DOI: 10.2478/ijasitels-2025-0002

Subsummation automata in conceptual graphs

Daniel C. Crăciunean[1]

[1]*Computer Science and Electrical and Electronics Engineering Department, Faculty of Engineering, "Lucian Blaga" University of Sibiu, Romania*
daniel.craciunean@ulbsibiu.ro

Abstract

One of the most important constraints faced by the development of artificial intelligence systems is the limited computing power. The logic problems involved in the construction of artificial intelligence systems are most often NP-complete in terms of computational complexity. In the case of conceptual graphs, logical inference is based on the subsummation relation between conceptual graphs. In this paper, we use homomorphisms of conceptual graphs as a model for the subsummation relation. Therefore, the problems related to the existence of relations between two conceptual graphs such as homomorphisms, isomorphisms or generalization, specialization relations become fundamental problems. Although all these decision problems are, in general, NP-complex, they can be efficiently solved in many cases, by certain strategies, without making too much compromise in terms of generality. In this paper, we propose a method to find and manipulate homomorphisms of conceptual graphs based on the concept of finite automaton. Thus, using the finite automaton to recognize predicate homomorphisms and moving a large part of the complexity of finding a homomorphism to the knowledge base construction phase represents our main contribution in this work.

Keywords: conceptual graph, knowledge base, conceptual graph homomorphism, subsummation relation, subsummation automaton

1 Introduction

A knowledge-based system requires both a formalism for specifying knowledge and reasoning mechanisms on this knowledge in order to solve different types of problems. The language of classical first-order logic (FOL) remains the main formalism used for specifying knowledge. Classical logic, in its current form, has sophisticated reasoning mechanisms that can handle complex problems. However, two important problems arise: on the one hand, not all users master mathematical logic, and on the other hand, the computational complexity of some reasoning exceeds the computing capacity available today. Therefore, a large part of the research effort in this area focuses on finding a compromise between the expressiveness of the knowledge representation formalism and the efficiency of the reasoning mechanisms.

A modeling process begins with the conceptualization of the domain, that is, with the detection of all the concrete elements in the modeling domain and their replacement with concepts [6] [7]. Thus, the conceptualization provides a common vocabulary, which integrates heterogeneous elements in the modeling domain. A complete specification of the conceptualization of the domain is called an ontology [3].

The conceptualization of the domain therefore provides the basic component of the ontology, the vocabulary formed by a set of terms, which name entities and relationships between these entities, and which represent the concepts necessary for modeling the domain.

Depending on the objectives of the model, the semantics of the concepts and relationships included in the vocabulary can be specified in natural language in the case of informal ontologies, or in a formal, axiomatic or prototype-based language in the case of formal ontologies.

Formal ontologies form the basic support for reasoning in the modeled domain. Therefore, the information contained in the ontology will have to support logical reasoning in the modeled domain. For this purpose, vocabulary elements are structured by equivalence relations, in order to capture terms that define the same concepts, or by partial order relations to specify hierarchies of generalization and specialization.

The effort made by researchers to find a compromise between the expressiveness of the knowledge representation formalism and the efficiency of reasoning mechanisms has led to several specifications, among which two types of representations stand out, based on semantic networks, namely: description logics and graph-based representations [4].

Although the language offered by description logics allows the specification of only decidable formulas from ordinal one logic based on unary and binary predicates, practice has demonstrated that they cover a large part of the modeling requirements [5].

When knowledge in a knowledge base is represented as a graph, it is confused with this graph which is called Knowledge Graph. The semantics of a Knowledge Graph is defined by the domain ontology. Over time, a lot of Knowledge Graphs have been built, including: Google Knowledge Graph [8], DBpedia [9], Yago [10], WordNet [11].

In 1976 Sowa [1] defines a version of Knowledge Graph, based on a bipartite multigraph which he calls a conceptual graph (CG) and which represents an important fragment of first-order logic. Inference in conceptual graphs is based on a subsummation relation specified by graph homomorphisms.

One of the most important constraints faced by the development of artificial intelligence systems is limited computing power. Even the simplest artificial intelligence problems, appeal to logic problems that from the point of view of computational complexity are most often NP-complete. In the case of conceptual graphs, logical inference is based on the subsummation relation between conceptual graphs. In this paper, we use homomorphisms of conceptual graphs as a model for the subsummation relation. Therefore, problems related to the existence of relations between two conceptual graphs, such as homomorphisms, isomorphisms or generalization-specialization relations between two conceptual graphs, become fundamental problems. Although all these decision problems are, in general, NP-complex, they can be efficiently solved in many cases, by certain strategies, without making too much compromise in terms of generality.

In this paper, we propose a method to find and manipulate homomorphisms of conceptual graphs based on the concept of finite automaton. Thus, the use of the finite automaton for recognizing homomorphisms and moving a large part of the complexity of finding a homomorphism to the knowledge base construction phase represents our main contribution in this paper.

Section 2 of the paper contains Background Notions, section 3 introduces the finite automaton as a mechanism for recognizing isomorphisms of conceptual graphs, section 4 presents an algorithm for constructing a finite automaton that recognizes homomorphisms of conceptual graphs. In the last section, some conclusions and observations are presented.

