Representing Multi-Scale Datalog +/- Using Hierarchical Graphs

Cornelius Croitoru, Madalina Croitoru

Representing Multi-Scale Datalog +/- Using Hierarchical Graphs

Cornelius Croitoru and Madalina Croitoru

Faculty of Computer Science, Al. I. Cuza University, Romania
GraphIK, University of Montpellier, France

Abstract We introduce a multi scale knowledge representation and reasoning formalism of Datalog+/- knowledge bases. This is defined on a novel graph transformation system that highlights a new type of rendering based on the additional expansion of relation nodes. Querying and integration capabilities of our approach are based on a FOL sound and complete homomorphism.

1 Introduction

The set of requirements for knowledge representation formalisms must include *(i)* the existence of a declarative semantics, *(ii)* a logical foundation, and *(iii)* the possibility of representing structured knowledge [2]. While many languages have followed these three directions, a lot of existing work focused mainly on the first two aspects. Here we address the third requirement and namely the need to represent hierarchical, multi scale knowledge. Therefore, our representation structures fulfil all these three conditions in a formal way. By hierarchical, multi scale, knowledge we understand knowledge that can be represented at different level of granularity. For instance, we can see the human body as made out of body parts such as hands, legs, lungs etc. or we can zoom in and look at the muscles and the bones or we can further zoom in and see how minerals and organic substances interact in our body. Such levels are not disconnected - a lack of Mg in the body can lead to muscle spasms that can lead to tingling in the legs. Multi scale knowledge bases are commonly used in Life Sciences [5] but not only. Supply Chain Management [16], Information Integration [19], Sensor Networks [1], Policy Rules [13] etc. all require to represent and reason about knowledge at various levels of granularities while being able to go from one level to the other easily.

The representation we propose is using the notion of a transitional description. The transitional description allows to go from one level of granularity to the next. This mechanism is used to define inductively hierarchical structures of depth d. We build upon the state of the art and, in this paper, consider the Datalog +/- language. By considering n-ary predicates and existential rules (i.e. rules that allow for existentially quantified variables in the conclusion) this language generalises certain Description Logics. Even if not endowed with a graphical depiction, Datalog +/- is logically equivalent to Conceptual Graphs with Rules and Negative Constraints. In this paper, while considering the core logical language of Datalog +/- we endow it with graph based logically sound and complete semantics.

The paper is structured as follows. In Section 2 we explain the choice of logical language and place ourselves within the state of the art for representing hierarchical knowledge. In Section 3 we recall basic notions needed throughout the paper such as facts, rules, knowledge base, etc. We also show how to endow the Datalog +/- language with a graph based syntax while staying sound and complete wrt semantics. Section 4 presents the hierarchical knowledge representation and reasoning formalism and Section 5 concludes the paper.

2 State of the Art

In this paper we consider a rule based language that gains more and more interest from a practical point of view, Datalog +/- [7]. We consider existential variables in the head of the rules as well as n-ary predicates and conflicts (and generalise certain subsets of Description Logics (e.g. DL-Lite) [3,8]). The tractability conditions of the considered rule based language rely on different saturation (chase) methods [17]. The language can be equivalently seen in a logically sound and complete graph based representation [14] [4].

The data structure discussed here evolved from Conceptual Graphs [18], Nested Conceptual Graphs [11] and respectively Layered Conceptual Graphs [12]. The idea of a detailed context of knowledge can be traced back to the definition of Simple Conceptual Graphs (SCGs) [18], to the work of [15] and to the definition of the more elaborate Nested Conceptual Graphs [11]. The querying capabilities associated with our approach are supported by the logically sound homomorphism operation, which is defined between a query and the hierarchical structure.

Except [12], existing work in representing hierarchical knowledge in diagrammatic way does not relate to the context in which the complex nodes appear. The complex nodes behave like *glass boxes*, corresponding to a "zoom" action. In [12] the authors use a similar notion of multi level granularity knowledge but their approach is closely following Conceptual Graphs. In this paper the language used is more generic following [7]. Last, a hierarchical extension of Datalog has also been proposed by [6] but the approach suffers from the lack of graph based rendering of the transitions between levels.

3 Basic Notions

We consider constants but no other functional symbols; a vocabulary W is composed of a set of predicates P and a set of constants C. Constants identify the individuals in the knowledge base and predicates represent n-ary relations between such individuals. We also consider X, a set of variables in the knowledge base.

Definition 1 (Vocabulary) *Let C be a set of constants and P a set of predicates. A vocabulary is a pair $W = (P,C)$ and arity is a function from P to $\mathbb{N}$. For all $p \in P$, $arity(p) = i$ means that the predicate p has arity i.*

We will consider an infinite set X of variables, disjoint from P and C. A term is an element of $C \cup X$. An atom is of form $p(t_1,...,t_k)$, where p is a predicate of arity k in W and the t_i are terms. For a given atom A, we note $terms(A)$, $csts(A)$ and $vars(A)$ the terms, constants and variables occurring in A.

