf{vrai}], sk), I^*(((x \leftrightarrow F) \wedge G))(v[x := \mathbf{faux}], sk))$

Si $I^*(F)(v, sk) = \mathbf{vrai}$ *alors*

$$
\begin{array}{ll}
& I^*(\exists x((x{\leftrightarrow}F){\wedge}G))(v, sk) \\
= & i_\vee(I^*(((x{\leftrightarrow}F){\wedge}G))(v[x := \mathbf{vrai}], sk), I^*(((x{\leftrightarrow}F){\wedge}G))(v[x := \mathbf{faux}], sk)) \\
= & i_\vee(i_\wedge(I^*((x{\leftrightarrow}F))(v[x := \mathbf{vrai}], sk), I^*(G)(v[x := \mathbf{vrai}], sk)), \\
& i_\wedge(I^*((x{\leftrightarrow}F))(v[x := \mathbf{faux}], sk), I^*(G)(v[x := \mathbf{faux}], sk))) \\
= & i_\vee(i_\wedge(i_\leftrightarrow(I^*(x)(v[x := \mathbf{vrai}], sk), I^*(F)(v[x := \mathbf{vrai}], sk)), I^*(G)(v[x := \mathbf{vrai}], sk)), \\
& i_\wedge(i_\leftrightarrow(I^*(x)(v[x := \mathbf{faux}], sk), I^*(F)(v[x := \mathbf{faux}], sk)), I^*(G)(v[x := \mathbf{faux}], sk))) \\
= & i_\vee(i_\wedge(i_\leftrightarrow(\mathbf{vrai}, \mathbf{vrai}), I^*(G)(v[x := \mathbf{vrai}], sk)), \\
& i_\wedge(i_\leftrightarrow(\mathbf{faux}, \mathbf{vrai}), I^*(G)(v[x := \mathbf{faux}], sk))) \\
= & i_\vee(i_\wedge(\mathbf{vrai}, I^*(G)(v[x := \mathbf{vrai}], sk)), i_\wedge(\mathbf{faux}, I^*(G)(v[x := \mathbf{faux}], sk))) \\
= & i_\vee(I^*(G)(v[x := \mathbf{vrai}], sk), \mathbf{faux}) \\
= & I^*(G)(v[x := \mathbf{vrai}], sk) \\
= & I^*(G)(v[x := I^*(F)(v, sk)], sk)
\end{array}
$$

Si $I^*(F)(v, sk) = \mathbf{faux}$ *alors*

$$
\begin{array}{ll}
& I^*(\exists x((x{\leftrightarrow}F){\wedge}G))(v, sk) \\
= & i_\vee(I^*(((x{\leftrightarrow}F){\wedge}G))(v[x := \mathbf{vrai}], sk), I^*(((x{\leftrightarrow}F){\wedge}G))(v[x := \mathbf{faux}], sk)) \\
= & i_\vee(i_\wedge(I^*((x{\leftrightarrow}F))(v[x := \mathbf{vrai}], sk), I^*(G)(v[x := \mathbf{vrai}], sk)), \\
& i_\wedge(I^*((x{\leftrightarrow}F))(v[x := \mathbf{faux}], sk), I^*(G)(v[x := \mathbf{faux}], sk))) \\
= & i_\vee(i_\wedge(i_\leftrightarrow(I^*(x)(v[x := \mathbf{vrai}], sk), I^*(F)(v[x := \mathbf{vrai}], sk)), I^*(G)(v[x := \mathbf{vrai}], sk)), \\
& i_\wedge(i_\leftrightarrow(I^*(x)(v[x := \mathbf{faux}], sk), I^*(F)(v[x := \mathbf{faux}], sk)), I^*(G)(v[x := \mathbf{faux}], sk))) \\
= & i_\vee(i_\wedge(i_\leftrightarrow(\mathbf{vrai}, \mathbf{faux})I^*(G)(v[x := \mathbf{vrai}], sk), ,) \\
& i_\wedge(i_\leftrightarrow(\mathbf{faux}, \mathbf{faux}), I^*(G)(v[x := \mathbf{faux}], sk))) \\
= & i_\vee(i_\wedge(\mathbf{faux}, I^*(G)(v[x := \mathbf{vrai}], sk)), i_\wedge(\mathbf{vrai}, I^*(G)(v[x := \mathbf{faux}], sk))) \\
= & i_\vee(\mathbf{faux}, I^*(G)(v[x := \mathbf{faux}], sk)) \\
= & I^*(G)(v[x := \mathbf{faux}], sk) \\
= & I^*(G)(v[x := I^*(F)(v, sk)], sk)
\end{array}
$$