2 Background Notions

Conceptual graphs were initially defined as graphical formalisms for modeling conceptual schemas specific to database systems, but soon found their place in many fields, especially related to artificial intelligence. Starting with the 80s, three directions in which they developed crystallized, namely: as a mechanism for representing knowledge and reasoning on this knowledge, as an interface between natural language and knowledge bases, and as a graphical language for representing an important fragment of first-order logic.

Most Knowledge Graph models are based on directed graphs with nodes of a single type, which represent conceptual entities, and edges between two nodes of the graph that express a semantic relationship between the two entities [13].

In the case of Conceptual Graphs, we have two types of nodes, namely: nodes that represent conceptual entities or concepts and nodes that represent conceptual relations, in which several conceptual entities participate, specified by edges of the graph. Conceptual relations are, most of the time, facts or actions expressed by verbs. The nodes that represent these actions or facts, together with the participants, which are the neighbouring entities, form elementary conceptual graphs in the shape of a star.

The nodes of the conceptual graph are marked with corresponding labels from the vocabulary of the modeled domain, depending on their meaning. Thus, the nodes that represent conceptual entities receive a type label and, possibly, a label that represents a representative of the respective type, and the nodes that represent actions receive only a type label. Also, because conceptual relationships will be represented by atomic formulas in first-order logic, and neighbouring entities represent the parameters of these formulas, the edges connecting neighbours are numbered in the order of their appearance in the parameter list.

Therefore, if the domain vocabulary is $V=(V_C, V_R, E)$ where: V_C contains the types of concepts, V_R, contains the types of relations, and E contains the individual labels, we can define a basic conceptual graph G over the vocabulary V as a tuple (V,G, η), where V is the vocabulary, $G=(C,R, \Gamma)$, is a bipartite multigraph, and $\eta=(\eta_C,\eta_R,\eta_\Gamma)$ is a marking application of the nodes and arcs of the graph (G) defined as follows: $\eta_C:C\rightarrow T_C\times (E\cup\{*\})$, $\eta_R:R\rightarrow T_R$; $\eta_\Gamma:\Gamma\rightarrow N$. We will sometimes denote the elements of Γ by triplets of the form (r,i,c) where r is a relational node, c is a conceptual node neighbouring the relational node r and i is the edge marking that represents the position of c in the parameter list of the relation c.

Both on the set of conceptual entities and on the set of conceptual relations an equivalence relation can be introduced called coreference relation. This equivalence relation determines classes of entities that specify the same entity or the same relation.

Basic conceptual graphs admit coreference only at the level of individual entities. Simple conceptual graphs are an extension of basic conceptual graphs that do not restrict coreference. Both basic conceptual graphs and simple conceptual graphs can be brought to a special form called normal form by cumulating all coreferent concepts into a single representative concept [2].

Practically, a conceptual graph in normal form no longer has different conceptual nodes that represent the same concept. It follows that, by normalizing conceptual graphs, the difference between normal basic conceptual graphs and normal simple conceptual graphs is eliminated. In the rest of this paper, we will use the name conceptual graph for both basic conceptual graphs and simple conceptual graphs.

To simplify the presentation, in the following, we will use the notation $\eta(x)$ for any $x \in C \cup R \cup \Gamma$, with the following meaning:

$$\eta(x)=\begin{cases} \eta_C(x) \ if \ x \in C \\ \eta_R(x) \ if \ x \in R \\ \eta_\Gamma(x) \ if \ x \in \Gamma \end{cases}$$

We will also use the simplified notation $G=(C,R,V,\eta)$, for a conceptual graph where, obviously, C is the set of conceptual nodes, R is the set of relational nodes, V is the vocabulary and η is the mapping of nodes and edges.

If $G= (C,R,V,\eta)$ is a conceptual graph then a conceptual subgraph G' of G is a conceptual graph $G'=(C',R',V',\eta')$ such that $C' \subseteq C$, $R' \subseteq R$, $V \subseteq V$ and η' is the corresponding restriction of η to the components of G'.

We observe that any conceptual subgraph of G can be obtained from G by repeatedly deleting isolated conceptual relations or nodes, no conceptual nodes neighbouring any relation can be deleted without also deleting the corresponding relation.

The subsummation relation in the case of conceptual graphs can be defined by homomorphisms.

If $G=(C_G,R_G,V,\eta_G)$ and $H=(C_H,R_H,V,\eta_H)$ are two conceptual graphs with a common vocabulary V then a homomorphism from G to H is a polymorphic mapping $\varphi:G \rightarrow H$ that maps conceptual nodes in G to conceptual nodes in H, relational nodes in G to relational nodes in H with the following conditions: if $(r,i,c) \in G$ then $(\varphi(r), i, \varphi(c)) \in H$ and for all nodes $x \in C_G \cup R_G$, we have $\eta_H(\varphi(x)) \leq \eta_G(x)$.

If there is a homomorphism from a conceptual graph G to a conceptual graph H then we say that G subsumes H and we denote this fact by $G \succcurlyeq H$. The subsummation relation $G \succcurlyeq H$ represents the fact that the semantics specified by H logically implies the semantics specified by G.

But determining the homomorphisms of conceptual graphs is not a simple problem [2]. In this paper we will show that using the notion of finite automaton for computing the homomorphisms of conceptual graphs comes with important advantages.