Definition 2 (Fact) *A fact is a finite, but possibly empty, set of atoms on a vocabulary. For a given fact F, we note atoms(F) the atoms occurring in F.*

Example. Let us consider a vocabulary $W = (P,C)$. $P = \{$man,woman$\}$, $C = \{Bob, Alice\}$ and $arity = \{$(man,1),(woman,1)$\}$. $man(Bob)$ and $woman(Alice)$ are two distinct atoms on W, and $F = \{man(Bob), woman(Alice)\}$ a fact.

We can represent facts as labelled ordered bipartite graphs where one class of partition represents the concepts (i.e. the unary predicates) and the other the relations. Such representation is well known in the literature (see [18] or [9]).

A *bipartite graph* is a graph $G = (V_G, E_G)$ with the nodes set $V_G = V_C \cup V_R$, where V_C and V_R are finite disjoint nonempty sets, and each edge $e \in E_G$ is a two element set $e = \{v_C, v_R\}$, where $v_C \in V_C$ and $v_R \in V_R$. Usually, a bipartite graph G is denoted as $G = (V_C, V_R; E_G)$. We call $G^{\emptyset}$ the empty bipartite graph without nodes and edges.

Let $G = (V_C, V_R; E_G)$ be a bipartite graph. The number of edges incident to a node $v \in V(G)$ is the degree, $d_G(v)$, of the node v. If, for each $v_R \in V_R$ there is a linear order $e_1 = \{v_R, v_1\}, \dots, e_k = \{v_R, v_k\}$ on the set of edges incident to v_R (where $k = d_g(v)$), then G is called an *ordered bipartite graph*. A simple way to express that G is ordered is to provide a labelling $l : E_G \to \{1, \dots, |V_C|\}$ with $l(\{v_R, w\})$ = index of the edge $\{v_R, w\}$ in the above ordering of the edges incident in G to v_R. l is called a *order labelling* of the edges of G. We denote an ordered bipartite graph by $G = (V_C, V_R; E_G, l)$.

For a vertex $v \in V_C \cup V_R$, the symbol $N_G(v)$ denotes its neighbours set, i.e. $N_G(v) = \{w \in V_C \cup V_R | \{v,w\} \in E_G\}$. Similarly, if $A \subseteq V_R \cup V_C$, the set of its neighbours is $N_G(A) = \cup_{v \in A} N_G(v) - A$. If G is an ordered bipartite graph, then for each $r \in V_R$, the symbol $N_G^i(r)$ denotes the i-th neighbour of r, i.e. $v = N_G^i(r)$ if and only if $\{r, v\} \in E_G$ and $l(\{r, v\}) = i$.

Throughout this paper we use a particular type of subgraph of a bipartite graph: $G^1 = (V_C^1, V_R^1; E_G^1)$ is a subgraph of $G = (V_C, V_R; E_G)$ if $V_C^1 \subseteq V_C$, $V_R^1 \subseteq V_R$, $N_G(V_R^1) \subseteq V_C^1$ and $E_G^1 = \{ \{v, w\} \in E_G | v \in V_C^1, w \in V_R^1\}$. In other words, we require that the (ordered) set of all edges incident in G to a vertex from V_R^1 must appear in G^1. Therefore, a subgraph is completely specified by its vertex set.

In particular, if $A \subseteq V_C$:

- The *subgraph spanned by* A *in* G, denoted as $\lceil A \rceil^G$, has $V_C(\lceil A \rceil^G) = A \cup N_G(N_G(A))$ and $V_R(\lceil A \rceil^G) = N_G(A)$.
- The *subgraph generated by* A *in* G, denoted as $\lfloor A \rfloor_G$, has $V_C(\lfloor A \rfloor_G) = A$ and $V_R(\lfloor A \rfloor_G = \{v \in N_G(A) | N_G(v) \subseteq A\}$.
- For $A \subseteq V_R$, *the subgraph induced by* A *in* G, denoted $[A]_G$, has $V_C([A]_G) = N_G(A)$ and $V_R([A]_G) = A$.

Example. Let us consider $F = \{man(Bob), woman(Alice), loves(Bob, Alice)\}$ a fact. The bipartite graph representation $G = (V_C, V_R; E_G)$ consists of $V_C = \{man(Bob), woman(Alice)\}$, $V_R = \{loves(Bob, Alice)\}$ and E_G the corresponding edges linking *Bob* to *Alice* via *loves*.

3.1 Semantics

Definition 3 (Interpretation) *Let* $W = (P, C)$ *be a vocabulary. An interpretation of* W *is a pair* $I = (\Delta, .^I)$ *where* Δ *is the domain of the interpretation, and* $.^I$ *a function where:* $\forall$ *c in* C, $c^I \in \Delta$ *and* $\forall$ *p in* P, $p^I \subseteq \Delta^{arity(p)}$.

An interpretation is non empty and can be possibly infinite.

Definition 4 (Model) *Let* F *be a fact on* W, *and* $I = (\Delta, .^I)$ *be an interpretation of* W. *We say that* I *is a model of* F *iff there exists an application* v : *terms(F)* $\rightarrow \Delta$ *(called a justification of* F *in* I*) such that:*

- $\forall$ $c \in csts(F)$, $v(c) = c^I$ *and*
- $\forall$ $p(t_1,...,t_k) \in atoms(F)$, $(v(t_1),...,v(t_k)) \in p^I$.

