# A Graphical Language to Query Conceptual Graphs

David Genest, Marc Legeay, Stéphane Loiseau, Christophe Béchade
*LERIA – Université d'Angers 2, Boulevard Lavoisier - 49045 Angers cedex 1 - France*
*Email: {firstname}.{name}@univ-angers.fr*

***Abstract*—This paper presents a general query language for conceptual graphs. First, we introduce kernel query graphs. A kernel query graph can be used to express an "or" between two sub-graphs, or an "option" on an optional sub-graph. Second, we propose a way to express two kinds of queries (ask and select) using kernel query graphs. Third, the answers of queries are computed by an operation based on graph homomorphism: the projection from a kernel query graph.**

***Keywords*-Conceptual graph ; Query language ; SPARQL.**

## I. Introduction

The conceptual graph model [1][2] provides a way to represent knowledge using labelled graphs. This model uses visual graphical representations in order to improve understanding of knowledge by an end-user but this model is also associated to a reasoning mecanism with several operations based on graph theory. Its querying method is based on its main operation, a graph homomorphism called *projection*: this operation determines if knowledge in a conceptual graph called *query graph* can be deduced from the knowledge base which is a conceptual graph called *factual graph*. Several extensions have been defined [3]. Inference rules [4] are used to represent implicit knowledge. These rules have a graphical representation, and rules operations are defined as graph operations. Constraints [2] are used to validate knowledge and, just like rules, they are defined as graphs. Software tools have been developed. For example, Cogui[1] is a well-known visual editor for building conceptual graph ontologies, facts, queries, rules and constraints. Cogitant[2] is a well-known reasoning engine allowing to easily build applications based on conceptual graphs since it provides a complete implementation of the model [5].

The basic querying method of the model is often too simple, that is why some extensions of the querying method have been proposed. [1] proposes to identify some nodes in a query graph to easily exploit the projection result. [6] proposes to express queries in a simple language composed of keywords to construct conceptual graph queries with the keywords representing pre-written query patterns. [7] proposes a solution to select what they call the smart answer of the projection result by using contextual knowledge from the base to distinguish between equivalent answers. These extensions are useful for certain kinds of applications, but, no general language to query conceptual graphs has been proposed. That is the reason why we propose to define a new query language that borrows ideas from Semantic Web.

Semantic web tools share with conceptual graph tools the aim to represent a knowledge base in a simple semantic network and to provide ways to exploit it. In Semantic Web tools, knowledge is represented with languages such as RDF [8] or OWL [9], and this knowledge is usually queried using *SPARQL* [10]. SPARQL provides more flexibility compared to conceptual graph to query. First, SPARQL allows to express a disjunction between several parts of a query (SPARQL uses the word "union" for that) and to identify some parts as required or as optional. Second, SPARQL provides *four kinds of queries*: ask, select, describe and construct queries. This paper proposes to extend Cogitant to improve conceptual graph query expressiveness by combining the simplicity of visual representation from conceptual graphs with the powerful querying model of Semantic Web. First, our language is based on graphs: a query has a graphical representation and answers are obtained by graph operations. Second, this query language allows disjunctions and optional parts in a conceptual graph query but also several kinds of conceptual graph queries.

The contribution of this paper is threefold. First, we introduce *kernel query graphs* that extend conceptual graphs. The notion of kernel query graph enables one to express disjunction conditions (an "or" between two of its subgraphs) and option conditions (a subgraph is preferred but not necessarily existing). Second, kernel query graphs provide a way to propose a conceptual graph query language that provides several kinds of queries. *Ask query* allows to know if the graph of the query is deducible from the factual graph. *Select query* allows to find and extract from the factual graph some nodes identified in the query graph as important. Third, the computing operation used to query and obtain *answers* is introduced: the *projection from a kernel query graph* to a factual graph. This projection is defined using the classic projection from a conceptual graph to a factual graph.

The paper is organized as follows. Section II provides background definitions on the conceptual graph model and the projection. Section III presents the kernel query graph model. Section IV presents the projection from a kernel query graph to a factual graph. Section V presents the ask query. Section VI presents the select query.