Lemme 35 *Soit QM une QBF prénexe et* $(Q_\exists, D) = decomposition_\exists(M, 1)$.
Alors $QM \equiv QQ_\exists D$.

Démonstration du lemme 35 *Nous notons* $C_n = (d_n{\leftrightarrow}z_n)$ *les équivalences générées par l'algorithme à la* $n^{ème}$ *itération, de même en est-il des* G_n, D_n *et* Q_n *; le nombre de ces* C_n *est le nombre p d'occurrences d'opérateurs dans la formule qui est aussi le nombre d'applications de* $decomposition_\forall$. *Nous démontrons pour une QBF prénexe QF que pour tout* $n, 1 \leq n \leq p$, $QF \cong Q\exists z_n \ldots \exists z_1(C_1{\wedge}\ldots(C_n{\wedge}G_n))$.

Nous supposons par hypothèse d'induction que

$$F = \exists z_1 \ldots \exists z_n(C_1{\wedge}\ldots(C_n{\wedge}G_n)).$$

– *Si* $G_n = x$ *ou* $G_n = \neg x$ *alors le résultat est vérifié.*

– *Si* G_n *est telle que* $G_n = [z_{n+1} \leftarrow d_{n+1}](G_{n+1})$ *alors*

$$\begin{array}{ll} & (C_1 \wedge \ldots (C_n \wedge [z_{n+1} \leftarrow d_{n+1}](G_{n+1}))) \\ & \quad \textit{par le lemme technique 8} \\ \equiv & \exists z_{n+1}(C_{n+1} \wedge (C_1 \wedge \ldots (C_n \wedge G_{n+1}))) \\ \equiv & \exists z_{n+1}(C_1 \wedge \ldots (C_n \wedge (C_{n+1} \wedge G_{n+1}))) \end{array}$$

donc grâce à $Q^{\equiv}.1$

$$\begin{array}{ll} & \exists z_1 \ldots \exists z_n(C_1 \wedge \ldots (C_n \wedge G_n)) \\ = & \exists z_1 \ldots \exists z_n(C_1 \wedge \ldots (C_n \wedge [z_{n+1} \leftarrow d_{n+1}](G_{n+1}))) \\ \equiv & \exists z_1 \ldots \exists z_n \exists z_{n+1}(C_1 \wedge \ldots (C_n \wedge (C_{n+1} \wedge G_{n+1}))) \end{array}$$

Lemme technique 9 *Soient* G *une QBF prénexe,* x *un symbole propositionnel,* F *une formule propositionnelle ne contenant pas* x*,* v *une valuation propositionnelle et* sk *une valuation fonctionnelle. Alors*

$$I^*(\forall x((x \leftrightarrow F) \rightarrow G))(v, sk) = I^*(G)(v[x := I^*(F)(v, sk)], sk).$$

Démonstration du lemme technique 9

$$\begin{array}{ll} & I^*(\forall x((x \leftrightarrow F) \rightarrow G))(v, sk) \\ = & i_\wedge(I^*(((x \leftrightarrow F) \rightarrow G))(v[x := \mathbf{vrai}], sk), I^*(((x \leftrightarrow F) \rightarrow G))(v[x := \mathbf{faux}], sk)) \end{array}$$