Definition 5 (Fact to logical formula) *Let* F *be a fact.* $\phi(F)$ *is the logical formula that corresponds to the conjunction of atoms in* F. *And* $\Phi(F)$ *corresponds to the existential closure of* $\phi(F)$.

Example Let us consider a fact $F = \{person(x), name(x, Bob), age(x, 25)\}$.

- $\phi(F) = person(x) \land name(x, Bob) \land age(x, 25)$.
- $\Phi(F) = \exists x\ person(x) \land name(x, Bob) \land age(x, 25)$.

Property 1 (Model equivalence). Let F be a fact and I be an interpretation of W. Then I is a model of F iff I is a model (in the FOL sense) of $\Phi(F)$.

Definition 6 (Entailment) *Let* F *and* G *be two facts,* F *entails* G *if every model of* F *is also a model of* G. *The entailment relation is then noted* $F \models G$.

Definition 7 (Homomorphism) *Let F and F' be facts. Let σ: $terms(F) \rightarrow terms(F')$ be a substitution, i.e. a mapping that preserves constants (if $c \in C$, then $\sigma(c) = c$). We then note $\sigma(F)$ the fact obtained from F by substituting each term t of F by $\sigma(t)$. Then σ is a homomorphism from F to F' iff the set of atoms in $\sigma(F) \subseteq F'$.*

Example Let $F = \{man(x_1)\}$ and $F' = \{man(Bob), woman(Alice)\}$. Let $\sigma : terms(F) \rightarrow terms(F')$ be a substitution such that $\sigma(x_1) = Bob$. Then σ is a homomorphism from F to F' since the atoms in $\sigma(F)$ are $\{man(Bob)\}$ and the atoms in F' are $\{man(bob), woman(Alice)\}$.

Property 2 (Entailment). Let F and Q be facts. $F \models Q$ iff there exists Π an homomorphism from Q to F.

In [18] homomorphism is denoted as projection and it is the fundamental operation on simple conceptual graphs. If we consider the bipartite depiction of facts mentioned above, a *projection from G to H* is a mapping

$\Pi : V_C(G) \cup V_R(G) \rightarrow V_C(H) \cup V_R(H)$ such that:

- $\Pi(V_C(G)) \subseteq V_C(H)$ and $\Pi(V_R(G)) \subseteq V_R(H)$;
- $\forall c \in V_C(G)$, $\forall r \in V_R(G)$ if $c = N^i_G(r)$ then $\Pi(c) = N^i_H(\Pi(r))$;
- $\forall v \in V_C(G) \cup V_R(G)$ $\lambda_G(v) \geq \lambda_H(\Pi(v))$ where λ is a labelling of nodes with elements from a set of finite partially ordered set (the terminology, the support - please see next section.).

A projection $G \rightarrow H$ exists if and only if G *entails* H (i.e. $G \geq H$). Subsumption checking is an NP-complete problem [9].

3.2 Rules

Rules are objects used to express that some new information can be inferred from another information. Rules are also used in order to define the terminological knowledge corresponding to a set of facts. In the rules we could include the hierarchy of concept types, the hierarchy of relations and more complicated rules that do not simply define generalisations specialisations of types.

Rules are built from two different parts called head and body. Once the body of a rule can be deduced from a fact, then the information in the head should also be considered when accessing information. Please note that the head could contain new variables not present in the body. In this case applying such rules should be done with care as it can generate an infinite number of new facts to be taken into account.

Definition 8 (Rule) *Let H and B be facts. A rule is a pair $R = (H, B)$ of facts where H is called the head of the rule and B is called the body of the rule. A rule is commonly noted $B \rightarrow H$.*

Definition 9 (Rule model) *Let W be a vocabulary, I an interpretation on W, and R a rule on W. We say that I is a model of R iff for every justification V_B of B in I there exists a justification V_H of H in I such that $\forall t \in vars(B) \cap vars(H)$, $V_B(t) = V_H(t)$.*

Definition 10 (Rule to logical formula) *Let $R = (H, B)$ be a rule. Let b_x be the variables from B, and h_x be the variables from H that are not in B, the logical formula corresponding to R is the following: $\Phi(R) = \forall\ b_x\ (\ \phi(B) \rightarrow \exists\ h_x\ \phi(H))$.*

Example Let us consider a rule $R = \{person(x),\ person(y),\ sibling(x, y)\} \rightarrow \{person(z),\ parent(x, z),\ parent(y, z)\}$. $\Phi(R) = \forall x, y(\ person(x) \land person(y) \land sibling(x, y) \rightarrow \exists z\ person(z) \land parent(x, z) \land parent(y, z))$.

Property 3 (Model equivalence). Let R be a rule and I be an interpretation of W. Then I is a model of R iff I is a model (in the FOL sense) of $\Phi(R)$.

