Tractable Feature Generation through Description Logics with Value and Number Restrictions

Nicola Fanizzi, Luigi Iannone, Nicola Di Mauro, and Floriana Esposito

Dipartimento di Informatica, Università degli Studi di Bari
Campus Universitario, Via Orabona 4, 70125 Bari, Italy
{fanizzi|iannone|ndm|esposito}@di.uniba.it

Abstract. In the line of a feature generation paradigm based on relational concept descriptions, we extend the applicability to other languages of the Description Logics family endowed with specific language constructors that do not have a counterpart in the standard relational representations, such as clausal logics. We show that the adoption of an enhanced language does not increase the complexity of feature generation, since the process is still tractable. Moreover this can be considered as a formalization for future employment of even more expressive languages from the Description Logics family.

1 Introduction

Many interesting tasks in AI such as natural language processing, computer vision, and planning require adequate relational representations. Examples include the problem of identifying relations of interest, identifying a speaker given the conference schedule, answering to free-form questions given relevant text, detecting people in an image or defining a policy for planning. Hence, multi-relational representations must be taken into account. The challenge is to provide the necessary expressivity, meeting the strong tractability constraints posed by such tasks.

The problem with learning unbiased relational representations is their intractability [1] which has led to the investigation of ways to impose some bias in order to make deductive and inductive reasoning more efficient. In order to overcome the inherent intractability, an increasing interest lately has been devoted to approaches based on *propositionalization* [2], whose final aim is to exploit the efficiency of propositional learning for inducing relational classifiers.

This work follows the approach to propositionalization that employs concept representations for abstracting relational structures by generating new relevant features through an efficient *generating function* (see [3]). Rather than during the learning process, features are intended to be generated before learning takes place.

Various relational representations which may serve as a starting point for feature construction, such as *concept graphs* and *frame systems*, have been unified in the framework of *Description Logics* (henceforth DLs) [4]. They are well suited for this purpose, indeed DLs descriptions are employed in KRR as a means for expressing concepts (as classes of individuals) and their properties. Besides, this family of languages is endowed with well-founded semantics and reasoning services (mostly for deductive inference) descending from a long research line.

Differently from other techniques for propositionalization, in the proposed approach DLs representations constitute intermediate (rich) languages for transforming domain elements into new features expressed in a new lexicon based on DLs through *feature generating functions* [3]. The results of this transformation may then be piped as the input for general-purpose propositional learners especially those that can handle examples with a variable number of (relevant) features which is much less than the total number of (possibly infinite) features [5] for inducing structures that are described like functions mapping propositional variables to DLs descriptions.

In this paper we aim at generalizing this approach in terms of more expressive DLs. Differently from the mentioned work introducing DLs in the paradigm [3], where, for the sake of tractability, a very simple DL language, the *Feature Description Logic* (FDL) is employed. Actually FDL is a very simple language supporting only existential attributes and conjunction which are well suited for existential descriptions like in an ILP context. Yet, with the advent of the Semantic Web it is likely that many other KBs expressed in DLs will be made available for interoperation.

However existential representations are not always well suited. Indeed, consider features as a sort of types defined like in frame-based systems, E-R models and object-oriented models [6], they have to be regarded as constructed on different constructors, namely those based on universal restrictions. Therefore, we extend the original feature construction framework towards different DLs which are still endowed with efficient reasoning services requested by the paradigm. Particularly, we propose a method based upon the $\mathcal{ALN}$ logic [7, 8, 4] thus yielding more expressiveness for the relational descriptions that are abstracted by the elicited features. These features, in turn, can be acquired to enrich the starting knowledge base building up a new representation.

The original algorithm for feature extraction produces only active features acting as positive examples for the adopted propositional learners, thus making a sort of Closed World Assumption, which contrasts with the mainstream in DLs reasoning: an inactive feature should be explicitly inferred from the knowledge base. By allowing negation in the language, it becomes natural to represent also negative examples.

The paper is organized as follows. In Sect. 2 the representation language is presented. The learning framework is illustrated in Sect. 3 and it is discussed in Sect. 4. Possible developments of the method are examined in Sect. 5.

2 The $\mathcal{ALN}$ Description Logic

$\mathcal{ALN}$ is a DLs language which allows for the expression of universal features and numeric constraints [4]. It has been adopted because of the tractability of the related reasoning services [9]. In order to keep this paper self-contained, syntax and semantics for the reference representation is briefly recalled with the characterization of the descriptions in terms of concept graphs.