Si $I^*(F)(v, sk) = \mathbf{vrai}$ *alors*

$$\begin{array}{ll} & I^*(\forall x((x \leftrightarrow F) \rightarrow G))(v, sk) \\ = & i_\wedge(I^*(((x \leftrightarrow F) \rightarrow G))(v[x := \mathbf{vrai}], sk), I^*(((x \leftrightarrow F) \rightarrow G))(v[x := \mathbf{faux}], sk)) \\ = & i_\wedge(i_\rightarrow(I^*((x \leftrightarrow F))(v[x := \mathbf{vrai}], sk), I^*(G)(v[x := \mathbf{vrai}], sk)), \\ & i_\rightarrow(I^*((x \leftrightarrow F))(v[x := \mathbf{faux}], sk), I^*(G)(v[x := \mathbf{faux}], sk))) \\ = & i_\wedge(i_\rightarrow(i_\leftrightarrow(I^*(x)(v[x := \mathbf{vrai}], sk), I^*(F)(v[x := \mathbf{vrai}], sk)), I^*(G)(v[x := \mathbf{vrai}], sk)), \\ & i_\rightarrow(i_\leftrightarrow(I^*(x)(v[x := \mathbf{faux}], sk), I^*(F)(v[x := \mathbf{faux}], sk)), I^*(G)(v[x := \mathbf{faux}], sk))) \\ = & i_\wedge(i_\rightarrow(i_\leftrightarrow(\mathbf{vrai}, \mathbf{vrai}), I^*(G)(v[x := \mathbf{vrai}], sk)), \\ & i_\rightarrow(i_\leftrightarrow(\mathbf{faux}, \mathbf{vrai}), I^*(G)(v[x := \mathbf{faux}], sk))) \\ = & i_\wedge(i_\rightarrow(\mathbf{vrai}, I^*(G)(v[x := \mathbf{vrai}], sk)), i_\rightarrow(\mathbf{faux}, I^*(G)(v[x := \mathbf{faux}], sk))) \\ = & i_\wedge(I^*(G)(v[x := \mathbf{vrai}], sk), \mathbf{vrai}) \\ = & I^*(G)(v[x := \mathbf{vrai}], sk) \\ = & I^*(G)(v[x := I^*(F)(v, sk)], sk) \end{array}$$

Si $I^*(F)(v, sk) = \mathbf{faux}$ *alors*

$$
\begin{array}{ll}
& I^*(\forall x((x{\leftrightarrow}F){\rightarrow}G))(v, sk) \\
= & i_\wedge(I^*(((x{\leftrightarrow}F){\rightarrow}G))(v[x := \mathbf{vrai}], sk), I^*(((x{\leftrightarrow}F){\rightarrow}G))(v[x := \mathbf{faux}], sk)) \\
= & i_\wedge(i_\rightarrow(I^*((x{\leftrightarrow}F))(v[x := \mathbf{vrai}], sk), I^*(G)(v[x := \mathbf{vrai}], sk)), \\
& i_\rightarrow(I^*((x{\leftrightarrow}F))(v[x := \mathbf{faux}], sk), I^*(G)(v[x := \mathbf{faux}], sk))) \\
= & i_\wedge(i_\rightarrow(i_\leftrightarrow(I^*(x)(v[x := \mathbf{vrai}], sk), I^*(F)(v[x := \mathbf{vrai}], sk)), I^*(G)(v[x := \mathbf{vrai}], sk)), \\
& i_\rightarrow(i_\leftrightarrow(I^*(x)(v[x := \mathbf{faux}], sk), I^*(F)(v[x := \mathbf{faux}], sk)), I^*(G)(v[x := \mathbf{faux}], sk))) \\
= & i_\wedge(i_\rightarrow(i_\leftrightarrow(\mathbf{vrai}, \mathbf{faux}), I^*(G)(v[x := \mathbf{vrai}], sk)), \\
& i_\rightarrow(i_\leftrightarrow(\mathbf{faux}, \mathbf{faux}), I^*(G)(v[x := \mathbf{faux}], sk))) \\
= & i_\wedge(i_\rightarrow(\mathbf{faux}, I^*(G)(v[x := \mathbf{vrai}], sk)), i_\rightarrow(\mathbf{vrai}, I^*(G)(v[x := \mathbf{faux}], sk))) \\
= & i_\rightarrow(\mathbf{vrai}, I^*(G)(v[x := \mathbf{faux}], sk)) \\
= & I^*(G)(v[x := \mathbf{faux}], sk) \\
= & I^*(G)(v[x := I^*(F)(v, sk)], sk)
\end{array}
$$