In conceptual graphs, the rules that define the terminology (hierarchy of concepts and relations) is defined in the so called support of the conceptual graph. The support is taken into account when performing projection (as explained in the previous section). In this paper we will also use the hierarchy of rules and concepts in the multi scale representation of knowledge. Therefore, we remind here the notion of support that will be used later on in the hierarchical knowledge base definitions.

The support is a tuple $S = (T_C, T_R)$ where T_C is a finite partially ordered set (poset), $(T_C, \leq)$, of *concept types*, defining a type hierarchy which has a greatest element $\top_C$, namely the universal type. In this specialisation hierarchy, $\forall x, y \in T_C$, $x \leq y$ is used to denote that x is a subtype of y. T_R is a finite set of *relation types* partitioned into k posets $(T_R^i, \leq)_{i=1,k}$ of relation types of arity i ($1 \leq i \leq k$), where k is the maximum arity of a relation type in T_R. Moreover, each relation type of arity i, $r \in T_R^i$, has an associated *signature* $\sigma(r) \in \underbrace{T_C \times \ldots \times T_C}_{i \text{ times}}$, which specifies the maximum concept type of each of its arguments. This means that if we use $r(x_1, \ldots, x_i)$, then x_j is a concept with $type(x_j) \leq \sigma(r)_j$ ($1 \leq j \leq i$). The partial orders on relation types of the same arity must be *signature-compatible*, i.e. it must be such that $\forall r_1, r_2 \in T_R^i\ r_1 \leq r_2 \Rightarrow \sigma(r_1) \leq \sigma(r_2)$. The sets T_C, T_R are mutually disjoint.

Before defining formally what a knowledge base is, let is make a note about how the support is related to facts. If we consider a fact $G = (V_C, V_R; E_G)$ then the nodes in the graph (fact), the concepts and the relation will respect signature wise the support (hierarchy given by the concept types and relation types). Formally, we can consider λ is a labelling of the nodes of G with elements from the support $S = (T_C, T_R)$: $\forall r \in V_R,\ \lambda(r) \in T_R^{d_G(r)}$; $\quad \forall c \in V_C,\ \lambda(c) \in T_C$ such that if $c = N_G^i(r)$, $\lambda(r) = t_r$ and $\lambda(c) = t_c$, then $t_c \leq \sigma_i(r)$.

3.3 Knowledge Base

Definition 11 (Knowledge base) *Let W be a vocabulary. A knowledge base (KB) is a pair K = $(F,\mathcal{R})$ where F is a fact on W and $\mathcal{R}$ is a set of rules on W.*

Definition 12 (KB model) *Let $K = (F,\mathcal{R})$ be a knowledge base and I be an interpretation. I is a model of K iff I is a model of F and also a model of every rule R_i in R.*

Definition 13 (Entailment) *Let K be a knowledge base and Q be a fact. K entails Q iff all models of K are also models of Q.*

Definition 14 (Logical representation) *Let $K = (F, \mathcal{R})$ be a knowledge base. $\Phi(K) = (\Phi(F),\Phi((R)))$ is the logical representation of K. $\Phi(F)$ is the logical formula of F and $\Phi(\mathcal{R}) = \bigcup_{r\in\mathcal{R}} \Phi(r)$.*

Property 4 (Model equivalence). Let K be a knowledge base and I be an interpretation. Then I is a model of K iff I is a model (in the FOL sense) of $\Phi(K)$.

In other words, given a knowledge base K and a conjunctive query Q, the RBDA problem consists in answering if Q can be deduced from K, denoted $K \models Q$.

Rule application can be performed of two different methods, called FORWARD CHAINING and BACKWARDS CHAINING.

Definition 15 (Applicable rule) *Let $R = (H, B)$ be a rule and F be a fact. R is applicable to F if there exists an homomorphism $\Pi : B \to F$. In this case, the application of R to F according to Π is a fact $\alpha(F, R, \Pi) = F \cup \Pi^{safe}(H)$.*

Please note the use of Π^{safe} instead of Π. Π^{safe} is an application that converts existential variables into fresh ones at the moment of joining new information with the initial fact. Such process is important in order to avoid unnecessary specializations. A derivation is the result of a finite sequence of rules application.

Definition 16 (Derivation) *Let F be a fact. F' is a derivation of F iff there exists a finite sequence of facts $F = F_0$, ..., $F_k = F'$ (called the derivation sequence) such that for every i there exists R and Π such that $F_i = \alpha(F_{i-1}, R, \Pi)$.*

Definition 17 (Saturation) *Let F be a fact and R be a set of rules. $\Pi_R(F)$ = $\{\Pi$:$B_R \to F\}$ is the set of homomorphisms of the body of applicable rules to F. $\alpha(F, R) = F \bigcup_{\pi\in\Pi_R(F)} \pi^{safe}(H_R)$ is the result of the application of all those rules. The saturation of a fact is the process of applying rules from the initial fact until no more new information can be added to the fact via rule application. Let the initial fact $F_0 = F$, and $F_i = \alpha(F_{i-1}, R)$, a fact is saturated when $F_i \equiv F_{i+1}$.*

Theorem 1 (Equivalence). *Let $K = (F, R)$ be a knowledge base and Q be a fact. The following assertions are all equivalent:*

- $K \models Q$
- *there exists a derivation F ... F' such that $F' \models Q$*
- *there exists an $n \in \mathbb{N}$ such that $F_n, R \models Q$*

When there are no rules in the ontology, the problem is then equivalent to homomorphism computation, which is a NP-hard problem. In the presence of rules, the problem is undecidable. Both the forward chaining and backwards chaining mechanisms are not certain of halting. This is easy to verify through the means of very simple examples.