[1] http://www.lirmm.fr/cogui
[2] http://cogitant.sourceforge.net

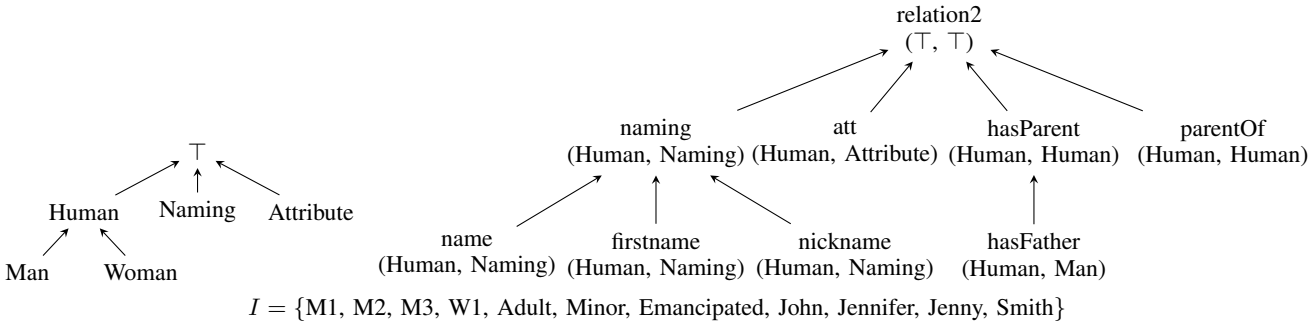



Figure 1. Vocabulary $(T_C, T_R, I, \sigma)$ with $T_C$ represented by the tree on the left, $T_R$ represented by the tree on the right, and the signature $\sigma$ of each relation is precised under each relation type.

## II. Conceptual graph model

The conceptual graph formalism used in this paper is based on the formalism defined by [2]. The *vocabulary* is a simple ontology representing some domain knowledge. It defines an ordered set of concept types, and ordered sets of relation types - one for each arity of relations. Every set is ordered by a specialization relation. Two types of entities are introduced: a specific entity is referred by an individual marker and an unspecific entity is referred by a generic marker. Relation type has a signature which specifies the arity and the most general concept type for each argument.

Definition 1 (Vocabulary). *A* vocabulary *is a 4-tuple $(T_C, T_R, I, \sigma)$, where:*

- *$T_C$, the set of concept types, is partially ordered by a relation $\leq$ and has a greatest element $\top$.*
- *$T_R$, the set of relation types, is partially ordered by a relation $\leq$ and is partitioned into subsets $T_R^1, \ldots, T_R^k$ of relation types of arity $1, \ldots, k$, respectively.*
- *$I$ is the set of individual markers. $*$ is called generic marker, $I \cup \{*\}$ represents the set of markers and is ordered as follows: $*$ is greater than any element of $I$ and elements of $I$ are pairwise incomparable.*
- *$\sigma$ is a mapping, which to every relation type $r \in T_R^j, 1 \leq j \leq k$ assigns a signature $\sigma(r) \in (T_C)^j$.*

Fig. 1 is the vocabulary used in the following examples.

A *conceptual graph* is a bipartite multigraph defined over a vocabulary. One set of nodes is called the set of concept nodes, and the other is called the set of relation nodes, representing links between concepts. A relation node is labelled by a relation type and a concept node is labelled by a pair formed by a concept type and a marker.

Definition 2 (Conceptual graph). *Let $\mathcal{V} = (T_C, T_R, I, \sigma)$ be a vocabulary. A* conceptual graph *defined over $\mathcal{V}$ is a 4-tuple $G = (C, R, E, l)$ satisfying the following conditions:*

- *$(C, R, E)$ is an undirected and bipartite multigraph. $C$ is the set of concept nodes, $R$ is the set of relation nodes, $E$ is the set of edges.*
- *$l$ is a labelling function of the nodes and edges of $(C, R, E)$ satisfying: A concept node $c$ is labelled by a pair $l(c) = (type(c), marker(c))$, in which $type(c) \in T_C$ and $marker(c) \in I \cup \{*\}$ ; Two different concept nodes cannot be labelled by the same individual marker ; A relation node $r$ is labelled by $l(r) \in T_R$ and the degree of $r$ is equal to the arity of $l(r)$ ; Edges incident to a relation node $r$ are totally ordered and they are labelled from $1$ to the arity of $l(r)$ ; If there is an edge between a concept node $c$ and a relation node $r$ labelled by $i$, then $type(c)$ must respect the type of the $i$-th argument of $\sigma(l(r))$.*

Two conceptual graphs, defined over the vocabulary of Fig. 1, are represented in Fig. 2. In order to simplify the reading of graphs presented here, we use a visual simplification: edges labelled 1 (resp. 2) are represented by an arrow from the concept (resp. relation) to the relation (resp. concept). The generic marker is not displayed. $G_f$ represents the fact that the man M1 has the man M2 as father, and M2 has a man as father (who is unknown).