A finite, initial and deterministic automaton is a tuple $M = (I, S, O, f, g, s_0)$ where I, S, O are nonempty and finite sets called the input alphabet, the set of states and the output alphabet and $f:S \times I \rightarrow S$, $g: S \times I \rightarrow O$ are the transition and output functions of the automaton [12].

3 Finite automaton and conceptual graph isomorphisms

Conceptual graphs are generally defined as bipartite graphs, but they can be defined, perhaps even more intuitively, as hypergraphs [2]. A hypergraph H is defined as a pair H=(X, A), where: X is a set of objects called the vertices of H, and A is a family of multisets with objects in X, called the hyperedges of H. If the multisets in A are ordered then the hypergraph is called an ordered hypergraph.

The ordered hypergraph can conveniently replace the bipartite graph in the definition of the basic conceptual graph. Thus, a conceptual hypergraph H, on a vocabulary $V=(T_C,T_R,E)$ is a triple H = (X, A, η), where: (X, A) is an ordered hypergraph, and η is the labelling application that attaches to a vertex x a pair of labels $(type(x),marker(x))\in(T_C\in(E\cup\{*\}))$ and to a hyperedge r, a label $type(r)\in T_R$.

Each vertex of a conceptual hypergraph represents a conceptual entity and each of its hyperedges represents a conceptual relation with the arity given by the cardinality of the hyperedge. Therefore, there is a natural correspondence between a conceptual hypergraph and a basic conceptual graph. If H=(X, A, η), is a conceptual hypergraph then the conceptual graph equivalent to H is G=(C,R,V,η), with the property that, C is isomorphic to X, R is isomorphic to A, and the labelling application η is common.

We observe that if H=(X, A, η), is a conceptual hypergraph, then each element of the set A represents a StarCG, that is, a CG formed by a single relation node together with its neighbors. To simplify our approach, we will specify each element $a\in A$, by a list $\alpha=[a_1,\ldots a_n,r]$, where $a_1,\ldots a_n$ are nodes that represent conceptual entities and r is a node that represents a conceptual relation. We will denote the set of conceptual nodes components of a, by $C(\alpha)=\{a_1,\ldots a_n\}$, and the relational node component of a by $R(\alpha)$.

In the following, we will consider only hypergraphs that do not have isolated conceptual nodes. Under these conditions, a conceptual hypergraph without isolated vertices is completely specified by the set A together with the labelling application η. Therefore, we will specify a conceptual hypergraph H without isolated nodes as a pair H=(A, η), where the meanings of A and η are as specified above.

Since the pair H=(A, η), represents a conceptual graph where A is the set of corresponding StarCGs and η is the labelling application, we will call the pair H=(A, η) with the meaning specified above a conceptual graph.

Example 1. The conceptual graph in Figure 1 can be specified as a pair H=(A, η), where: $A=\{[a_2,a_1,r^1], [a_1,a_3,a_2,r^2], [a_3,a_4,r^3]\}$ and η is defined as follows: η(a1)=" Profesor", η(a2)="UndergradStudent", η(a3)="UndergradCourse", η(a4)="BachelorDegree", η(r1)=" HasMentor", η(r2)="Teaches", η(r3)="Curriculum".

If $H_1=(A_1, \eta_1)$ and $H_2=(A_2, \eta_2)$, are two conceptual graphs, without isolated vertices, then a homomorphism from H_1 to H_2, is a map $\varphi:A_1\rightarrow A_2$, which maps each StarCG $\alpha=[a_1,\ldots a_n,r^\alpha]\in A_1$, to a StarCG $\beta=[b_1,\ldots b_n,r^\beta]\in A_2$, such that $b_i=\varphi(a_i)$, $\eta_1(a_i)\geqslant\eta_2(b_i)$ for all $i=1,\ldots,n$ and $r^\beta=\varphi(r^\alpha)$, $\eta_1(r^\alpha)\geqslant\eta_2(r^\beta)$. It is obvious that the problem regarding the existence of a homomorphism between two conceptual graphs is equivalent to the problem regarding the existence of a homomorphism between two conceptual hypergraphs.

An isomorphism between two conceptual graphs $H_1=(A_1, \eta_1)$ and $H_2=(A_2, \eta_2)$, is a bijective mapping φ, from A_1 to A_2, with the property that for any StarCG $\beta=\varphi(\alpha)$, we have:

i) if $a_i \in C(\alpha)$ and $r^{\alpha}=R(\alpha)$ then $b_i \in C(\beta)$ and $r^{\beta}=R(\beta)$

ii) if $b_i \in C(\beta)$ and $r^{\beta}=R(\beta)$ then $a_i \in C(\varphi^{-1}(b_i))$ and $r^{\alpha}=R(\varphi^{-1}(r^{\beta}))$

iii) for all components $x \in C(\alpha) \cup R(\alpha)$ we have $\eta_1(x)= \eta_2(\varphi(x))$

Under these conditions if H=(A, η), is a conceptual graph then a conceptual subgraph H' of H, is a pair H'=(A', η') where A'⊆A and η' is the restriction of η to the components that appear in A'. If A' contains a single StarCG, we will also use the name StarCG for the conceptual graph H'.