Forward chaining Let $K = (F,R)$ be a knowledge base, $F = person(Bob)$, and $R = \{\{person(x)\} \rightarrow \{parent(y, x), person(y)\}\}$.

Let $Q = \{parent(x,Tom)\}$ be a fact. Asking a forward chaining mechanism if Q can be deduced from K may eventually never stop. The mechanism will first verify if Q can be deduced from F, if there is an x having Tom as parent in F. As it is not the case, rules will be applied and F will be enriched into $F' = \{person(Bob),parent(p_1, Bob),person(p_1)\}$. The mechanism will then verify if Q can be deduced from F'. As it is still not the case, it will once again apply rules and enrich F' into F". And it will do it infinitely as in this case, no answer will be ever found to the query.

Backwards chaining Let $K = (F,R)$ be a knowledge base, and $R = \{\{p(x, y),p(y, z)\} \rightarrow \{p(x, z)\}\}$.

Let $Q = \{p(a, b)\}$ be a fact. Asking a backwards chaining mechanism if Q can be deduced from K may also eventually never stop. The mechanism will first verify if $\{p(a, b)\}$ can be deduced from F. If that is the case, the mechanism will stop. Otherwise, it will rewrite the initial query Q into a new query $Q_1 = \{p(a, x_0),p(x_0, b)\}$. Q is deduced from K if Q_1 is deduced from K. If Q_1 can not be deduced from F, the mechanism will rewrite the query again, for example with $Q_2 = \{p(a, x_0),p(x_0, x_1),p(x_1, b)\}$. Such sequence of rewritings may never end. Any finite rewriting corresponds to a finite sequence, for example, of length k of form $\{p(a, x_0)$ ... $p(x_k, b)\}$. The facts could always contain a sequence of length $k + 1$.

4 Multi Scale Representation and Reasoning

In this section we introduce the notion of layered graphs that form the basis of our proposal for multi scale knowledge representation and reasoning. We detail the syntax and the semantics of layered graphs as well as a graph based projection inspired algorithm.

4.1 Representing multi scale knowledge

We introduce the concept of a "complex node" that intuitively will be the node that will be expanded to generate the layers in the hierarchical representation. The complex nodes will only be concept nodes but their neighbours (relation nodes) will also be expanded. In order to formalise the transition from one level to the other, we introduce transitional descriptions.

Definition 1. *Let $G = (V_C, V_R; E_G)$ be a bipartite graph. A transitional description associated to G is a pair $\mathcal{TD} = (D, (G.d)_{d \in D \cup N_G(D)})$ where*

- *$D \subseteq V_C$ is a set of complex nodes.*
- *For each $d \in D \cup N_G(D)$ $G.d$ is a bipartite graph.*
- *If $d \in D$ then $G.d$ is the description of the complex node d. Distinct complex nodes $d, d' \in D$ have disjoint descriptions $G.d \cap G.d' = G^{\emptyset}$.*
- *If $d \in N_G(D)$ then either $G.d = G^{\emptyset}$ or $G.d \neq G^{\emptyset}$ and, in this case, $N_G(d) - D \subseteq V_C(G.d)$ and $V_C(G.d) \cap V_C(G.d') \neq \emptyset$ if and only if $d' \in N_G(d) \cap D$.*

A transitional description of a bipartite graph G provides a set D of complex nodes in one of the classes of the bipartition, each complex node having associated a description. This descriptions are disjoint bipartite graphs.

The neighbors of complex nodes either have empty descriptions or are described as bipartite graphs. These bipartite graphs contain in one of the classes of the bipartition, (V_C), all the atomic neighbors of the initial graph. The remaining nodes in each of these classes are new nodes or are taken from the descriptions of the corresponding complex neighbors of the initial graph.