The projection is the query operation used in the conceptual graph model. Giving a conceptual graph $G_q$ (query graph), and another conceptual graph $G_f$ (factual graph), defined over the same vocabulary, there is a projection from $G_q$ to $G_f$ if the information represented by $G_q$ can be deduced by the one represented by $G_f$.

Definition 3 (Projection and image). *Let $G_q = (C_q, R_q, E_q, l_q)$ and $G_f = (C_f, R_f, E_f, l_f)$ be two conceptual graphs defined over the same vocabulary.*
*A* projection *$\pi$ from $G_q$ to $G_f$ is a mapping from $C_q \cup R_q$ to $C_f \cup R_f$ such that: $\forall c \in C_q, \forall r \in R_q$, if $rc \in E_q$ then $\pi(r)\pi(c) \in E_f$ and $l_q(rc) = l_f(\pi(r)\pi(c))$ ; $\forall n \in C_q \cup R_q, l_f(\pi(n)) \leq l_q(n)$. Note that the label of concept nodes is ordered as follow : $(t, m) \leq (t', m') \leftrightarrow t \leq t' \wedge m \leq m'$. Nodes of $G_f$ corresponding to nodes of $G_q$ are called the* images *of $G_q$ by $\pi$.*

In Fig. 2, there is a projection from the query graph $G_q$ to the factual graph $G_f$ (represented with bold arrows).

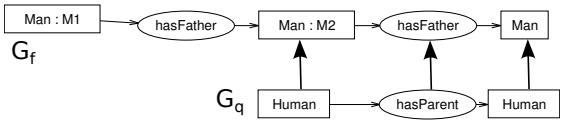



Figure 2. Projection from $G_q$ to $G_f$.

## III. Kernel query graph

In the conceptual graph model, the query graph is a conceptual graph, this paper proposes to construct more complex queries by using a richer query than a conceptual graph, these queries are based on kernel query graphs. A kernel query graph enables one to precise disjunction conditions or option conditions in a conceptual graph. It is built with a *hierarchy* of blocks which structure a conceptual graph: a *block* contains a part of the conceptual graph and is used to describe a disjunction, an option or simply a standard part of the conceptual graph. A block is defined as a *set of nodes* of a conceptual graph associated with a *block type*. Every concept node in the set must have its linked relation nodes in the set: thus a concept node of the block is characterized by all the relations it is linked to. The different block types proposed precise the condition of the block: `Disjunction`, `Option` or `Standard`. A disjunction block is composed of child blocks and expresses a disjunction between its child blocks, i.e., it is used to search for at least one of the child blocks in the factual graph. An option block expresses that the part of the graph it contains is facultative, i.e., it is used to search for a graph with the option part in the factual graph, but, if it is not possible, a graph without the option part is suitable.

Definition 4 (Block of a graph). *Let $G = (C, R, E, l)$ be a conceptual graph. A* block of $G$ *is a pair $b = (N, T)$ where: $N \subseteq C \cup R$ is the* set of nodes *of the block such that: $\forall c \in N \cap C, \forall r \in R$, if $rc \in E$ then $r \in N$ ; $T \in \{\texttt{Standard}, \texttt{Option}, \texttt{Disjunction}\}$ is the* block type.

The conceptual graph in Fig. 3 has five blocks. Blocks are represented by labelled frames, nodes of the block belong to the frame and a label specifies the type of each block.

A tree is used to hierarchize the blocks of a conceptual graph. Blocks are the nodes of the tree. *Root block*, *parent block*, *ancestor blocks* and *child blocks* are defined thanks to the vocabulary of trees. The hierarchy imposes that the parent-child relation between blocks verifies the set of nodes of the child block is included in the set of nodes of the parent block, and if one block has a node in common with another one, then one is the ancestor of the other. The root block of the hierarchy is a standard block which contains all the nodes of the graph. A disjunction block is used to express an "or" between its children, it must have at least 2 child blocks and all the nodes of its set of nodes are in the sets of its child blocks.

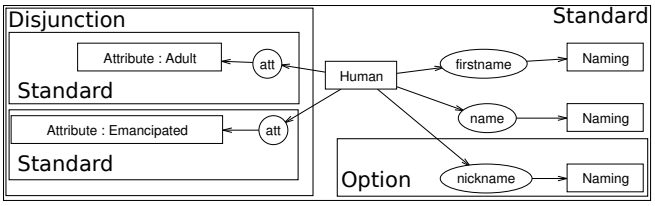



Figure 3. A kernel query graph $G_k$.