In DLs, primitive *concepts* $N_C = \{A, \ldots\}$ are interpreted as subsets of a certain domain of objects and primitive *roles* $N_R = \{R, S, \ldots\}$ are interpreted as binary relations on such a domain. In $\mathcal{ALN}$, more complex concept descriptions are built using atomic concepts and primitive roles by means of the constructors presented in Table 1. Their meaning is defined by an *interpretation* $I = (\Delta^I, \cdot^I)$, where Δ^I is the *domain* of the

Table 1. Constructors and related interpretations for $\mathcal{ALN}$.

NAME	INTENSION	EXTENSION
top concept	$\top$	Δ
bottom concept	$\bot$	$\emptyset$
primitive concept	A	$A^I \subseteq \Delta$
primitive negation	$\neg A$	$\Delta \setminus A^I$
concept conjunction	$C_1 \sqcap C_2$	$C_1^I \cap C_2^I$
value restriction	$\forall R.C$	$\{x \in \Delta \mid \forall y\ (x,y) \in R^I \rightarrow y \in C^I\}$
at-most restriction	$\leq n.R$	$\{x \in \Delta \mid \ \lvert\{y \in \Delta \mid (x,y) \in R^I\}\rvert \leq n\}$
at-least restriction	$\geq n.R$	$\{x \in \Delta \mid \ \lvert\{y \in \Delta \mid (x,y) \in R^I\}\rvert \geq n\}$

interpretation and the functor $\cdot^I$ stands for the *interpretation function* mapping the intension of concept and role descriptions to their extension.

A *knowledge base* $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ contains two components: a T-box $\mathcal{T}$ and an A-box $\mathcal{A}$. $\mathcal{T}$ is a set of concept definitions $C \equiv D$, meaning $C^I = D^I$, where C is the concept name and D is a description given in terms of the language constructors. Differently from ILP, each (non primitive) concept has a single definition. Moreover, the DLs definitions are assumed not to be recursive, i.e. concepts cannot be defined in terms of themselves.

The A-box $\mathcal{A}$ contains extensional assertions on concepts and roles, e.g. $C(a)$ and $R(a,b)$, meaning, respectively, that $a^I \in C^I$ and $(a^I, b^I) \in R^I$. Note that, differently from the examples in the ILP setting, the concept description C can be more complex than LP facts. For instance they could assert a universal property of the an individual: $(\forall R.(A \sqcap \neg B))(a)$ that is, role R relates a exclusively to individuals[1] that are instances of the concept $A \sqcap \neg B$.

Example 2.1. Examples of $\mathcal{ALN}$ descriptions are [2]:
$\mathsf{Polygamist} \equiv \mathsf{Person} \sqcap \forall \mathsf{isMarriedTo.Person} \sqcap \geq 2.\mathsf{isMarriedTo}$
$\mathsf{Bigamist} \equiv \mathsf{Person} \sqcap \forall \mathsf{isMarriedTo.Person} \sqcap = 2.\mathsf{isMarriedTo}$
$\mathsf{MalePolygamist} \equiv \mathsf{Male} \sqcap \mathsf{Person} \sqcap \forall \mathsf{isMarriedTo.Person} \sqcap \geq 2.\mathsf{isMarriedTo}$

The notion of *subsumption* between DLs concept descriptions can be given in terms of the interpretations defined above:

Definition 2.1 (subsumption). *Given two concept descriptions C and D, C* subsumes *D iff it holds that $C^I \supseteq D^I$ for every interpretation I. This is denoted denoted by $C \sqsupseteq D$. The induced equivalence relationship, denoted $C \equiv D$, amounts to $C \sqsupseteq D$ and $D \sqsupseteq C$.*

Note that this notion is merely semantic and independent of the particular DLs language adopted. It is easy to see that this definition also applies to the case of role descriptions.

The most important difference between DLs and clausal logics arises. Indeed, while in the context of DLs reasoning the *Open World Assumption* (OWA) is adopted, in ILP the *Closed World Assumption* (CWA) is generally required.

[1] It holds even in case no such $R-$filler is given.
[2] Here $(= n.R)$ is an abbreviation for $(\leq n.R \sqcap \geq n.R)$.