Definition 2. *If $\mathcal{TD} = (D, (G.d)_{d \in D \cup N_G(D)})$ is a transitional description associated to the bipartite graph $G = (V_C, V_R; E_G)$, then the graph $\mathcal{TD}(G)$ obtained from G by applying $\mathcal{TD}$ is constructed as follows:*

1. *Take a new copy of $\lfloor V_C - D \rfloor_G$.*
2. *For each $d \in D$, take a new copy of the graph $G.d$ and make the disjoint union of it with the current graph constructed.*
3. *For each $d \in N_G(D)$, identify the nodes of $G.d$ which are already added to the current graph (i.e. the atomic nodes of G that are neighbours of d and the nodes of $G.d'$ which appear in $G.d$). For each complex neighbour d' of d in G, add the remaining nodes of $G.d$ as new nodes in the current graph and link all these nodes by edges as described in $G.d$ (in order to have an isomorphic copy of $G.d$ as a subgraph in the current graph).*

Figure 1 shows an example of transitional description, where graph G has $V_C = \{a, b, c, d, e\}$, $V_R = \{1, 2, 3, 4\}$ and the set of complex nodes from V_C (shown as bold rectangles) is $D = \{a, c, d\}$. The description of these nodes, namely $G.a$, $G.c$, and $G.d$, follows the same rule of node labelling (i.e. rectangle nodes are denoted by letters and oval nodes are denoted by numbers) and has a prefix association. $N_G(D)$ is the set $\{1, 2, 3\}$ whose description is shown in Figure 1.

Note that the description associated with node 1 is the empty graph. The nodes of $V_C(G.2)$ are (i) the atomic node b of G, (ii) the nodes ca and cb from $G.c$, and (iii) the node dc from $G.d$, since $N_G(2) = \{b, c, d\}$, b is an atomic node of G and c, d are complex nodes of G. In the description of $G.3$, a new node $3a$ appears besides nodes e and db, dc.

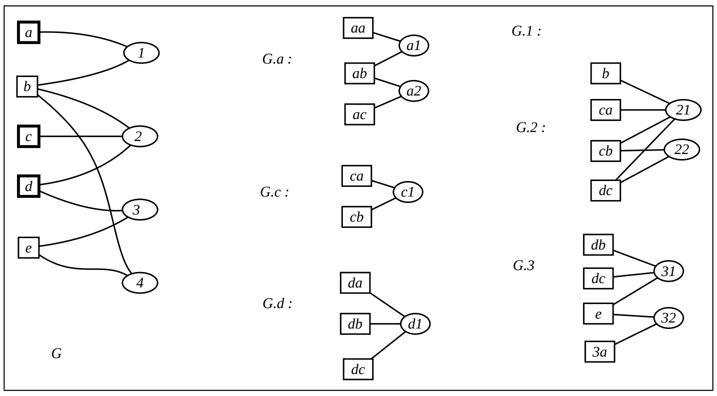

Figure 1. Transitional Description

Figure 2 illustrates this construction for graph G and for the transitional description of G depicted in Figure 1.

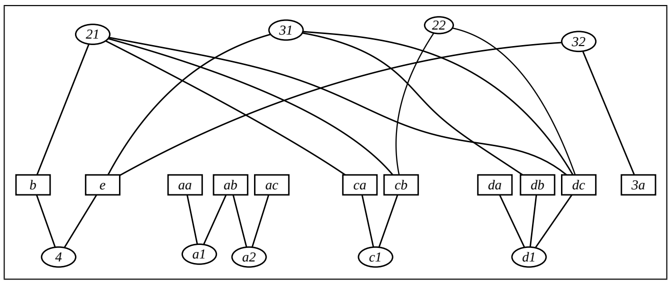

Figure 2. The Graph Obtained from Applying a Transitional Description

Theorem 2. *If $G = (V_C, V_R; E_G)$ is a bipartite graph and $\mathcal{TD}$ is a transitional description associated to G, then the graph $\mathcal{TD}(G)$ obtained from G by applying $\mathcal{TD}$ is also a bipartite graph.*

Proof. Let $H = \mathcal{TD}(G)$. By the construction of the graph H described in the previous definition, we have $V(H) = V_C(H) \cup V_R(H)$, where

$$V_C(H) = (V_C(G) - D) \cup \cup_{d \in D \cup N_G(D)} V_C(G.d)$$

and

$$V_R(H) = V_R(\lfloor V_C(G) - D \rfloor_G) \cup \cup_{d \in D \cup N_G(D)} V_R(G.d).$$

Furthermore, each edge of the graph H is either from $\lfloor V_C(G) - D \rfloor_G$ or from some bipartite description, hence has an endpoint in $V_C(H)$ and the other in $V_R(H)$.

Note that if $G.d = G^{\emptyset}$ for $d \in N_G(D)$, we have no description available for relation vertex d. This either depends on a lack of information or on an inappropriate expounding. The idea traces back to the notion of context in [18] or to the more elaborate notion of nested conceptual graph [10]. However, as our approach is not just a diagrammatic representation, the bipartite graph structure is taken into account.

This hierarchical representation allows, if we have a interconnected world described by a set of facts (a bipartite graph) and if we can provide details about both some complex concepts and their relationships, to construct a second level of knowledge about this world, describing these new details. This process can be similarly performed with the last constructed level, thus obtaining a coherent set of layered representations of the initial world. Please note that at each level, we need to highlight the specific set of hierarchical rules (concepts and rules) for that level. This means that at different levels we can use different terminologies that are specific to that level.