Lemme 36 *Soit l'algorithme de décomposition* $decomposition_\forall$ *ci-dessous :*
Entrée: *Une formule propositionnelle* F
Entrée: *Un entier* n
Sortie: *Une paire composée d'un lieur et d'une formule propositionnelle*
si $F = \top$ *ou* $F = \bot$ *ou* $F = \neg x$ *ou* $F = x$ *avec* $x \in \mathcal{SP}$ ***alors***
retourner (ε, F)
sinon
$F = [z_n \leftarrow (x \circ y)](G)$, *avec* $z_n, |x|, |y| \in \mathcal{SP}$
$(Q, D) = decomposition_\forall(G, n + 1)$
retourner $(Q \forall z_n, (((x \circ y){\leftrightarrow}z_n){\rightarrow}D))$
fin si
Soit QM *une QBF prénexe et* $(Q_\forall, D) = decomposition_\forall(M, 1)$.
Alors $QM \cong QQ_\forall D$.

Démonstration du lemme 36 *Nous notons* $C_n = (d_n{\leftrightarrow}z_n)$ *les équivalences générées par l'algorithme à la* $n^{ème}$ *itération, de même en est-il des* G_n, D_n *et* Q_n *; le nombre de ces* C_n *est le nombre* p *d'occurrences d'opérateurs dans la formule qui est aussi le nombre d'applications de* $decomposition_\forall$. *Nous démontrons pour une QBF prénexe* QF *que pour tout* $n, 1 \leq n \leq p$, $QF \cong Q \forall z_n \ldots \forall z_1(C_1{\rightarrow}\ldots(C_n{\rightarrow}G_n))$.

Nous supposons par hypothèse d'induction que

$$F = \forall z_1 \ldots \forall z_n(C_1{\rightarrow}\ldots(C_n{\rightarrow}G_n)).$$

- *Si* $G_n = x$ *ou* $G_n = \neg x$ *alors le résultat est vérifié.*
- *Si* G_n *est telle que* $G_n = [z_{n+1} \leftarrow d_{n+1}](G_{n+1})$ *alors*

$$
\begin{array}{ll}
& (C_1{\rightarrow}\ldots(C_n{\rightarrow}[z_{n+1} \leftarrow d_{n+1}](G_{n+1}))) \\
& \quad \textit{par les lemme technique 9} \\
\cong & \forall z_{n+1}(C_{n+1}{\rightarrow}(C_1{\rightarrow}\ldots(C_n{\rightarrow}G_{n+1}))) \\
\cong & \forall z_{n+1}(C_1{\rightarrow}\ldots(C_n{\rightarrow}(C_{n+1}{\rightarrow}G_{n+1})))
\end{array}
$$

donc grâce à $Q^{\cong}$.1

$$\begin{array}{ll} & \forall z_1 \ldots \forall z_n (C_1 \rightarrow \ldots (C_n \rightarrow G_n)) \\ = & \forall z_1 \ldots \forall z_n (C_1 \rightarrow \ldots (C_n \rightarrow [z_{n+1} \leftarrow d_{n+1}](G_{n+1}))) \\ \cong & \forall z_1 \ldots \forall z_n \forall z_{n+1} (C_1 \rightarrow \ldots (C_n \rightarrow (C_{n+1} \rightarrow G_{n+1}))) \end{array}$$

Lemme 37 *Pour toute QBF prénexe close F il existe une QBF prénexe close sous forme normale de contraintes G telle que $F \cong G$.*

Démonstration du lemme 37 *Corollaire direct des lemmes 35 et 36 et de l'équivalence logique :*

$$(A \rightarrow (B \rightarrow C)) \overset{0.1}{\equiv} (\neg A \vee (\neg B \vee C)) \overset{0.22}{\equiv} ((\neg A \vee \neg B) \vee C) \overset{0.6}{\equiv} (\neg (A \vee B) \vee C) \overset{0.1}{\equiv} ((A \wedge B) \rightarrow C)$$