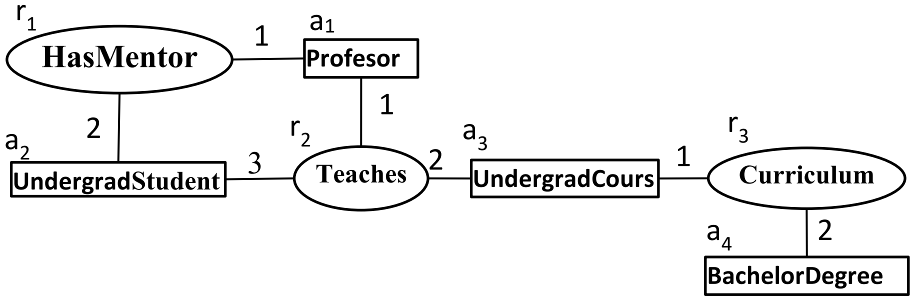

Figure 1. H – Conceptual Graph

Example 2. The conceptual graph H'=(A', η'), where: A'={[a_2,a_1,r^1], [a_1,a_3,a_2,r^2], [a_3,a_4,r^3]} and η' is defined as follows: η'(a1)="Profesor", η'(a2)="UndergradStudent", η'(a3)="UndergradCourse", η'(a4)="BachelorDegree", η'(r1)=" HasMentor", η'(r2)="Teaches", η(r3)="Curriculum" is a conceptual subgraph of H.

In this paper we will consider that all the conceptual graphs we use are brought to normal form. Normal form is not a restriction because any conceptual graph can be brought to normal form in polynomial time [2]. We observe that if $\eta=(\eta_C,\eta_R,\eta_\Gamma)$, is the marking application of a normalized conceptual graph then the component η_C is injective.

Next, we will specify a finite automaton associated with a conceptual graph G, which recognizes all the component StarCGs of G. If H=(A, η) is a conceptual graph, then each StarCG, $\alpha_k \in A$, is of the form $\alpha_k=[a_{k_1}, a_{k_2}, \ldots, a_{k_{|r_k|}}, r_k] \in A$, where we denote by $|r_k|$ the arity of r_k. The finite automaton attached to the StarCG component α_k (Figure 2) is defined as follows:

$M_k = (I_k, S_k, f_k, s_{0,k}, F_k)$, where:

$I_k=\{\eta(a_{k_1}), \eta(a_{k_2}), \ldots, \eta(a_{k_{|r_k|}}), \eta(r_k)\}$,

$S_k=\{a_{k_1}, a_{k_2}, \ldots, a_{k_{|r_k|}}, r_k\}$, $F_k =\{r_k\}$,

$f_k : S_k \times I$ is the transition function defined as follows: $f_k(s,\eta(a_k))=a_k$;

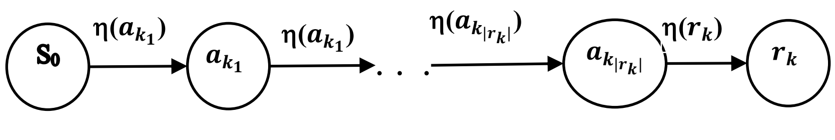

Figure 2. The finite automaton that recognizes a StarCG

For each StarCG component of a conceptual graph we can construct an elementary finite automaton.

Example 3. For the StarCG components of the conceptual graph H in Figure 1 we can construct the finite automata in Figure 3, defined as follows:

M_1=((I_1, S_1, f_1, $s_{0,1}$, F_1), where:

I_1={Profesor, UndergradStudent, HasMentor},

S_1={S_0, a_1, a_2, r_1} , F_1={r_1},

f_1:$S_1 \times I_1$ is the transition function defined as can be seen in Figure 3.

M_2=((I_2, S_2, f_2, $s_{0,2}$ F_2), where:

I_2={Profesor, UndergradCours, UndergradStudent, Teaches},

S_2={S_0, a_1, a_3, a_2, r_2} , F2={r_2},

f_2:$S_2 \times I_2$ is the transition function defined as can be seen in Figure 3.

M_3=((I_3, S_3, f_3, $s_{0,3}$ F_3), where:

I_3={UndergradCours, BachelardDegree, Curriculum},

S_3={S_0, a_3, a_4, r_3} , F3={r_3},

f_3:$S_3 \times I_3$ is the transition function defined as can be seen in Figure 3.

