Reasoning in description logics with variables: preliminary results regarding the EL logic.

Lakhdar Akroun, Lhouari Nourine, Farouk Toumani

Reasoning in description logics with variables: preliminary results regarding the $\mathcal{EL}$ logic

Lakhdar Akroun, Lhouari Nourine, and Farouk Toumani

[1] INRIA, Grenoble, France
lakhdar.akroun@inria.fr
[2] LIMOS, CNRS, Blaise Pascal University, Clermont-Ferrand, France
nourine, ftoumani@isima.fr

Abstract. This paper studies the extension of description logics with variables ranging over infinite domains of concept names and role names. As a preliminary work, we consider more specifically the extension of the logic $\mathcal{EL}$ with variables and we investigate in this context two reasoning mechanisms, namely compliance (a kind of matching) and pattern containment. The main technical results are derived by establishing a correspondance between the $\mathcal{EL}$ logic and finite variable automata.

1 Introduction

We consider description logics augmented with variables ranged over concept names and role names. As an example, consider the following description:

$$B \equiv Person \sqcap \exists works\text{-}for.X \sqcap \exists graduated\text{-}from.X \qquad (1)$$

where the variable X takes its values from an infinite set of possible atomic concept names. B specifies the set of persons that work for the same type of organization they were graduated from. Specifications of type (1) are called hereafter a *pattern definition* and B is called a *pattern name* (or simply a pattern). An *instanciation* of a pattern is given by variable valuations. For example, if the variable X is assigned as value the atomic concept name *University* (respectively, *eSchool*), we obtain the following description B_1 (respectively, B_2) which is *compliant* with the pattern B:

$$B_1 \equiv Person \sqcap \exists works\text{-}for.University \sqcap \exists graduated\text{-}from.University$$

$$B_2 \equiv Person \sqcap \exists works\text{-}for.eSchool \sqcap \exists graduated\text{-}from.eSchool$$

Variables can also be used in role places as illustrated by the following pattern definition:

$$B_3 \equiv Person \sqcap \exists Y.Z \sqcap \exists .graduated\text{-}from.Z$$

The concept B_3 specifies the set of persons that have a relation (i.e., any kind of role) with the same type of organization they are graduated from. Indeed, the

concept B_1 is compliant with B_3 and it is also the case of the concept B_4 defined as follows:

$$B_4 \equiv Person \sqcap \exists evaluates.eSchool \sqcap \exists graduated\text{-}from.eSchool$$

Our framework supports terminological cycles as illustrated below with the pattern B_5 which specifies the persons that work for the same type of organization they are graduated from and have a relative who also work for the same type of organization she is graduated from.

$$B_5 \equiv Person \sqcap \exists works\text{-}for.X' \sqcap \exists graduated\text{-}from.X' \sqcap \exists has\text{-}relative.B_5$$

For a description logic $\mathcal{L}$, we denote by $\mathcal{L}_\mathcal{V}$ the obtained logic augmented with concept variables and role variables. We study the following reasoning mechanisms in this framework (formal definitions are given later in the paper):

- Compliance which asks whether a description E is *compliant* with a pattern C.
- Pattern containment which, given two pattern definitions C_1 and C_2, asks whether every description E compliant with C_1 is also compliant with C_2.

Indeed, the notion of a concept pattern (i.e., a concept description containing variables) is not new and has already been used, in particular, in the context of two non-standard reasonings, namely matching [1] and unification [2,4]. Given a concept pattern D and a concept description C, the matching problem asks whether there is a substitution σ of the variables by concept descriptions such that $C \sqsubseteq \sigma(D)$. Unification is a generalization of matching. It takes as input concept patterns and asks whether there exist a substitution of the variables by concept descriptions that makes the concept patterns equivalent. Our definition of concept patterns deviate from the one used in the literature with respect to the following features: (i) our definition of concept patterns is more liberal in the sense that we allow concept variables as well as role variables while usually only concept variables are allowed in concept patterns, (ii) we support cyclic pattern definitions and, inspired from the guarded variable automata theory, we consider two different types of semantics of variables (i.e., refreshing and not refreshing semantics), and (iii) our interpretation of variables is however more restrictive in this paper since we consider only *atomic variables* (i.e., we assume that variables take their values from an infinite set of atomic concept names and role names). An extension of this framework to variables that stand for descriptions would be an interesting research direction. From the reasoning perspective, our notion of compliance coincides with matching however, up to our knowledge, the notion of pattern containment has never been investigated in the literature.

We expect the proposed framework to be useful in various applications. For example, concept patterns have already been proven to be beneficial in applications where a given user is interested *'to search the knowledge base for concepts having a certain not completely specified form'* [6]. Targeting a similar purpose, we envision our framework to be useful as query language to: (i) formulate queries

over terminologies. For example, the concept B_3 given above can be viewed as a query over a given terminology while the concepts B_1 and B_2 are examples of *answers* to such a query, or (ii) to formulate ontological queries (i.e., queries over an ontology) in the context of an Ontology Based Data Access (OBDA) approach [7]. The reasoning mechanisms studied in this paper can prove particularly relevant to handle query evaluation and optimisation in such contexts.

Organization of the paper. Section 2 presents some preliminary notions regarding state machines and guarded variable automata. Section 3 recalls some basic notions of the $\mathcal{EL}$ description logic and describes the extension of this logic with variables, the obtained logic is called $\mathcal{EL}_{\mathcal{V}}$. Section 4 studies the compliance and the containment problems in the context of the $\mathcal{EL}_{\mathcal{V}}$ logic. We conclude and draw future research directions in section 5. Proofs are omitted and are included in the extended version of this paper [13].