Lemme technique 10 *1.* $I^*(\forall x Q(F \wedge G))(v, sk) = \mathbf{vrai}$ *si et seulement si*

$$I^*(\forall x Q F)(v, sk) = \mathbf{vrai} \text{ } et \text{ } I^*(\forall x Q G)(v, sk) = \mathbf{vrai}$$

2. si $I^*(\exists x Q(F \wedge G))(v, sk) = \mathbf{vrai}$ *alors*

$$I^*(\exists x Q F)(v, sk) = \mathbf{vrai} \text{ } et \text{ } I^*(\exists x Q G)(v, sk) = \mathbf{vrai}$$

3. $I^*(\exists x Q(F \vee G))(v, sk) = \mathbf{vrai}$ *si et seulement si*

$$I^*(\exists x Q F)(v, sk) = \mathbf{vrai} \text{ } ou \text{ } I^*(\exists x Q G)(v, sk) = \mathbf{vrai}$$

4. si

$$I^*(\forall x Q F)(v, sk) = \mathbf{vrai} \text{ } ou \text{ } I^*(\forall x Q G)(v, sk) = \mathbf{vrai}$$

alors $I^*(\forall x Q(F \vee G))(v, sk) = \mathbf{vrai}$

Démonstration du lemme technique 10 *1.* $I^*(\forall x Q(F \wedge G))(v, sk) = \mathbf{vrai}$ *si et seulement si*

$$i_\wedge(I^*((F \wedge G))(v[x := \mathbf{vrai}], sk(\mathbf{vrai})), I^*((F \wedge G))(v[x := \mathbf{faux}], sk(\mathbf{faux}))) = \mathbf{vrai}$$

si et seulement si

$$I^*((F \wedge G))(v[x := \mathbf{vrai}], sk(\mathbf{vrai})) = \mathbf{vrai} \text{ } et \text{ } I^*((F \wedge G))(v[x := \mathbf{faux}], sk(\mathbf{faux})) = \mathbf{vrai}$$

si et seulement si

$$\begin{array}{l} I^*(F)(v[x := \mathbf{vrai}], sk(\mathbf{vrai})) = \mathbf{vrai} \text{ } et \\ I^*(G)(v[x := \mathbf{vrai}], sk(\mathbf{vrai})) = \mathbf{vrai} \text{ } et \\ I^*(F)(v[x := \mathbf{faux}], sk(\mathbf{faux})) = \mathbf{vrai} \text{ } et \\ I^*(G)(v[x := \mathbf{faux}], sk(\mathbf{faux})) = \mathbf{vrai} \end{array}$$

si et seulement si $I^*(\forall x Q F)(v, sk) = \mathbf{vrai}$ *et* $I^*(\forall x Q G)(v, sk) = \mathbf{vrai}$.

2. *Soit* $I^*(\exists xQ(F \wedge G))(v, sk) = \mathbf{vrai}$.
   Si $sk = (x, \hat{x}) \cup sk'$ *alors*

$$I^*(Q(F \wedge G))(v[x := \hat{x}], sk) = \mathbf{vrai}$$

   alors

$$I^*(QF)(v[x := \hat{x}], sk) = \mathbf{vrai} \text{ } et \text{ } I^*(QG)(v[x := \hat{x}], sk) = \mathbf{vrai}$$

   alors $I^*(\exists xQF)(v, sk) = \mathbf{vrai}$ *et* $I^*(\exists xQG)(v, sk) = \mathbf{vrai}$.
   Sinon

$$\begin{array}{ll} & I^*(\exists xQ(F \wedge G))(v, sk) \\ = & i_\vee(I^*(Q(F \wedge G))(v[x := \mathbf{vrai}], sk), I^*(Q(F \wedge G))(v[x := \mathbf{faux}], sk)) \\ = & \mathbf{vrai} \end{array}$$