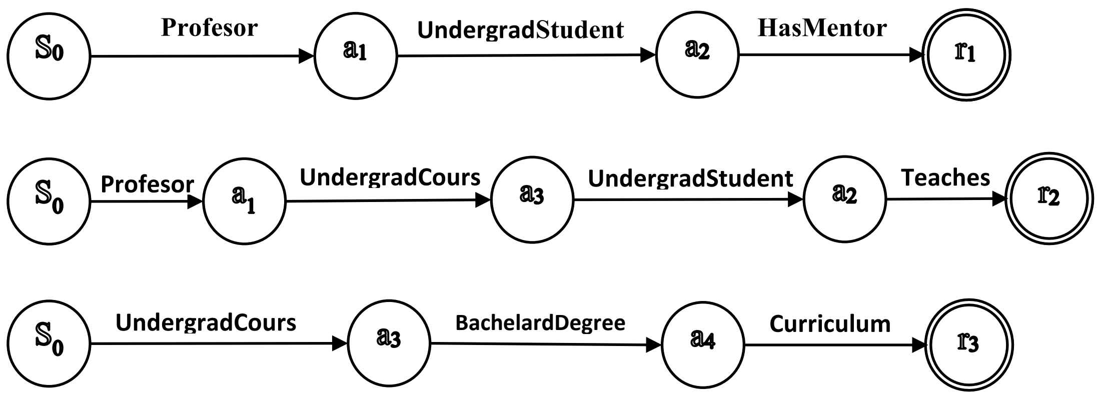

Figure 3. Finite automata associated with the conceptual graph H

Let there be two finite automata M_1 = (I_1, S_1, f_1, F_1) and M_2 = (I_2, S_2, f_2, F_2). The direct sum of the two automata is the automaton $M=M_1 \oplus M_2=(I_1 \cup I_2, S_1 \sqcup S_2, f_1 \oplus f_2, F_1 \sqcup F_2)$, where we denoted by ⊔ the disjoint sum of two sets of states, and $f_1 \oplus f_2$ and $g_1 \oplus g_2$ are defined as follows:

$$f_1 \oplus f_2(s,i)=\begin{cases} f_1(s,i) \text{ if } s \in S_1 \\ f_2(s,i) \text{ if } s \in S_2 \end{cases}$$

$$g_1 \oplus g_2(s,i)=\begin{cases} g_1(s,i) \text{ if } s \in S_1 \\ g_2(s,i) \text{ if } s \in S_2 \end{cases}$$

Example 4. The direct sum of the three automata specified in Example 3, $M=M_1 \oplus M_2 \oplus M_3$ can be seen in Figure 4.

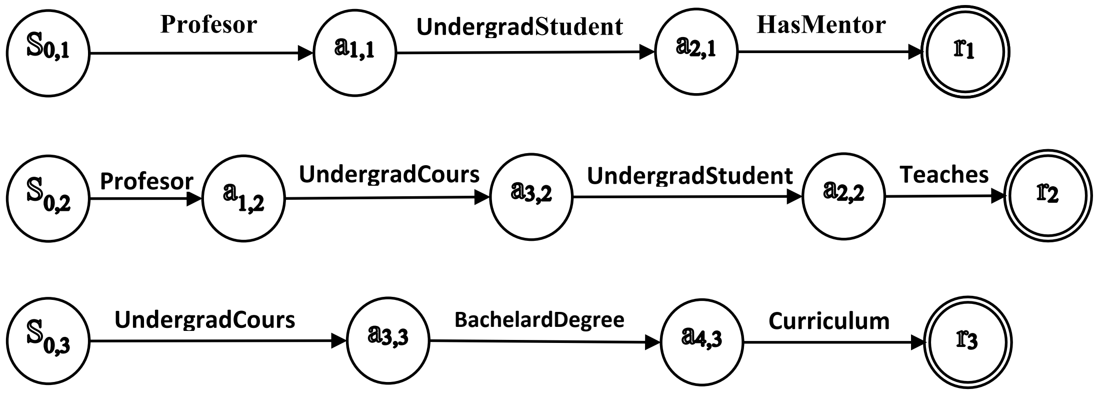

Figure 4. Direct sum of the automata in Figure 3

Direct sum only renames the states of the automata so that they are distinct. The direct sum of two conceptual graphs is a specialization operation that does not connect the two graphs. The direct sum of two finite automata behaves in the same way. We present below an algorithm that connects several automata into a single finite automaton. Moreover, the resulting automaton is deterministic.

The CONECT algorithm. This algorithm connects several finite automata into a single deterministic finite automaton.

Input: The input to the algorithm is represented by a set of finite automata M_k, $1 \le k \le n$.

Output: The initial deterministic finite automaton $M=(I, S, f, S_0, F)$ equivalent to the direct sum of the finite automata Mk, $1 \le k \le n$, from the input.

Method: In the first step, the algorithm calculates the nondeterministic finite automaton $N=(I_N, S_N, f_N, F_N)$, as the direct sum of all the automata M_k, $1 \le k \le n$. Then, the algorithm creates new states that are subsets of the set S_N, of states of the automaton N. The algorithm also defines a new transition function between the new states.

P1. The direct sum $N=M_1 \oplus M_2 \oplus \dots \oplus M_n=(I_N, S_N, f_N, F_N)$ is made. $I \leftarrow I_N$ is made.

P2. The initial state is defined $S_0=\{s_{0,1}, s_{0,2}, \dots, s_{0,n}\}$, where each $s_{0,k}$ is the initial state of the automaton M_k. Make $S \leftarrow \{S_0\}$ and $m \leftarrow 0$.

P3. As long as there are unmarked states in S, execute steps P4 and P5.

P4. Choose an unmarked state s from S and mark it.

P5. For each input $e \in I_N$, calculate the set $P_e=\{x \in S_N | \ x= f_N(s,e)\}$. If $P_e \neq \emptyset$, make $m \leftarrow m+1$, $S_m= P_e$, $S \leftarrow S \cup \{S_m\}$. Define $f(s,e) \leftarrow S_m$.

Example 5. The result of applying the CONECT algorithm for the automata specified in Example 3 can be seen in Figure 5.

If H is a conceptual graph then we will denote by M(H), the automaton attached to H. The automaton M(H) is obtained by applying the CONECT algorithm to the elementary automata corresponding to the component StarCGs of H.

We will say that a conceptual graph $G=(A_G, \eta_G)$ is recognized by the automaton M if the automaton M recognizes all the components of the StarCGs in AG.