2 Preliminaries

We first recall the notion of state machines [9] and simulation preorder between state machines. A State Machine M is a tuple $< \Sigma_M, Q_M, F_M, Q^0_M, \delta_M >$, where: Σ_M is a finite alphabet, Q_M is a set of states with $Q^0_M \subseteq Q_M$ the set of initial states and $F_M \subseteq Q_M$ the set of final states, $\delta_M \subseteq Q_M \times \Sigma_M \times Q_M$ is a set of labeled transitions. If Q_M is finite then M is called a finite state machine. If $q \in Q^0_M$ is an initial state of a machine M, we say that M is *rooted at* q.

Let $M =< \Sigma_M, Q_M, F_M, Q^0_M, \delta_M >$ and $M' =< \Sigma_{M'}, Q_{M'}, F_{M'}, Q^0_{M'}, \delta_{M'} >$ be two (eventually, infinite) state machines. A state $q_1 \in Q_M$ is simulated by a state $q'_1 \in Q_{M'}$, noted $q_1 \preceq q'_1$, iff the following two conditions hold : (i) $\forall a \in \Sigma_M$ and $\forall q_2 \in Q_M$ such that $(q_1, a, q_2) \in \delta_M$, there exists $(q'_1, a, q'_2) \in \delta_{M'}$ such that $q_2 \preceq q'_2$, and (ii) if $q_1 \in F_M$, then $q'_1 \in F_{M'}$. M is simulated by M', noted $M \preceq M'$, iff $\forall q_M \in Q^0_M, \exists q'_{M'} \in Q^0_{M'}$ s.t. $q_M \preceq q_{M'}$.

We briefly introduce now the notion of (Guarded) Variable Automata [5]. Let $\mathcal{X}$ be a finite set of variables and Σ an infinite alphabet of letters. The set $\mathcal{G}$ of guards is inductively defined as follows: $G := true \mid \alpha = \beta \mid \alpha \neq \beta \mid G \wedge G$ where $\alpha, \beta \in \Sigma \cup \mathcal{X}$. A GVA is a tuple $A = (\Sigma, \mathcal{X}, Q, Q_0, \delta, F, \mu)$, where Σ is an infinite set of letters, $\mathcal{X}$ is a finite set of variables, Q is a finite set states, $Q_0 \subseteq Q$ is a finite set of initial states, $\delta : Q \times (\Sigma_A \cup \mathcal{X}) \times \mathcal{G} \to 2^Q$ is a transition function where $\Sigma_A \subset \Sigma$ is a finite set of alphabet, $F \subseteq Q$ is a finite set of final states and $\mu : \mathcal{X} \to 2^Q$ is called the refreshing function.

In the sequel, we write (q, t, g, q') to denote a transition from q to q', when $q' \in \delta((q, t, g))$. At every point in time, a run of a GVA A is determined by its instantaneous description (or simply, configuration). A configuration of a GVA A is given by a pair $id = (q, \sigma)$ where $q \in Q$ is a state in A and $\sigma : \mathcal{X} \to \Sigma$ is a variable valuation. A valuation σ is extended to be the identity over the elements of Σ. The satisfaction of a guard g by a valuation σ, denoted $\sigma_i \models g$, is defined as usual. A run of A starts at an initial configuration $id_0 = (q_0, \sigma_0)$, with $q_0 \in Q_0$

an initial control state of A and σ_0 an arbitrary valuation of the variables of $\mathcal{X}$[3]. Then A moves from a configuration $id_i = (q_i, \sigma_i)$ to a configuration $id_j = (q_j, \sigma_j)$ over the letter $\sigma_i(t)$ if there is a transition $(q_i, t, g, q_j) \in \delta$ s.t. $\sigma_i \models g$ and $\sigma_i(x) = \sigma_j(x), \forall x \in \mathcal{X} \setminus \mu(q_j)$ (i.e., σ_i and σ_j coincide on the values of the variables that are not refreshed at the state q_j). Hence, given a GVA A, the runs of a A is captured by an infinite state machine called the extended machine of A and denoted $E(A)$. Roughly speaking, $E(A)$ is made of all the configurations of A and all the transitions between these configurations. Simulation between two guarded automata A and B, noted $A \preceq B$, is defined as simulation between their associated state machines, i.e., $A \preceq B$ iff $E(A) \preceq E(B)$.

3 Description logic with variables

Let N_A and N_R be respectively two disjoint and potentially infinite sets of atomic concept names and role names and let $\mathcal{L}$ be a description logic. Let N_C be an infinite set of concept names including the atomic concept names (i.e., $N_A \subseteq N_C$). Concept descriptions specified in the logic $\mathcal{L}$ (or $\mathcal{L}$-descriptions) are built from concept names N_C and role names N_R using the constructors of $\mathcal{L}$. The semantics of $\mathcal{L}$-descriptions is defined as usual. Let $N_{C_\mathcal{T}} \subset N_C$ and $N_{R_\mathcal{T}} \subset N_R$ be respectively two finite sets of concept names and role names. An *$\mathcal{L}$-terminology* $\mathcal{T}$ over the set of concept names $N_{C_\mathcal{T}}$ and the set of role names $N_{R_\mathcal{T}}$ is a set of concept definitions of the form $A \equiv D$, where $A \in N_{C_\mathcal{T}}$ is a concept name and D is an $\mathcal{L}$-description. Atomic concept names (i.e., elements of N_A) are prohibited from appearing in a left-hand side of a definition while the the concept names occurring of $N_{C_\mathcal{T}} \setminus N_A$, called *defined concepts*, must appear at the left-hand side of a definition. We assume that a terminology does not contain multiple definitions and we allow cyclic definitions. In this paper, we study reasoning in the context of a gfp-semantics [3].