Definition 5 (Hierarchy). *Let $G = (C, R, E, l)$ be a conceptual graph. A* hierarchy *$\mathcal{H}$ of $G$ is a tree $(B, A, r)$ where the root $r$ is the block $(C \cup R, \texttt{Standard})$, the set of nodes of the tree $B$ is the set of blocks of $G$, and $A$ is the set of arcs. The hierarchy is such that:*

- *Let $b = (N_b, T_b) \in B \setminus \{r\}$, if $parent(b) = (N_p, T_p)$ then $N_b \subseteq N_p$.*
- *Let $b = (N_b, T_b) \in B, b' = (N_{b'}, T_{b'}) \in B$ if $N_b \cap N_{b'} \neq \emptyset$ then $N_b \subseteq N_{b'}$ or $N_{b'} \subseteq N_b$.*
- *Let $b = (N_b, \texttt{Disjunction})$ then $|children(b)| \geq 2$ and $N_b = \bigcup_{(N_c, T_c) \in children(b)} N_c$.*

The hierarchy of the conceptual graph in Fig. 3 appears on the graph by the inclusion of blocks.

Definition 6 (Kernel query graph). *A* kernel query graph *$G_k = (G, \mathcal{H})$ is a pair where $G$ is a conceptual graph and $\mathcal{H}$ is a hierarchy of $G$.*

Fig. 3 represents a kernel query graph $G_k = (G, \mathcal{H})$. It will express a search for an adult or emancipated human who have a first name, a last name and, if it exists, a nickname.

## IV. Projection from a kernel query graph

A kernel query graph can be developed into a set of conceptual graphs. This set of conceptual graphs, called the *set of developed graphs* of the kernel query graph, defines the semantics of a kernel query graph. This set represents the set of conceptual graphs which we are looking for. This set enables one to define a projection from a kernel query graph to a factual conceptual graph as a projection from a developed graph to the factual graph. The idea to obtain a developed graph from a kernel query graph is to a create kernel query graph with only standard blocks. It is made by a sequence of *developments* that change block types only in `Standard`, remove option blocks, and remove some children of disjunction blocks. Note that when a block is removed from a kernel query graph, not only the nodes of the block and connected edges are removed from the conceptual graph, but the block and all its descendants are also removed.

Definition 7 (Development operation). *Let $G_k = (G, \mathcal{H})$ be a kernel query graph with $\mathcal{H} = (B, A, r)$.*

- *Let $b \in B$ be an option block.*
  *The graph $G_k$, in which the type of $b$ is changed in*

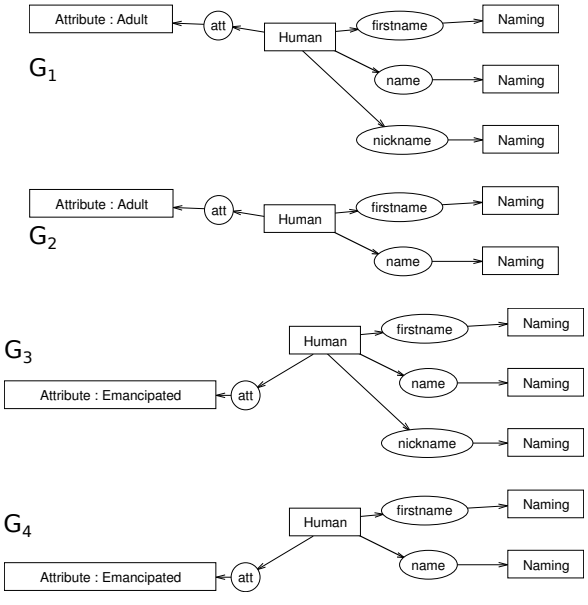



Figure 4. Developed graphs from $G_k$.

`Standard`*, is a* development *of* $G_k$*. The graph* $G_k$*, in which* $b$ *is removed, is a* development *of* $G_k$*.*

- *Let* $b \in B$ *be a disjunction block.*
  *The graph* $G_k$*, in which the type of* $b$ *is changed in* `Standard` *and in which every child of* $b$ *except one is removed, is a* development *of* $G_k$*.*

DEFINITION 8 (Set of developed graphs). *Let* $G_k = (G, \mathcal{H})$ *be a kernel query graph. The* set of developed graphs $DG$ *of* $G_k$ *is the set of all the conceptual graphs associated with the kernel query graphs containing only standard blocks obtained by a sequence of developments from* $G_k$*.*

The conceptual graphs in Fig. 4 form the set of developed graphs $DG = \{G_1, G_2, G_3, G_4\}$ of the kernel query graph $G_k$ from Fig. 3.