Example 2.2. Considering again the concepts described in Ex. 2.1, the assertions:
$\mathcal{A} = \{\ \mathsf{Person}(\mathsf{Bob}), \mathsf{Person}(\mathsf{Meg}), \mathsf{Person}(\mathsf{Pam}), \mathsf{Male}(\mathsf{Bob}), \neg\mathsf{Male}(\mathsf{Meg}),$
$\neg\mathsf{Male}(\mathsf{Pam}), \mathsf{isMarriedTo}(\mathsf{Bob}, \mathsf{Meg}), \mathsf{isMarriedTo}(\mathsf{Bob}, \mathsf{Pam})\ \}$
would entail that Bob is an instance of Polygamist if the CWA is adopted; otherwise also $\forall \mathsf{isMarriedTo}.\mathsf{Person}(\mathsf{Bob})$ should be known for Polygamist(Bob) to hold.

Semantically equivalent (yet syntactically different) descriptions can be given for the same concept. However they can be reduced to a canonical form by means of equivalence-preserving rewriting rules, e.g. $\forall R.C_1 \sqcap \forall R.C_2 \equiv \forall R.(C_1 \sqcap C_2)$ (see [7, 4]). The normal form employs the notation needed to access the different parts (*sub-descriptions*) of a concept description C:

- $\mathsf{prim}(C)$ denotes the set of all (negated) concept names occurring at the top level of the description C;
- $\mathsf{val}_R(C)$ denotes conjunction of concepts $C_1 \sqcap \cdots \sqcap C_n$ in the value restriction of role R, if any (otherwise $\mathsf{val}_R(C) = \top$);
- $\mathsf{min}_R(C) = \max\{n \in \mathbb{N} \mid C \sqsubseteq (\geq n.R)\}$ (always a finite number);
- $\mathsf{max}_R(C) = \min\{n \in \mathbb{N} \mid C \sqsubseteq (\leq n.R)\}$ (if unlimited then $\mathsf{max}_R(C) = \infty$).

Definition 2.2 ($\mathcal{ALN}$ normal form). *A concept description C is in $\mathcal{ALN}$ normal form iff $C = \top$ or $C = \bot$ or*

$$C = \prod_{P \in \mathsf{prim}(C)} P \sqcap \prod_{R \in N_R} (\forall R.C_R \sqcap \geq n.R \sqcap \leq m.R)$$

where $C_R = \mathsf{val}_R(C)$, $n = \mathsf{min}_R(C)$ and $m = \mathsf{max}_R(C)$.

The complexity of normalization is polynomial [4]. Besides, subsumption can be checked in polynomial time too [9]. Note also that we are considering the case of subsumption with respect to empty terminologies that suffices for our purposes. Otherwise deciding this relationship may be computationally more expensive.

Although subsumption between concept descriptions is merely a semantic relationship, a more syntactic relationship can be found for a language of moderate complexity like $\mathcal{ALN}$ that allows for a structural characterization of subsumption [10].

Proposition 2.1 (subsumption in $\mathcal{ALN}$). *Given two $\mathcal{ALN}$ concept descriptions C and D in normal form, it holds that $C \sqsupseteq D$ iff all the following relations hold between the sub-descriptions:*

- $\mathsf{prim}(C) \subseteq \mathsf{prim}(D)$
- $\forall R \in N_R\colon\ \mathsf{val}_R(C) \sqsupseteq \mathsf{val}_R(D)$
- $\mathsf{min}_R(C) \leq \mathsf{min}_R(D) \wedge \mathsf{max}_R(C) \geq \mathsf{max}_R(D)$

Hence subsumption checking is accordingly polynomial like $O(n \log n)$, where n is the size of concept C. In the following we will refer to concepts descriptions in normal form unless a different case is explicitly stated.

The tree-structured representation of concept description are defined as follows [7]:

Definition 2.3 (description tree). *A* description tree *for a concept C in $\mathcal{ALN}$ normal form is a tree $\mathcal{G}(C) = (V, E, v_0, l)$ with root v_0 where:*

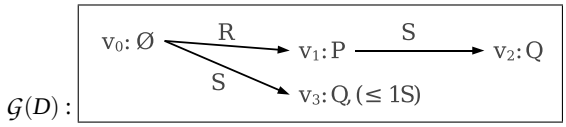

Fig. 1. The concept $D \equiv \forall R.(P \sqcap \forall S.Q) \sqcap \forall S.(Q \sqcap \leq 1S)$ as a description tree.