Introducing variables. Let N_A, N_R and N_C defined as previously. For a description logic $\mathcal{L}$, we note by $\mathcal{L}_\mathcal{V}$ the corresponding description logic augmented with variables. To define the logic $\mathcal{L}_\mathcal{V}$, we extend the sets of concept names and roles names with variables. Let N_{cv} and N_{rv} be respectively the sets of concept and role variables. The sets N_{cv}, N_{rv} and $N_R \cup N_C$ are pairwise disjoints. In the sequel, we use the letters $X, Y, \ldots$ to denote variables. An $\mathcal{L}_\mathcal{V}$-terminology is defined over the set of concept terms $N_{C_\mathcal{T}} \subset N_C \cup N_{cv}$ and role terms $N_{R_\mathcal{T}} \subset N_R \cup N_{rv}$. Hence, $\mathcal{L}_\mathcal{V}$-patterns (or simply patterns), are built from concept terms $N_{C_\mathcal{T}}$ and role terms $N_{R_\mathcal{T}}$ using the $\mathcal{L}$-constructors. Therefore, an *$\mathcal{L}_\mathcal{V}$-terminology* is a set of pattern definitions of the form $C \equiv D$, where $C \in N_{C_\mathcal{T}} \setminus (N_A \cup N_{cv})$ is a concept name and D is an $\mathcal{L}_\mathcal{V}$-description. We allow indeed cyclic definitions in $\mathcal{L}_\mathcal{V}$-terminologies.

[3] Note that we adopt a slight different vision of configurations than [5] who considers, for example, a unique initial configuration $id_0 = (q_0, \emptyset)$.

Pattern valuation. Concept (respectively, role) variables take their values from the infinite set of atomic concept names (respectively, role names). This is captured by the notion of valuation. A variable valuation is a mapping $\sigma : N_{cv} \cup N_{rv} \rightarrow N_A \cup N_R$ which maps concept variables into atomic concepts and role variables into role names. We denote by $\mathcal{V}al$ the infinite set of all such possible valuations. Valuations are straightforwardly extended to $\mathcal{L}_\mathcal{V}$-patterns by considering that a valuation is the identity on elements of $N_C \cup N_R$. Continuing with the example of section 1, and taking a valuation σ_1 such that $\sigma_1(X) = University$, $\sigma_1(Y) = works\text{-}for$ and $\sigma_1(Z) = University$ we obtain $\sigma_1(B) \equiv B_1$ and $\sigma_1(B_3) \equiv B_1$. If we consider now a valuation σ_2 such that $\sigma_2(X) = \sigma_2(Z) = eSchool$ and $\sigma_2(Y) = evaluates$ we obtain $\sigma_2(B) \equiv B_2$ and $\sigma_2(B_3) \equiv B_4$. Hence, for an $\mathcal{L}_\mathcal{V}$-pattern C, $\sigma(C)$ is an $\mathcal{L}$-description. As a consequence, an $\mathcal{L}_\mathcal{V}$-pattern C describes a (potentially infinite) set of $\mathcal{L}$-descriptions (corresponding to the potentially infinite number of possible valuations). We extend the notion of valuation to $\mathcal{L}_\mathcal{V}$-teminologies as follows: given an $\mathcal{L}_\mathcal{V}$-terminology $\mathcal{T}$ and a variable valuation σ, we denote by $\sigma(\mathcal{T})$ the terminology made of the definitions of the form $A \equiv \sigma(D)$ such that $A \equiv D$ is a pattern definition in $\mathcal{T}$. Therefore, an $\mathcal{L}_\mathcal{V}$-terminology $\mathcal{T}$ specifies an (infinite) set of $\mathcal{L}$-terminologies $\sigma(\mathcal{T}), \forall \sigma \in \mathcal{V}al$.

Variants of variable semantics. Several possibilities exist to define the semantics of variables depending on the restrictions imposed on variable valuations. We borrow the notion of *non-deterministic reassignment* of variables from variable automata semantics [10,8,5], to define in this paper two classes of variable semantics: refreshing vs. non refreshing variable semantics. The demarcation between these two kinds of semantics lies in the valuation of variables that appear in the scope of a terminological cycle. A non refreshing semantics requires to have a unique valuation of such variables while a refreshing semantics enables to assign different values to the same variable for each *unfolding* of a cycle. To make the meaning of each semantics clear, let us consider again the pattern B_5 given at section 1. A one-step unfolding of B_5 leads to the following pattern description: $Person \sqcap \exists works\text{-}for.X' \sqcap graduated\text{-}from.X' \sqcap \exists has\text{-}relative.(Person \sqcap \exists works\text{-}for.X'' \sqcap graduated\text{-}from.X'' \sqcap \exists has\text{-}relative.B_5)$. A non-refreshing semantics allows only valuations σ that satisfy $\sigma(X') = \sigma(X'')$ while a refreshing semantics permits to have different valuations for X' and X'' (i.e., we may have $\sigma(X') \neq \sigma(X'')$). Indeed, in a non-refreshing semantics, the same variable can be used for both X' and X'' and hence the number of variables used in a (unfolded) definition is always finite. The case of refreshing semantics is different since in this case a cyclic pattern refers (implicitly) to an infinite number of variables. To keep the number of variables finite, we allow the possibility to refresh the values of a given variable during the unfolding of a given definition. For example, a valuation of the pattern B_5 using the assignment σ_i leads to the definition: $\sigma_i(B_5) \equiv Person \sqcap \exists works\text{-}for.\sigma_i(X') \sqcap graduated\text{-}from.\sigma_i(X') \sqcap \exists has\text{-}relative.\sigma_{i+1}(B_5)$.