Definition 3. *Let d a nonegative integer. A hierarchical KB of depth d is a family* $\mathbf{HG} = \langle\ G^0, \mathcal{TD}^0, \ldots, \mathcal{TD}^{d-1}\ \rangle$ *where:*

- $G^0 = [S^0, (V_C^0, V_R^0; E^0)]$ *is a knowledge base composed of a support S and a set of facts $(V_C^0, V_R^0; E^0)$;*
- $\mathcal{TD}^0$ *is a transitional description associated to G^0,*
- *for each k, $1 \leq k \leq d-1$, $\mathcal{TD}^k$ is a transitional description associated to $G^k = [S^k, (V_C^k, V_R^k; E^k)] = \mathcal{TD}^{k-1}(G^{k-1})$.*

We can define substructures of the HG model which can be useful to devise a customisable and versatile functionality that deals with large graph structures.

Definition 4. *If* $\mathbf{HG_1} = \{HG_1^0, \ldots, HG_1^{d_1}\}$ *and* $\mathbf{HG_2} = \{HG_2^0, \ldots, HG_2^{d_2}\}$ *are two hierarchical KBs, then* $\mathbf{HG_2}$ *is a subgraph of* $\mathbf{HG_1}$ *if $d_2 \leq d_1$ and $\exists k$, $0 \leq k \leq d_1 - d_2$, such that for all $i \in \{0, \ldots, d_2\}$: G_2^i is a subgraph of G_1^{k+i}, $D_2^i \subseteq D_1^{k+i}$ and $G_2.d$ is a subgraph of $G_1.d$ for each $d \in D_2^i \cup N_{G_2^i}(D_2^i)$.*

4.2 Reasoning with multi scale knowledge knowledge

Graph projection can be extended to hierarchical structures; however, with knowledge integration in mind, we only consider the case where queries are simple graphs (and not hierarchical graphs).

Definition 5. *A descending path of length k in $HG = \langle\ G^0, \mathcal{TD}^0, \ldots, \mathcal{TD}^{d-1}\ \rangle$ is a sequence $P = v_0, \ldots, v_k$ $(k \leq d)$,where $v_i \in V(G^i)$ and, for each i, $1 \leq i \leq k$, condition $v_i \in V(G.v_{i-1})$ holds. The last vertex of P is denoted as $end(P)$. Moreover k, i.e. the length of P, is denoted by $length(P)$. The set of all descending paths of HG is referred as $\mathcal{P}(HG)$.*

Definition 6. *Let $HG = \langle\ G^0, \mathcal{TD}^0, \ldots, \mathcal{TD}^{d-1}\ \rangle$ be a hierarchical knowledge base and Q a query. A hierarchical projection from Q to HG is a mapping $\Pi : V_C(Q) \cup V_R(Q) \rightarrow \mathcal{P}(HG)$ such that $\forall v \in V(Q)$, if $\Pi(v) = P_v$:*

- *if $v \in V_C(Q)$, then $end(P_v) \in V_C(G^{length(P_v)}) - D^{length(P_v)}$ and if $v \in V_R(Q)$ then $end(P_v) \in V_R(G^{length(P_v)}) - N_{G^{length(P_v)}}(D^{length(P_v)})$;*
- *$\forall c \in V_C(Q)$, $\forall r \in V_R(Q)$ if $c = N^i_G(r)$, then $length(P_c) \leq length(P_r)$ and for each v on P_r at distance k from the start vertex of P_r such that $length(P_c) \leq k \leq length(P_r)$, we have $N^i_{G^k}(v) = end(P_c)$.*

If there is projection from Q to HG, then *HG subsumes Q*. Similar as explained in the previous section we an consider a logical semantics of the hierarchical knowledge bases and show its soundness with respect to graph operations. This semantics is sketched below.

Let $\mathbf{HG} = \langle\ G^0, \mathcal{TD}^0, \ldots, \mathcal{TD}^{d-1}\ \rangle$ be a hierarchical knowledge base and $S_{HG} = (T_C, T_R)$ be the union of the supports of its levels. A ternary predicate is assigned to each concept type from T_C, and an $n+1$-ary predicate is assigned to each relation type of arity n from T^n_R. Each predicate has the same name as the element of the support it is associated to. If $t \in T_C$, then the ternary predicate $t(x, y, z)$ holds. Intuitively, this means that (i) at level x, y is a concept vertex, (ii) the concept represented by this vertex is z, and (iii) its type is t. Similarly, if $t \in T^n_R$, then predicate $t(x, z_1, \ldots, z_n)$holds. This means that (i) a relation vertex on the level x exists and that (ii) the relation represented by this vertex is $t(z_1, \ldots, z_n)$.

The formula $\Psi^*(HG)$ is constructed as the existential closure of x_k, where $0 \leq k \leq d-1$ are the variables that represent the levels and $\Psi^*(G^k)$ be the formula obtained by adding x_k as the first argument of each member predicate for every level. Then,

$$\Psi^*(HG) = \exists x_0 \exists x_1 \ldots \exists x_{d-1}(\wedge_{k=0}^{d-1} \Psi^*(G^k))$$

Theorem 3. *Hierarchical projection is sound and complete with respect to Ψ^*.*

The proof is similarly to [11].