   si et seulement si

$$\begin{array}{l} (I^*(F)(v[x := \mathbf{vrai}], sk) = \mathbf{vrai} \text{ } et \\ I^*(G)(v[x := \mathbf{vrai}], sk) = \mathbf{vrai}) \text{ } ou \\ (I^*(F)(v[x := \mathbf{faux}], sk) = \mathbf{vrai} \text{ } et \\ I^*(G)(v[x := \mathbf{faux}], sk) = \mathbf{vrai}) \end{array}$$

   si

$$\begin{array}{l} (I^*(F)(v[x := \mathbf{vrai}], sk) = \mathbf{vrai} \text{ } ou \\ I^*(F)(v[x := \mathbf{vrai}], sk) = \mathbf{vrai}) \text{ } et \\ (I^*(G)(v[x := \mathbf{faux}], sk) = \mathbf{vrai} \text{ } ou \\ I^*(G)(v[x := \mathbf{faux}], sk) = \mathbf{vrai}) \end{array}$$

   si et seulement si

$$I^*(\exists xQF)(v, sk) = \mathbf{vrai} \text{ } et \text{ } I^*(\exists xQG)(v, sk) = \mathbf{vrai}$$

3. *La démonstration est similaire à celle de 1.*
4. *La démonstration est similaire à celle de 2 en en inversant le sens.*

Démonstration du théorème 9 *Grâce au 1 et 2 du lemme technique 10, par récurrence sur le nombre de quantificateurs.*

Lemme 38 (correction des règles) *Soit une QBF prénexe close* $Q(K \wedge F)$ *avec* K *une contrainte.*

1. *Si* $Q|_{\mathcal{SP}(K)}K$ *est une instance d'un schéma contradictoire alors* $Q(K \wedge F)$ *est non-valide.*
2. *Si* $Q|_{\mathcal{SP}(K)}K$ *est une instance d'un schéma tautologique alors* $Q(K \wedge F) \cong QF$.
3. *Si* $Q|_{\mathcal{SP}(K)}K$ *est une instance d'un schéma associé à une substitution instanciée sur les symboles propositionnels de* K *en une substitution* σ *au sein d'une règle de propagation/simplification alors* $Q(K \wedge F) \cong Q(\Sigma \wedge \sigma(F))$ *avec* Σ *la formule propositionnelle associée à la substitution.*

4. *Si $Q|_{\mathcal{SP}(K)}K$ est une instance d'un schéma associé à une substitution instanciée sur les symboles propositionnels de K en une substitution σ au sein d'une règle de propagation alors $Q(K \wedge F) \cong Q(\Sigma \wedge \sigma((K \wedge F)))$ avec Σ la formule propositionnelle associée à la substitution.*

Démonstration du lemme 38 *1. Par le lemme technique 10, si $Q(K \wedge F)$ est valide alors $Q|_{\mathcal{SP}(K)}K$ et QF le sont aussi donc si $Q|_{\mathcal{SP}(K)}K$ ou QF ne sont pas valides alors $Q(K \wedge F)$ ne l'est pas non plus or $Q|_{\mathcal{SP}(K)}K$ est une instance d'un schéma contradictoire donc $Q|_{\mathcal{SP}(K)}K$ est non-valide donc $Q(K \wedge F)$ est aussi non-valide.*

2. *$Q|_{\mathcal{SP}(K)}K$ est une instance d'un schéma tautologique donc $K \equiv \top$ donc $(K \wedge F) \equiv F$ donc $Q(K \wedge F) \cong QF$.*
3. *Σ étant la formule propositionnelle associée à la substitution σ, $(\Sigma \wedge \sigma((K \wedge F))) \equiv (K \wedge F)$ or $\sigma(K) \equiv \top$ puisque tous les symboles propositionnels sont forcés donc $Q(K \wedge F) \cong Q(\Sigma \wedge \sigma(F))$.*
4. *Σ étant la formule propositionnelle associée à la substitution σ, $(\Sigma \wedge \sigma((K \wedge F))) \equiv (K \wedge F)$ donc $Q(K \wedge F) \cong Q(\Sigma \wedge \sigma((K \wedge F)))$.*

Lemme 39 *La relation $\sim$ est une relation d'équivalence pour les contraintes quantifiées.*