Hence, the valuation of a $\mathcal{L}_\mathcal{V}$-pattern C is provided by a (potentially infinite) sequence of valuations $\sigma_0, \ldots, \sigma_n$, where σ_0 is the initial valuation and where each valuation σ_{i+1} coincides with the valuation σ_i on the non refreshed variables (i.e.,

$\sigma_i(X') = \sigma_{i+1}(X)'$ if X' is not refreshed) while σ_{i+1} assigns new values to the refreshed variables.

We use the following notation to distinguish between these two classes of semantics: r-semantics denotes refreshing variables semantics (valuations $\mathcal{V}al^r$) while nr-semantics denotes non refreshing variables semantics (valuations $\mathcal{V}al^{nr}$). Let $t \in \{r, nr\}$ and C be an $\mathcal{L}_\mathcal{V}$-pattern. We denote by $\mathcal{D}^t(C)$= $\{\sigma(C) \mid \forall \sigma \in \mathcal{V}al^t \}$ the infinite set of $\mathcal{L}$-descriptions obtained from the pattern C by the valuations of $\mathcal{V}al^t$.

Reasoning in description logics with variables. Let $\mathcal{T}$ and $\mathcal{T}'$ two $\mathcal{L}_\mathcal{V}$-terminologies. We say that $\mathcal{T}$ and $\mathcal{T}'$ are *coherent* if $N_{C_\mathcal{T}} \cap N_{C_{\mathcal{T}'}} = \emptyset$.

Let $t \in \{r, nr\}$. Let C and D be two $\mathcal{L}_\mathcal{V}$-patterns of an $\mathcal{L}_\mathcal{V}$-terminology $\mathcal{T}'$ and let E be an $\mathcal{L}$-description of an $\mathcal{L}$-termonilogy $\mathcal{T}''$ coherent with $\mathcal{T}'$. Let $\mathcal{T}$= $\mathcal{T}' \cup \mathcal{T}''$. In this paper we are interested by the following reasonings.

- E is compliant with C w.r.t. a t-semantics, noted $E \leqslant^t_\mathcal{T} C$, iff there exists a valuation $\sigma \in \mathcal{V}al^t$ such that $E \sqsubseteq_{\sigma(\mathcal{T})} \sigma(C)$ (i.e., E is subsumed by $\sigma(C)$ w.r.t. the terminology $\sigma(\mathcal{T})$).
- C is contained in D w.r.t. a t-semantics, noted $C \ll^t_{\mathcal{T}'} D$, iff for every $\mathcal{L}$-description E of an $\mathcal{L}$-termonilogy $\mathcal{T}''$, we have $E \leqslant^t_\mathcal{T} C$ implies $E \leqslant^t_\mathcal{T} D$ (i.e., every description E which is compliant with C w.r.t. a t-semantics is also compiant with D w.r.t. a t-semantics),

4 The case of the logic $\mathcal{EL}_\mathcal{V}$

In this paper, we study reasoning in the context of a gfp-semantics while we believe that our framework can be extended to descriptive and lfp-semantics as well. We are interested by classes of description logics in which reasoning w.r.t. a gfp-semantics can be characterized using a finite state machine (e.g., automata or graphs ([3,12]).

We provide below a characterization of compliance and pattern containment, w.r.t. a gfp-semantics, in $\mathcal{EL}_\mathcal{V}$. We explain briefly how to turn any $\mathcal{EL}_\mathcal{V}$-pattern definition C into a variable automata $\mathcal{A}^t_C$. To achieve this task, we adapt the $\mathcal{EL}$-normal form proposed in [3] to $\mathcal{EL}_\mathcal{V}$-patterns. Let $\mathcal{T}$ be an $\mathcal{EL}_\mathcal{V}$-terminology. An $\mathcal{EL}_\mathcal{V}$-pattern definition $C \equiv D \in \mathcal{T}$ is in a normal form if D is of the form

$$D \equiv V_1 \sqcap \ldots \sqcap V_m \sqcap \exists W_1.D_1 \sqcap \ldots \sqcap \exists W_n.D_n$$

for $m, n \geq 0$ and each V_i is either an atomic concept name or a concept variable (i.e., $V_i \in N_A \cup N_{cv}$, for $i \in [1, m]$), and D_j is a defined concept name (i.e., $D_j \in N_{C_\mathcal{T}} \setminus (N_A \cup N_{cv})$, for $j \in [1, n]$ and each W_i is a role term (i.e., $R_i \in N_{R_\mathcal{T}}$ (R_i). A pattern terminology $\mathcal{T}$ is said *normalized* if all the pattern definitions it contains are in a normal form. W.o.l.g., we assume that two pattern definitions in a normalized terminology use disjoint sets of variables. It is worth noting that, since concept variables do not range over defined concept names, the

normalization process proposed in [3] can be straightforwardly extended to our context to transform any $\mathcal{EL}_{\mathcal{V}}$-terminology into a normalized one. As an example, a normalized terminology containing the defined concepts of our running example is given below:

$B \equiv Person \sqcap \exists works\text{-}for.D_1 \sqcap \exists graduated\text{-}from.D_1$
$B_1 \equiv Person \sqcap \exists works\text{-}for.D_2 \sqcap \exists graduated\text{-}from.D_2$
$B_2 \equiv Person \sqcap \exists works\text{-}for.eSchool \sqcap \exists graduated\text{-}from.eSchool$
$B_4 \equiv Person \sqcap \exists evaluates.D_3 \sqcap \exists graduated\text{-}from.D_3$
$B_5 \equiv Person \sqcap \exists works\text{-}for.D_4 \sqcap \exists graduated\text{-}from.D_4 \sqcap \exists has\text{-}relative.B_5$
$D_1 \equiv X$
$D_2 \equiv University$
$D_3 \equiv eSchool$
$D_4 \equiv X'$

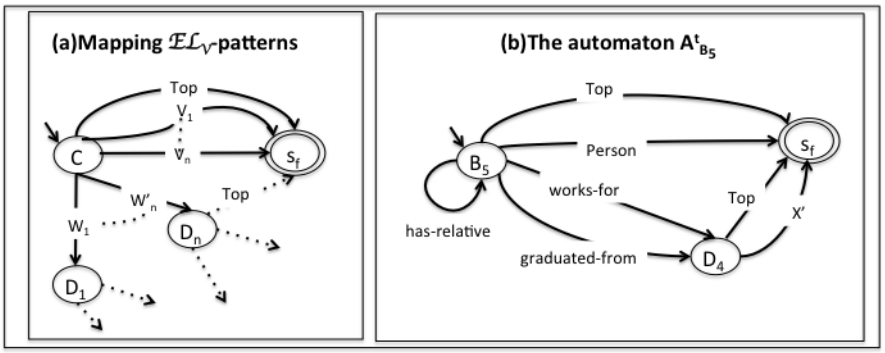

Fig. 1. Mapping $\mathcal{EL}_{\mathcal{V}}$-pattern definitions into variable automata.

Mapping $\mathcal{EL}_{\mathcal{V}}$-patterns into variable automata. Let $\mathcal{T}$ be a normalized $\mathcal{EL}_{\mathcal{V}}$-terminology. We explain below how to map defined concepts and patterns of $\mathcal{T}$ into variable automata. Let $Q_{\mathcal{T}} = N_{C_{\mathcal{T}}} \setminus (N_A \cup N_{cv})$ be a set of states made of the defined concepts of $\mathcal{T}$. Each concept definition or $\mathcal{EL}_{\mathcal{V}}$-pattern $C \equiv V_1 \sqcap \ldots \sqcap V_m \sqcap \exists W_1.D_1 \sqcap \ldots \sqcap \exists W_n.D_n$ of $\mathcal{T}$ is turned into a variable automaton $A^t_C = (\Sigma_C, \mathcal{X}_C, Q_C, Q_0, \delta, F, \mu)$ defined as follows:

- the alphabet $\Sigma_C \subseteq N_A \cup N_R \cup \{Top\}$, is made of a subset of atomic concept names, role names and the Top concept,
- the set of variables $\mathcal{X}_C \subseteq N_{cv} \cup N_{rv}$, is made of a subset of concept and role variables,
- the set of states $Q_C \subseteq Q_{\mathcal{T}} \cup \{C, S_f\}$, is made of the states of $Q_{\mathcal{T}}$ reacheable from the state C,
- $Q_0 = \{C\}$ is the set of initial states of the automaton and $F = \{S_f\}$ is its set of final states,

– the transitions are unguarded (i.e., $\mathcal{G} = \emptyset$). The transition function δ is defined as follows:
  - $(q, Top, true, S_f) \in \delta$, $\forall q \in Q_C$, i.e., there is an edge labeled Top from every node q in Q_C to the final state S_f,
  - $(C, V_i, true, S_f) \in \delta$, $\forall i \in [1, m]$, i.e., each term V_i is turned into an edge, labeled V_i, from the node C to the final state S_f.
  - $(C, W_i, true, D_i) \in \delta$, $\forall i \in [1, n]$, i.e., each term $\exists W_i.D_i$ is turned into an edge, labeled W_i, from the node C to the node D_i.
– the refreshing function μ is defined as follows:
  - if $t = r$ then $\mu(x) = Q_C, \forall x \in \mathcal{X}$
  - if $t = nr$ then $\mu(x) = \emptyset, \forall x \in \mathcal{X}$

The proposed mapping of a pattern $C \equiv V_1 \sqcap \ldots \sqcap V_m \sqcap \exists W_1.D_1 \sqcap \ldots \sqcap \exists W_n.D_n$ into an automaton $A^t_C = (\Sigma_C, \mathcal{X}_C, Q_C, Q_0, \delta, F, \mu)$ is depicted at figure 1(a) while the figure 1(b) shows the variable automaton $A^t_{B_5}$ corresponding to the pattern B_5 of section 1. Figure 2(a) shows $E(A^r_{B_5})$, the extended automaton of $A^r_{B_5}$, for the case of a variable refreshing semantics (i.e. $t = r$). Note that,