It is obvious that if G=(A_G, η_G) and H=(A_H, η_H) are conceptual graphs and the conceptual graph G is recognized by the automaton M(H), then G is isomorphic to a conceptual subgraph of H.

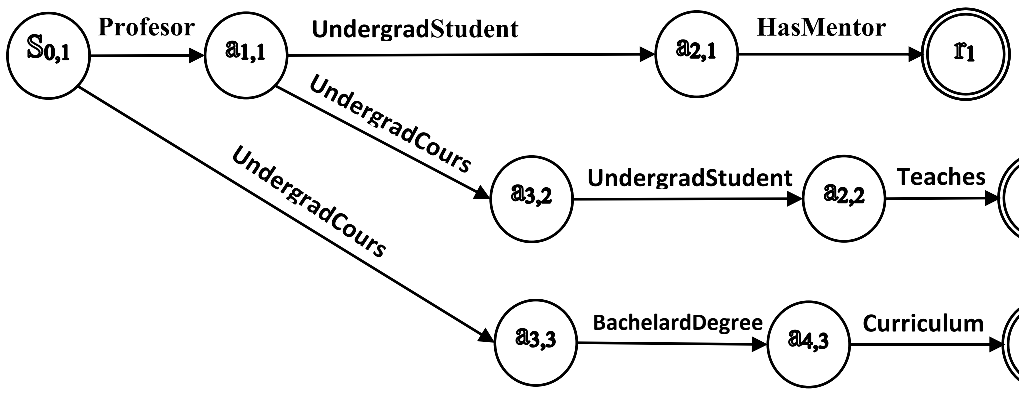

Figure 5. The finite automaton associated with the conceptual graph H

4 Finite automata and conceptual graph homomorphisms

As we saw in the previous section, a bijective homomorphism between two conceptual graphs is an isomorphism between the two bipartite graphs in the definition of a conceptual graph that preserves labels. Also in the previous section, we saw that to a conceptual graph H we can associate a finite automaton that recognizes any conceptual graph isomorphic to a conceptual subgraph of H.

Coreference and generalization relations are of utmost importance in specifying and determining inference between logical formulas expressed by conceptual graphs [14]. The generalization relation is defined at the level of conceptual and relational labels and extends to the level of conceptual graphs as a subsummation relation.

We have several important results regarding the connection between the generalization relation and conceptual graph homomorphisms [2]. One of these results is that a conceptual graph G is a generalization of a conceptual graph H if and only if there is a homomorphism from G to H. Also, a conceptual graph G is a generalization of a conceptual graph H if and only if H is a specialization of G. In this section we will define a finite automaton associated with a conceptual graph H that recognizes any conceptual graph G, with the property that there is a homomorphism of conceptual graphs from G to H. Our approach is based on the observation that, if we have two conceptual graphs G and H, then there is a homomorphism from G to H, if and only if there is a generalization of H isomorphic to G.

First, we will define three generalization operations applicable to StarCGs. The generalization operations for StarCGs are unary operations that apply to a StarCG G and return another StarCG H that is a generalization of G, that is, there is a homomorphism from H to G. These operations are, Split, Duplicate and Increase and will be applied to a StarCG G=(A={α},η) as described below.

The operation Split(G,c,c_1,c_2,l_1,l_2) receives as input a StarCG G, a conceptual node c existing in G, two names c_1 and c_2 for the new conceptual nodes and two new labels $l_1 \neq l_2$; $l_1,l_2 \notin C(\alpha)$, $l_1 \geq \eta(c)$ si $l_2 \geq \eta(c)$. The operation will replace the conceptual node c with two conceptual nodes c_1 and c_2, of the same type and will modify the labeling application as follows: $\eta(c_1)=l_1$ and $\eta(c_2)=l_2$.

We note that a concept node cannot be divided into two with the same label because the result is no longer normalized. A conceptual node can be divided into two concepts with different labels but both greater than or equal to the StarBG label.

The Duplicate(G,r,r_1) operation receives as input a StarCG G, a relational node r existing in G and a name r_1 for a new relational node. The operation simply adds a new relational node r_1, of the same type as r, with the same label and with the same neighbours.

The Increase(G,x,l) operation receives as argument a StarCG G, a conceptual or relational node x and a label $l\notin C(\alpha)$, $l>\eta(x)$. The operation modifies the labelling application such that $\eta(x)=l$.

Example 6. Consider the StarCG G=(A, η), from Figure 6, where: $A=\{[a_1,a_3,a_2,r^2]\}$ and η is defined as follows: $\eta(a_1)$="Profesor", $\eta(a_2)$="UndergradStudent", $\eta(a_3)$="UndergradCourse", $\eta(r_2)$="Teaches".

Suppose that in the vocabulary of the conceptual graph G we have the following hierarchy of concepts: TeachingStaff ≽ Profesor, Student ≽ UndergradStudent, Course ≽ UndergradCourse, Degree ≽ BachelorDegree , Attends ≽ Teaches.

With this hierarchy the elementary automata in Figure 7 are the automata associated with some examples of generalizations of G.