Démonstration du lemme 39 *Trivialement la relation $\sim$ est une relation d'équivalence puisque τ est une fonction (pour la transitivité).*

Démonstration du théorème 10 *L'équivalence se démontre par récurrence sur le nombre d'application de l'élimination des schémas tautologiques, des règles de propagation/simplification et des règles de propagation et le lemme 38. La terminaison se démontre par induction selon l'ordre lexicographique sur l'inclusion des lieurs et le nombre de contraintes dans la matrice (si les lieurs sont identiques) : toute application des règles de propagation/simplification et des règles de propagation ôte au moins un quantificateur du lieur et la détection d'un schéma tautologique diminue le nombre de contraintes.*

Lemme 40 *Une contrainte quantifiée représentante d'un schéma est ce même schéma dont les littéraux ont été substitués par des symboles propositionnels et les complémentaires par des symboles propositionnels précédés d'une négation.*

Soit le schéma $\mathcal{QE}$, sa contrainte quantifiée représentante R et une base littérale optimale bl telle que $bl^* \cong R$.

- *Le schéma est contradictoire si et seulement si la base littérale est non_valide ;*
- *le schéma $\mathcal{QE}$ est tautologique si et seulement si la base littérale optimale bl est équivalente à la base littérale $\langle Q \mid (\top, \top)^n \rangle$ avec n le nombre de quantificateurs du lieur ;*
- *le schéma induit une règle de propagation (/simplification) si et seulement si toutes les substitutions sont telles que :*
  - *$[x \leftarrow \top]$ si et seulement si $grd(x, bl) = (\top, \bot)$;*
  - *$[x \leftarrow \bot]$ si et seulement si $grd(x, bl) = (\bot, \top)$;*
  - *$[x \leftarrow \neg y]$ si et seulement si $grd(x, bl) = (\neg y, y)$;*

– $[x \leftarrow y]$ *si et seulement si* $grd(x, bl) = (y, \neg y)$.

Démonstration du lemme 40 – *Le schéma est contradictoire si et seulement si la contrainte quantifiée représentante est non valide si et seulement si la base littérale est non_valide.*

– *Soit* $\{x_i\}_{1 \leq i \leq n}$ *les symboles propositionnels de* QR*. Un schéma* $\mathcal{QE}$ *est tautologique si et seulement si, par définition 43, la matrice de la contrainte quantifiée représentante* QR *de ce schéma est une tautologie si et seulement si* $R \equiv \top \equiv \bigwedge_{1 \leq i \leq n} ((\neg x_i \vee \top) \wedge (x_i \vee \top))$ *si et seulement si* $QR \cong Q \bigwedge_{1 \leq i \leq n} ((\neg x_i \vee \top) \wedge (x_i \vee \top))$ *si et seulement si* $bl^* \cong Q \bigwedge_{1 \leq i \leq n} ((\neg x_i \vee \top) \wedge (x_i \vee \top))$ *si et seulement si* $bl \cong \langle Q \mid (\top, \top)^n \rangle$.

– *La base littérale étant optimale et réduite* ($w \prec v$)*,*
  – $grd(v, bl) = (\top, \bot)$ *si et seulement s'il existe uniquement un modèle pour la QBF* $Q[v \leftarrow \top](R)$ *;*
  – $grd(v, bl) = (\bot, \top)$ *si et seulement s'il existe uniquement un modèle pour la QBF* $Q[v \leftarrow \bot](R)$ *;*
  – $grd(v, bl) = (\neg w, w)$ *si et seulement si* $((\neg v \vee \neg w) \wedge (v \vee w)) \equiv \top$ *si et seulement si* $(v \leftrightarrow \neg w) \equiv \top$ *si et seulement si* $[v \leftarrow \neg w]$ *est une substitution de la règle induite par le schéma ;*
  – $grd(v, bl) = (w, \neg w)$ *si et seulement si* $((\neg v \vee w) \wedge (v \vee \neg w)) \equiv \top$ *si et seulement si* $(v \leftrightarrow w) \equiv \top$ *si et seulement si* $[v \leftarrow w]$ *est une substitution de la règle induite par le schéma.*