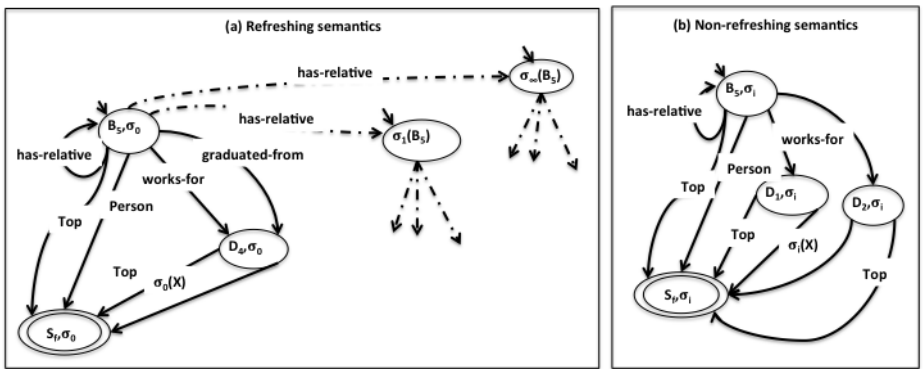

Fig. 2. The extended automaton $E(A^t_{B_5})$.

this machine includes an infinite set of initial states (i.e., the configurations (B_5, σ_i), $\forall \sigma_i \in \mathcal{V}al^r$). In the case of a refreshing semantics, the cyclic role has-relative relates a given state (B_5, σ_i) to all the other possible states (B_5, σ_j), $\forall \sigma_j \in \mathcal{V}al^r$, thereby making each machine rooted at an initial configuration (B_5, σ_i) an infinite state machine. In the case of a non-refreshing semantics (i.e. $t = nr$), $E(A^{nr}_{B_5})$ is made of an infinite set of, pairwise disconnected, finite state machines rooted at (B_5, σ_i), $\forall \sigma_i \in \mathcal{V}al^{nr}$ (c.f., figure 2(b)). The following lemma establishes a connection between subsumption in $\mathcal{EL}_\mathcal{V}$ and simulation between instances of variable automata.

Lemma 1. *Let $t \in \{r, nr\}$ and let C and D be two $\mathcal{EL}_\mathcal{V}$-patterns in a $\mathcal{EL}_\mathcal{V}$-terminology $\mathcal{T}$. Let $\sigma \subseteq \mathcal{V}al^t$, then $\sigma(C) \sqsubseteq_{\sigma(\mathcal{T})} \sigma(D)$ iff $\sigma(A^t_D) \preceq \sigma(A^t_C)$.*

Let $\mathcal{T}$ be an $\mathcal{EL}$-terminology $\mathcal{T}$. This lemma is derived from the observation that the set of all the automata $E(A^t_C)$, of the patterns C of $\mathcal{T}$, corresponds to a slight modification of the notion of $\mathcal{EL}$-description graph [3] of the $\mathcal{EL}$-terminology $\sigma(\mathcal{T})$, where: *(i)* a set of final states (S_f, σ) with $\sigma \in \mathcal{V}al^t$, marked as a final states, is added to the graph, and *(ii)* the atomic concepts are turned into edges from the nodes representing defined concepts to the final states (S_f, σ). It is clear that such a transformation of a description graph of a terminology $\mathcal{T}$ preserves the simulation relation between the nodes of the initial graph. Hence, the characterization of subsumption w.r.t. gfp-semantics using the simulation relation is still valid in our context.

Therefore reasoning over C can be reduced to reasoning over the corresponding variable automata A^t_C. However, despite the correspondance established by lemma 1, reduction of compliance and containment to simulation between variable automata is not straightforward and requires additional transformations. Consider for example the following pattern definitions in a terminology $\mathcal{T}= \{E \equiv \exists r_1.D_3, A \equiv \exists X.D_1 \sqcap \exists Y.D_2, D_i \equiv Top,\ \text{for } i \in \{1,2,3\}\}$. The corresponding automata A^t_E and A^t_A are depicted at figure 3(a). It is easy to check that E is compliant with A w.r.t. any t-semantics. Indeed, for any valuation σ which satisfy $\sigma(X) = \sigma(Y) = r_1$, we have $\sigma(A^t_A) \preceq A^t_E$ and hence $E \sqsubseteq_{\sigma(\mathcal{T})} \sigma(A)$ (which implies that E is compliant with A). However, we have clearly $A^t_A \not\preceq A^t_E$. As a witness of non simulation take any valuation σ' which satisfy $\sigma'(X) \neq r_1$ or $\sigma'(Y) \neq r_1$, and in this case we have $A^t_{\sigma'(A)} \not\preceq A^t_{\sigma'(E)}$ (which implies that $A^t_A \not\preceq A^t_E$). Despite this fact, it is still possible to reduce compliance to simulation after a transformation of the automata A^t_A and A^t_E into two new automata, denoted $\bar{A}^t_A$ and $\bar{A}^t_E$ (c.f., figure 3(b)). The main intuition underlying the proposed transformation is to construct new automata that satisfy the following property: $\bar{A}^t_A \preceq \bar{A}^t_E$ iff $\exists \sigma \in \mathcal{V}al^t$ s.t. $\sigma(A^t_A) \preceq A^t_E$. In the similar way, additional transformations are needed to reduce containment to simulation as stated in the following theorem.