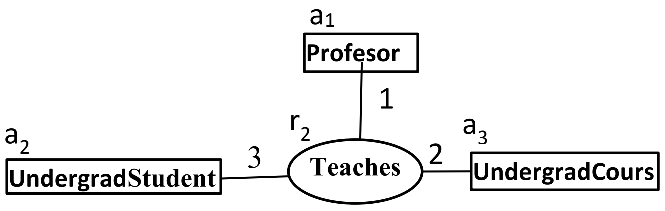

Figure 6. Elementary Conceptual Graph (StarCG)

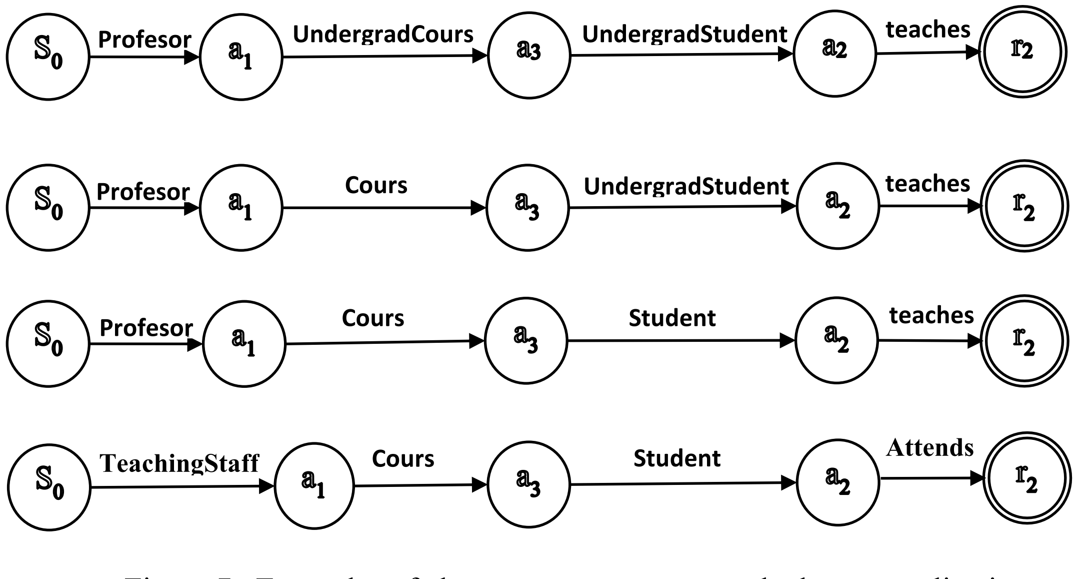

Figure 7. Examples of elementary automata attached to generalizations

Let us now consider a conceptual graph H=(A, η) , where A={$\alpha_1,\alpha_2,\ldots,\alpha_n$}. Then the conceptual graph H is the union of its constituent StarCGs of the form $H_k=(A_K, \eta_k)$, where each component A_K= {$\alpha_k=[a_{k_1}, a_{k_2}, \ldots, a_{k_{|r_k|}}, r_k]$} and η_k is the restriction of the map η to A_K.

Let H=(A, η), A={$\alpha_1,\alpha_2,\ldots,\alpha_n$} be a conceptual graph and $H_k=(A_K, \eta_k)$, $1\le k\le n$. We will denote by $\mathcal{G}_H$ the set of all normalized conceptual graphs that generalize StarCG components of H:$\mathcal{G}_H$={G| ∃ k≥1 such that $G \succcurlyeq H_k$}.

If we have a conceptual graph G=(A, η) then for any two StarCGs α,β∈A we have $\eta_\alpha(x)=\eta_\beta(x)=\eta(x)\ \forall x\in C(\alpha)\cap C(\beta)$. Therefore, two StarCGs, $G=(A_G, \eta_G)$ and $H=(A_H, \eta_H)$ can be assembled into a single conceptual graph (A, η) if and only if $\eta_G(x)=\eta_H(x)$ $\forall x\in C(A_G)\cap C(A_H)$ si $\eta_G(r)=\eta_H(r)$ $\forall x\in R(A_G)=R(A_H)$.

We will now introduce a partial union operation of two normalized conceptual graphs.

Let $G=(A_G,\eta_G)$ and $H=(A_H,\eta_H)$ be two conceptual graphs such that for any two StarCGs $\alpha\in A_G$ and $\beta\in A_H$ we have $\eta_G(x)=\eta_H(x)\ \forall x\in C(\alpha)\cap C(\beta)$. The union of G with H is the conceptual graph P=($A_G \cup A_H$, η) where η is defined as follows:

$$`\eta(x)=\begin{cases}\eta_G(x)\ if\ x\in C(G)\cup R(G)\\ \eta_H(x)\ if\ x\in C(H)\cup R(H)\\ \eta_G(x)\ if\ x\in \big(C(G)\cap C(H)\big)\cup(R(G)\cap R(H))\end{cases}$$

We must note that the union of two conceptual graphs can only be done when the two labelling applications coincide on the common conceptual or relational nodes. In our case we will also add the condition that the result of the union of two normalized conceptual graphs is also a normalized conceptual graph.

For a conceptual graph H, we now consider the set of all conceptual graphs that can be obtained by the operation of partial union of graphs in $\mathcal{G}_H$, that is, the closure of this set which we denote by $\mathcal{G}_H^*$. It is obvious that $\mathcal{G}_H^*$ contains all normalized generalizations of H.

We have a very important result for our approach, namely that, if we have two conceptual graphs G and H, then there is a homomorphism from G to H, if and only if there is a generalization of H isomorphic to G [2].