In order to define the projection from a kernel query graph to a factual graph, we first search for the projections from each developed graphs (which are conceptual graphs) to the factual graph. The projection from a developed graph is generally a projection from the kernel query graph. The exception is due to the option: the optional part of a graph is required if this part belongs to the factual graph. In this case, the projection which does not contain the option block is not a projection from the kernel query graph. Note that this projection is a restriction[3]of the projection containing the optional information. The set of projections from a kernel query graph to a factual graph is defined as the set of projections from developed graphs of the kernel query graph, except for the projections that are restrictions of others projections.

[3]Reminder: a function $f : D \to I$ is a restriction of $f' : D' \to I$, denoted $f \subset f'$, if $D \subset D'$ and $\forall d \in D, f(d) = f'(d)$.

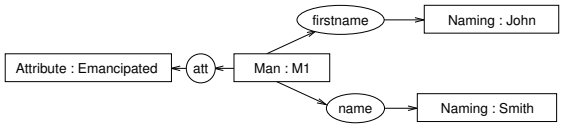



Figure 6. Image of an answer to an ask query.

DEFINITION 9 (Projection from a kernel query graph). *Let* $G_k = (G, \mathcal{H})$ *be a kernel query graph,* $G_f$ *be a conceptual graph defined over the same vocabulary than* $G$*, and* $\Pi_{DG}$ *be the set of projections from the developed graphs* $DG$ *of* $G_k$ *to* $G_f$*. The set of the* projections from $G_k$ to $G_f$ *is:* $\Pi = \{\pi \in \Pi_{DG} | \nexists \pi' \in \Pi_{DG}, \pi \subset \pi'\}$.

To compute the set of projections from a kernel query graph to a conceptual graph, a naive algorithm computes the set of developed graphs, their projections to the factual graph and remove those that are restrictions. For performance reasons, the algorithm we implemented computes the projection block by block following the hierarchy from the root, testing the alternatives of disjunction and testing if the option part of option blocks is in the factual graph.

There is only one projection $\pi$ from the kernel query graph $G_k$ in Fig. 3 to the conceptual graph $G_f$ in Fig. 5. The image of the projection is pictured in Fig. 6. It is a sub-graph of $G_f$ and it corresponds to a projection from the developed graph $G_4$ in Fig. 4 to $G_f$.

## V. ASK QUERY

An ask query allows to know if the kernel query graph of the query is deducible from the factual conceptual graph. This type of query corresponds to the classic querying of the conceptual graph model with disjunction and option conditions added. An ask query is only a kernel query graph. An answer to an ask query is a projection from the kernel query graph to the factual graph.

DEFINITION 10 (Ask query and its answer). *Let* $G_f$ *be a conceptual graph defined over the vocabulary* $\mathcal{V}$*. An* ask query *is a kernel query graph* $G_k = (G, \mathcal{H})$*, with* $G$ *defined over* $\mathcal{V}$*. An* answer to an ask query $G_k$ *is a projection* $\pi$ *from* $G_k$ *to* $G_f$*.*

The kernel query graph $G_k$ in Fig. 3 can be used as an ask query. As explained earlier, the answer to the query can been seen in Fig. 6.

## VI. SELECT QUERY

In an ask query, the image of some nodes can not be more important than others, but, we would like to express that some nodes are more important than others. This paper proposes the select queries which allow to find and extract from the factual conceptual graph some nodes identified as important in the query.

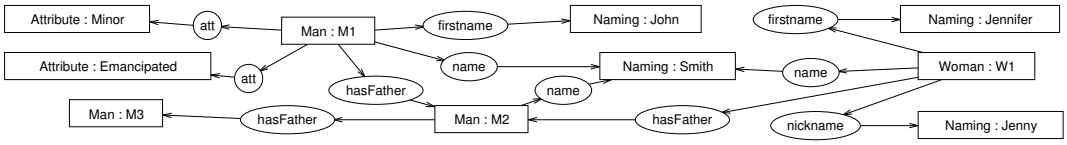
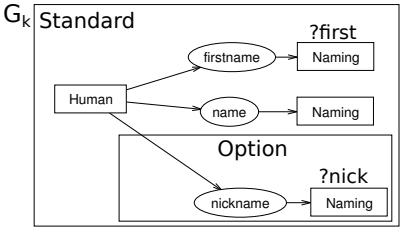



Figure 5. Conceptual graph $G_f$.