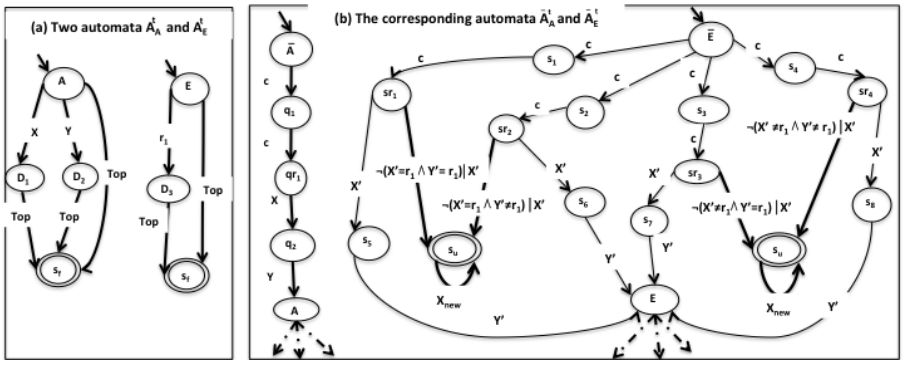

Fig. 3. Reducing compliance to simulation.

Theorem 1. *Let $\mathcal{T}'$ be an $\mathcal{EL}_{\mathcal{V}}$-terminology and C and D two $\mathcal{EL}_{\mathcal{V}}$-patterns in $\mathcal{T}$ and let E be a definition in and $\mathcal{EL}$-terminology $\mathcal{T}''$ coherent with $\mathcal{T}'$. Let $\mathcal{T} = \mathcal{T}' \cup \mathcal{T}''$. Then there exists guarded variable automata $\hat{A}^t_C, \tilde{A}^t_D, \bar{A}^t_E$ and $\bar{A}^t_C$ such that:*

(i) $E \leqslant^t_{\mathcal{T}} C$ *iff* $\bar{A}^t_C \preceq \bar{A}^t_E$, *and*
(ii) $C \ll^t_{\mathcal{T}} D$ *iff* $\tilde{A}^t_D \preceq \hat{A}^t_C$.

The proof of this theorem is given in the extended version of this paper [13]. We explain below the case (i) (compliance). $E \leqslant^t_{\mathcal{T}} C$ iff $\bar{A}^t_C \preceq \bar{A}^t_E$. Figure 3(b) illustrates the construction of the automata $\bar{A}^t_A$ and $\bar{A}^t_E$ corresponding to the automata of A^t_A and A^t_E of figure 3(b). The main idea is to prefix the automata A^t_A and A^t_E with a set of transitions that enable to ensure that: $\bar{A}^t_A \preceq \bar{A}^t_E$ iff $\exists \sigma \in \mathcal{V}al^t$ s.t. $\sigma(A^t_A) \preceq A^t_E$. The new sr_i and qr_i states correspond to states where all the variables are refreshed. Simulation can be viewed as a game between a player P_E, which moves in the automaton $\bar{A}^t_E$ with an assigned goal to prove simulation, and a player P_A, which moves in the automaton $\bar{A}^t_A$ with an assigned goal to prove that there is no simulation. The rational behind the proposed construction is to enable P_E to force a choice of a valuation σ_t that satisfy simulation if such a valuation exists. This is achieved by the two first transitions labeled with the constant symbol c. At the beginning of the game, P_A choose an arbitrary valuation σ_0 and moves from the state $\bar{A}$ to q_1. The player P_E have a choice between several transitions c, that goes in the example from state $\bar{E}$ to one of the states $s_1, \ldots, s_4$. By this deterministic choice, P_E have the possibility to choose between several classes of valuations, each class being defined w.r.t. to the relation of the variables of the automata A^t_A with the constants that appear in A^t_E. Once such a choice is performed, the player P_A moves upon the constant c to the state qr_1 where he has the possibility to refresh the variables in order to comply with the class of valuation selected by P_E. The rest of the prefix construction enables the player P_E to synchronize his own variables with the variables of P_A in order to be aware of the valuation chosen by P_A. These variables are then used in the guards that enable P_E to enter a *'universal'* final state s_u in the cases where the player P_A cheats (i.e., at the state qr_1, the player P_A picks a valuation that do not belong to the class selected by P_E). Hence, $\bar{A}^t_A \preceq \bar{A}^t_E$ iff there is a valuation σ_t (that belongs to the class of valuations picked by P_E) such that $\sigma_t(A^t_A) \preceq A^t_E$.

Based on the previous theorem, we can provide the following result w.r.t. the considered reasoning in $\mathcal{EL}_{\mathcal{V}}$.

Theorem 2. *Let $t \in \{r, nr\}$. Then:*

(i) Compliance in $\mathcal{EL}_{\mathcal{V}}$ w.r.t. a t-semantics is NP-complete,
(ii) Pattern containment in $\mathcal{EL}_{\mathcal{V}}$ w.r.t.a t-semantics is in 2-ExpTime.

Consider first the case (i). We recall that, E is compliant with C w.r.t. a t-semantics, noted $E \leqslant^t_{\mathcal{T}} C$, iff there exists a valuation $\sigma \in \mathcal{V}al^t$ such that $E \sqsubseteq_{\sigma(\mathcal{T})} \sigma(C)$. We know from our construction that $E \leqslant^t_{\mathcal{T}} C$ is equivalent to: $\exists \sigma$ s.t $\sigma(A^t_C) \preceq A^t_E$.