Using the CONECT algorithm applied to the set $\mathcal{G}_H$, we obtain an automaton that recognizes all conceptual graphs in $\mathcal{G}_H^*$. We denote by $\mathcal{M}$(H) the automaton: $\mathcal{M}$(H)=CONECT($\mathcal{G}_H$) and call it a subsummation automaton.

We have seen that a normalized conceptual graph $H=(A_H, \eta_H)$ is recognized by the automaton M if the automaton M recognizes all StarCG components in A_H. From this it follows that, if $G=(A_G, \eta_G)$ and $H=(A_H, \eta_H)$ are conceptual graphs and the conceptual graph G is recognized by the automaton $\mathcal{M}$(H), then G is isomorphic to a conceptual graph $G_1\in\mathcal{G}_H^*$, that is, there is an isomorphism $\eta:G\rightarrow G_1$. But since G_1 is a generalization of H, it follows that there is a natural homomorphism $\pi:G_1\rightarrow H$. It follows that the map $\varphi=\pi o \mu$, is a homomorphism from the conceptual graph G to the conceptual graph H.

5 Observations and conclusions

The ontological component of a knowledge base contains general knowledge about the domain, which is called ontological knowledge. But knowledge bases contain, in addition to ontological knowledge, various facts and statements about specific entities, which are logical formulas with truth values, and which are called factual knowledge.

Thus, a knowledge base models an open world, which includes information about a potentially infinite set of entities, by using universal quantifiers applied to unknown individuals. The main purpose of a knowledge base is achieved by the associated inference engine, which deduces new knowledge by logical reasoning on some implicit information in the knowledge base.

In the case of conceptual graphs, logical inference is based on homomorphisms of conceptual graphs. This paper proposes the use of finite automata to determine the existence of homomorphisms between two conceptual graphs. Our approach involves moving the complexity of the problem from the phase of determining the homomorphisms to the phase of building the knowledge base.

We note that in our approach the knowledge base must contain, in addition to the conceptual graph, the attached finite subsummation automaton. We can also observe that the subsummation automaton can reach a very large number of states. To eliminate this shortcoming, this automaton will have to undergo an optimization process that will be followed in subsequent works.

References

[1] Sowa, John F., “Conceptual graphs for a database interface, IBM Journal of Research and Development”, 20:4, 336-357, (1976).
[2] Michel Chein, Marie-Laure Mugnier, “Graph-based Knowledge Representation, Computational Foundations of Conceptual Graphs”, Springer-Verlag London Limited, (2009).
[3] Thomas R. Gruber, “A translation approach to portable ontology specifications”, Knowledge Acquisition Volume 5, Issue 2, June 1993,Pages 199-220, (1993)
[4] Lehmann D, Magidor M, “What does a conditional knowledge base entail?” Artificial Intelligence, Volume 55, Issue 1, May 1992, Pages 1-60, (1992)
[5] Bienvenu M, Ortiz M, “Ontology-mediated query answering with data-tractable description logics”. Lecture notes of the 11th international reasoning web summer school. LNCS, vol 9203, Springer, Berlin, pp 218–307, (2015)
[6] Daniel-Cristian Craciunean, Daniel Volovici, “Conceptualization of Modeling Method in the Context of Categorical Mechanism”, in Dimitris Karagiannis and others, Domain Specific Conceptual Modeling, Springer Nature Switzerland AG (2022).
[7] Daniel-Cristian Craciunean, “Categorical Mechanisms in Multi-level Modeling Methods”, Publishing House of the "Lucian Blaga" University of Sibiu, (2023).
[8] Ehrlinger L, Wöß W, “Towards a definition of knowledge graphs”, Joint Proceedings of the Posters and Demos Track of 12th International Conference on Semantic Systems - SEMANTiCS2016 and 1st International Workshop on Semantic Change & Evolving Semantics (SuCCESS16) 48(1–4):2, (2016)

[9] Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z., “DBpedia: A Nucleus for a Web of Open Data.” In: Aberer, K., et al. The Semantic Web. ISWC ASWC 2007 2007. Lecture Notes in Computer Science, vol 4825. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-76298-0_52, (2007)
[10] Rebele T, Suchanek F, Hoffart J et al, “Yago: a multilingual knowledge base from wikipedia, wordnet, and geonames.” In: International semantic web conference, Springer, p 177–185, (2016)
[11] Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi, “WordNet::Similarity - Measuring the Relatedness of Concepts.” In Demonstration Papers at HLT-NAACL 2004, pages 38–41, Boston, Massachusetts, USA. Association for Computational Linguistics, (2004).
[12] K. V. N Sunitha, N. Kalyani, “Formal Languages and Automata Theory”, Pearson India Education Services Pvt. Ltd, (2015)
[13] Bordes A, Weston J, Collobert R et al, “Learning structured embeddings of knowledge bases”, Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI, San Francisco, California, USA, August 7-11, (2011)
[14] Jamie Caine, Simon Polovina, “From Enterprise Concepts to Formal Concepts: A University Case Study, in Graph Structures for Knowledge Representation and Reasoning”, 5th International Workshop, GKR 2017 Melbourne, VIC, Australia, August 21, 2017, Revised Selected Papers, Lecture Notes in Artificial Intelligence, (2017)