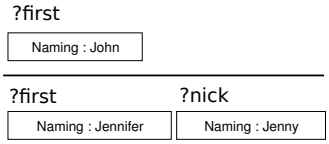




Figure 7. Select query and its two answers.

DEFINITION 11 (Select query). *Let $S_N$ be a set of names. A* select query *is a pair $Q_s = [G_k, sel]$, where $G_k = (G, \mathcal{H})$ is a kernel query graph with $G = (C, R, E, l)$, and $sel$ is a naming function defined over the set of* selector names *$S_N$ as follows: $sel : S_N \to C \cup R$.*

An answer to a select query is a function which associates each name of the set of selector names to a node of the factual graph. For a projection from the kernel query graph to the factual graph, each name is associated to the image (if it exists) in the factual graph of the node of kernel query graph that is linked to the name. A node may have no image by the projection due to disjunction or option condition.

DEFINITION 12 (Answer to a select query). *Let $G_f = (C_f, R_f, E_f, l_f)$ be a conceptual graph, $Q_s = [G_k, sel]$ be a select query with $sel$ defined over the names $S_N$, and $\pi$ be a projection from $G_k$ to $G_f$. An* answer to the select query *$Q_s$ is a partial function from $S_N$ in $C_f \cup R_f$ which associates to $s \in S_N$ the node $\pi(sel(s))$ of $G_f$ if it exists.*

Fig. 7 presents the select query $Q_s = [G_k, sel]$, where the naming function $sel$, defined over $S_N = \{\texttt{?first}, \texttt{?nick}\}$, is directly represented on $G_k$ and its two answers on $G_f$ (Fig. 5). The query allows to know the first name, and if it exists the nickname, of every human who have a first name and a last name in the factual graph.

## VII. CONCLUSION

This paper introduced the kernel query graphs that are used to query a factual conceptual graph, representing a knowledge base, by means of the projection from kernel query graph to conceptual graph. Kernel query graphs are used in two kinds of queries, and we have defined two other kinds of queries. They allow to obtain describing information from the factual conceptual graph linked to a particular node of the query, and also to deduce new facts using knowledge extracted from the factual conceptual graph [11]. These four kinds of queries provide a general language to query conceptual graphs. The graphical and visual aspect of the querying model is an important choice, which has been made to be close to conceptual graphs. This query language has been implemented as an extension of the Cogitant library.

## REFERENCES

[1] J. F. Sowa, *Conceptual Structures: Information Processing in Mind and Machine*. Addison-Wesley, 1984.

[2] M. Chein and M.-L. Mugnier, *Graph-based knowledge representation: computational foundations of conceptual graphs*. Springer, 2009.

[3] J.-F. Baget and M.-L. Mugnier, "The SG family: Extensions of simple conceptual graphs," in *Proceedings of IJCAI 2001*, B. Nebel, Ed. Morgan Kaufmann, 2001, pp. 205–212.

[4] E. Salvat, "Theorem proving using graph operations in the conceptual graph formalism," in *Proc. of the 13th European Conf. on Artificial Intelligence (ECAI'98)*, 1998.

[5] D. Genest, "Cogitant 5.3.0 - reference manual," http://cogitant.sourceforge.net/files/cogitant.pdf, Mar. 2014.

[6] C. Pradel, O. Haemmerlé, and N. Hernandez, "Expressing conceptual graph queries from patterns: how to take into account the relations," in *Conceptual Structures for Discovering Knowledge*. Springer, 2011, pp. 229–242.

[7] N. Moreau, M. Leclère, and M. Croitoru, "Distinguishing answers in conceptual graph knowledge bases," in *Conceptual Structures: Leveraging Semantic Technologies*. Springer, 2009, pp. 233–246.

[8] F. Manola, E. Miller, and B. McBride, "RDF primer," *W3C recommendation*, 2004.

[9] D. L. McGuinness, F. Van Harmelen *et al.*, "OWL web ontology language overview," *W3C recommendation*, 2004.

[10] E. Prud'Hommeaux, A. Seaborne *et al.*, "SPARQL query language for RDF," *W3C recommendation*, 2008.

[11] M. Legeay, "Interrogation de connaissances exprimées en graphes conceptuels," Master's thesis, Université d'Angers, France, 2013.