Since the semantics presented here are similar to the semantics of [11] a few words to compare the two formalisms are necessary. Transitional descriptions are a syntactical device which allows a successive construction of bipartite graphs. The knowledge detailed on a level of a hierarchy is put in context by using descriptions for relation nodes as well, while [11] only details the concept nodes and thus can be viewed as a particular instance of the formalism shown here.

5 Conclusion

This paper presented a transformation system that could be an appropriate hierarchical model for real world applications that require consistent transformation at different granularity levels. We presented the syntax of the extension and then demonstrated that the syntactic extension proposed is accompanied by sound and complete reasoning mechanisms.

Future work will focus on the application of such formalisation on a clear used case issued from life sciences. We are interested to see if the syntactic extension (i.e. the transitional descriptions) are easily elicited from non computing end users. To this end specific interfaces must be carefully devised in order to easily capture such knowledge.

The sound and complete reasoning mechanisms based on graphs are of great importance in a practical setting. We are thus interested to see how the projection operation can be best visualised for non computing experts. If demonstrated, this will be one of the main salient points of using such expressive formalism based on graph based syntax as opposed to approaches such as [6].

6 Acknowledgements

The second author acknowledges the support of ANR grants ASPIQ (ANR-12-BS02-0003), QUALINCA (ANR-12-0012) and DURDUR (ANR-13-ALID-0002). The work of the second author has been carried out part of the research delegation at INRA MISTEA Montpellier and INRA IATE CEPIA Axe 5 Montpellier.

References

1. I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *Communications magazine, IEEE*, 40(8):102–114, 2002.
2. F. Baader. Logic-based knowledge representation. In *Artificial Intelligence Today, Recent Trends and Developments*, number 1600, pages 13–41, 1999.
3. F. Baader, S. Brandt, and C. Lutz. Pushing the el envelope. In *Proc. of IJCAI 2005*, 2005.
4. J.-F. Baget, M. Croitoru, and B. P. L. Da Silva. Alaska for ontology based data access. In *The Semantic Web: ESWC 2013 Satellite Events*, pages 157–161. Springer, 2013.
5. C. J. Baker and K.-H. Cheung. *Semantic web: Revolutionizing knowledge discovery in the life sciences*. Springer Science & Business Media, 2007.
6. A. Benczúr, C. Hajas, and G. Kovács. Datalog extension for nested relations. *Computers & Mathematics with Applications*, 30(12):51–79, 1995.

7. A. Calì, G. Gottlob, and T. Lukasiewicz. A general datalog-based framework for tractable query answering over ontologies. *Web Semantics: Science, Services and Agents on the World Wide Web*, 14:57–83, 2012.
8. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The dl-lite family. *J. Autom. Reasoning*, 39(3):385–429, 2007.
9. M. Chein and M.-L. Mugnier. Conceptual graphs: Fundamental notions. *Revue d'Intelligence Artificielle*, 6-4:365–406, 1992.
10. M. Chein and M.-L. Mugnier. Positive nested conceptual graphs. In D. Lukose, H. S. Delugach, M. Keeler, L. Searle, and J. F. Sowa, editors, *ICCS*, volume 1257 of *Lecture Notes in Computer Science*, pages 95–109. Springer, 1997.
11. M. Chein, M.-L. Mugnier, and G. Simonet. Nested graphs: A graph-based knowledge representation model with fol semantics. In S. C. S. Anthony G. Cohn, Lenhard K. Schubert, editor, *KR*, pages 524–535. Morgan Kaufmann, 1998.
12. M. Croitoru, E. Compatangelo, and C. Mellish. Hierarchical knowledge integration using layered conceptual graphs. In *Conceptual Structures: Common Semantics for Sharing Knowledge*, pages 267–280. Springer, 2005.
13. M. Croitoru, L. Xiao, D. Dupplaw, and P. Lewis. Expressive security policy rules using layered conceptual graphs. *Knowledge-Based Systems*, 21(3):209–216, 2008.
14. B. P. L. Da Silva, J.-F. Baget, and M. Croitoru. A generic platform for ontological query answering. In *Research and Development in Intelligent Systems XXIX*, pages 151–164. Springer, 2012.
15. J. Esch and R. Levinson. An implementation model for contexts and negation in conceptual graphs. In G. Ellis, R. Levinson, W. Rich, and J. F. Sowa, editors, *ICCS*, volume 954 of *Lecture Notes in Computer Science*, pages 247–262. Springer, 1995.
16. R. B. Handfield and E. L. Nichols. *Introduction to supply chain management*, volume 1. prentice Hall Upper Saddle River, NJ, 1999.
17. M.-L. Mugnier. Ontological query answering with existential rules. In *Proceedings of the 5th International Conference on Web Reasoning and Rule Systems*, RR'11, pages 2–23. Springer-Verlag, 2011.
18. J. Sowa. *Conceptual Structures: Information Processing in Mind and Machine.* Addison-Wesley, 1984.
19. H. Wache, T. Voegele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann, and S. Hübner. Ontology-based integration of information-a survey of existing approaches. In *IJCAI-01 workshop: ontologies and information sharing*, volume 2001, pages 108–117. Citeseer, 2001.