The *Compliance* problem is in *NP*: given an oracle which guess a valuation σ, then it is possible to check in polynomial time if $\sigma(A^t_C) \preceq A^t_E$.

The *NP-hardness* is proved by a reduction from graph 3-colorability problem [11]. Let $G = (V, E)$ be a graph. G is 3-colorable if it is possible to assign to each node in V one of the three colors in such a way that every two nodes connected by an edge have different colors. Starting from G and the three colors $\{r, g, b\}$, we construct a *compliance* problem such that the graph G is 3-colorable iff $E_{color} \leqslant^t_{\mathcal{T}} C_G$. Where E_{color} is a definition in $\mathcal{T}'$ and C_G a $\mathcal{EL}_{\mathcal{V}}$-pattern definition in $\mathcal{T}''$ and $\mathcal{T} = \mathcal{T}' \cup \mathcal{T}''$.

Construction of the terminology $\mathcal{T}'$. For each color $c \in \{b, r, g\}$, we include in $\mathcal{T}'$ a concept definition E_c as follows:

- $E_b \equiv \exists b.E_{Top}$
- $E_r \equiv \exists r.E_{Top}$
- $E_g \equiv \exists g.E_{Top}$
- $E_{Top} \equiv Top$

In addition, $\mathcal{T}'$ contains the definition $E_{color} \equiv \exists r.E_b \sqcap \exists g.E_b \sqcap \exists r.E_g \sqcap \exists b.E_g \sqcap \exists b.E_r \sqcap \exists g.E_r$.

Construction of the $\mathcal{EL}_{\mathcal{V}}$-terminology $\mathcal{T}''$. Given a graph $G = (V, E)$ with $V = \{n_1, \ldots, n_m\}$, we include in $\mathcal{T}''$ the following set of $\mathcal{EL}_{\mathcal{V}}$-patterns:

- $\{N_i \equiv \exists X_i.N_{Top} \mid for\ i \in [1, m]\}$ and
- $N_{Top} \equiv Top$

In addition, $\mathcal{T}''$ includes the pattern: $C_G \equiv \bigsqcap_{(n_i,n_j)\in E} \exists X_i.N_j$.

Let $\mathcal{T} = \mathcal{T}' \cup \mathcal{T}''$. It is then easy to check that G is 3-colorable iff $E_{color} \leqslant^t_{\mathcal{T}} C_G$.

Complexity of pattern containment (case (ii)) is obtained from theorem 1 which reduces an $\mathcal{EL}_{\mathcal{V}}$-pattern containment test $C \ll^t_{\mathcal{T}} D$ into a simulation test $\tilde{A}^t_D \preceq \hat{A}^t_C$ between two GVA. The automaton $\hat{A}^t_C$ is exponential in the size of C and D. Hence, knowing from [5] that simulation is in EXPTIME, we obtain an immediate 2-EXPTIME upper bound for the $\mathcal{EL}_{\mathcal{V}}$-pattern containment problem.

5 Conclusion

This paper addresses the problems of pattern compliance and containment for description logics with variables. It considers a framework that cater for cyclic terminologies and defines two semantics of variables which differ w.r.t. to the possibility or not to refresh the variables. The paper provides preliminary results regarding the description logic $\mathcal{EL}_{\mathcal{V}}$, obtained from an extension of $\mathcal{EL}$ with variables. Future research work will be devoted to the extension of the approach to more expressive logics.

References

1. F. Baader, R. Küsters, A. Borgida, and D. McGuinness. Matching in description logics. *Journal of Logic and Computation*, 9(3):411–447, 1999.
2. F. Baader and P. Narendran. Unification of concept terms in description logics. In *ECAI'98*, pages 331–335, 1998.
3. Franz Baader. Terminological cycles in a description logic with existential restrictions. In *IJCAI*, pages 325–330, 2003.
4. Franz Baader, Stefan Borgwardt, and Barbara Morawska. Extending unification in $\mathcal{EL}$ towards general TBoxes. In *KR'12*, pages 568–572, 2012.
5. Walid Belkhir, Yannick Chevalier, and Michael Rusinowitch. Guarded Variable Automata over Infinite Alphabets. October 2013.
6. F. Baader and R. Küsters. Matching in description logics with existential restrictions. In *DL'99*, Sweden, 1999.
7. Georg Gottlob, Giorgio Orsi, and Andreas Pieris. Ontological queries: Rewriting and optimization (extended version). *CoRR*, abs/1112.0343, 2011.
8. Orna Grumberg, Orna Kupferman, and Sarai Sheinvald. Variable automata over infinite alphabets. In *LATA*, pages 561–572, 2010.
9. J.E. Hopcroft and J.D. Ullman. Formal languages and their relation to automata. *ACM Classic Books Series*, 1969.
10. Michael Kaminski and Daniel Zeitlin. Finite-memory automata with nondeterministic reassignment. *Int. J. Found. Comput. Sci.*, 21(5):741–760, 2010.
11. R.M. Karp. Reducibility among combinatorial problems. In R.E. Miller and J.W. Thatcher, editors, *Complexity of Computer Computations*. Plenum Press, 1972.
12. R. Küsters. Characterizing the Semantics of Terminological Cycles in $\mathcal{ALN}$ using Finite Automata. In *KR'98*, pages 499–510, 1998.
13. L.Akroun, L. Nourine, and F. Toumani. Reasoning in description logics with variables: preliminary results (extended version). Technical report, Blaise Pascal University, http://www.isima.fr/ftoumani/dlvextended2015.pdf, 2015.