- *each node $v \in V$ is labelled with a finite set $l(v) \subseteq N_C \cup \{\neg A \mid A \in N_C\} \cup \{\geq n.R \mid n \in \mathbb{N},\ R \in N_R\} \cup \{\leq n.R \mid n \in \mathbb{N},\ R \in N_R\}$*
- *each edge in E is labelled with $\forall R$, where $R \in N_R$*

Proposition 2.2 (equivalence). *An $\mathcal{ALN}$ description C is semantically equivalent to an $\mathcal{ALN}$ description tree $\mathcal{G}(C)$ of size polynomial in the size of C, which can be constructed in polynomial time.*

Proof. Let C be a concept description C in $\mathcal{ALN}$ normal form. It corresponds to the tree $\mathcal{G}(C) = (V, E, v_0, l)$ defined recursively on the depth d of nested restrictions in C:

$(d = 0)$ $\mathcal{G}(C) = (\{v_0\}, \emptyset, v_0, l)$ with $l(v_0) = \mathsf{prim}(C) \cup \bigcup_{R \in N_R} (\geq n.R \sqcap \leq m.R)$

$(d > 0)$ let $N_R(C)$ the names of roles at the top level of C. For any $R \in N_R(C)$, let $\mathcal{G}(C_R) = (V_R, E_R, v_{0R}, l_R)$ be the description tree of $C_R = \mathsf{val}_R(C)$, where w.l.o.g. the V_R's are pairwise disjoint and $v_0 \notin \bigcup_{R \in N_R} V_R$. Then:

- $V = \{v_0\} \cup \bigcup_{R \in N_R} V_R$
- $E = \{v_0 R v_{0R}\} \cup \bigcup_{R \in N_R} E_R$
- if $v = v_0$ then $l(v) = \mathsf{prim}(C) \cup \bigcup_{R \in N_R} (\geq n.R \sqcap \leq m.R)$
  otherwise $l(v) = l_R(v)$ (being $v \in V_R$)

Example 2.3. The concept description $D \equiv \forall R.(P \sqcap \forall S.Q) \sqcap \forall S.(Q \sqcap \leq 1S)$ is equivalent to the tree depicted in Fig. 1.

Instance checking can be characterized in terms of homomorphisms between trees and graphs [7]:

Definition 2.4 (A-box description graph). *Let $\mathcal{A}$ be an $\mathcal{ALN}$ A-box, a be an individual occurring in $\mathcal{A}$ ($a \in \mathsf{Ind}(\mathcal{A})$) and $C_a = \bigsqcap_{C(a) \in \mathcal{A}} C$. Let $\mathcal{G}(C_a) = (V_a, E_a, a, l)$ denote the description tree of C_a. $\mathcal{G}(\mathcal{A}) = (V, E, l)$ is a* A-box description graph *with:*

- $V = \bigcup_{a \in \mathsf{Ind}(\mathcal{A})} V_a$
- $E = \{aRb \mid R(a,b) \in \mathcal{A}\} \cup \bigcup_{a \in \mathsf{Ind}(\mathcal{A})} E_a$
- $l(v) = l_a(v)$ *for all* $v \in V_a$

Subsumption and instance checking can be used to translate an individual of the domain (an instance of the target concept) into a set of features suitable for propositional algorithms. Thus DLs that allow for efficient subsumption procedures, such as $\mathcal{ALN}$, are to be preferred.

3 Feature Generation Based on $\mathcal{ALN}$ Descriptions

Now we recall the main issues of the paradigm developed in [3] adapted to a more generic DLs context. The main point is the feature extraction method that can be used to generate propositional formulae in terms of expressive features through subsumption queries from arbitrarily complex data represented by concept graphs.

Assertions in an A-box are described in terms of the relational language (primitive concept and role names). The aim is then producing a classifier that predicts new assertions to hold for selected elements. The adopted propositional learning algorithm requires examples to be generated on the basis of the available relational assertions in the A-box. These examples make up a set of *active* propositions (features) holding for the target concepts. The propositions can be thought of as ground assertions described in terms of the adopted DLs language.

Since the negation of primitive concepts is also allowed in this DLs language, it is possible to represent explicitly also negative information of instances of the target concept, something that was not possible in the original feature extraction paradigm [3]. This method actually works with the adoption of the CWA which is something unusual for dealing with DLs representations. The method can generate a large number of features belonging to a limited number of types represented by DLs descriptions. Thus the propositional learning algorithm to be employed has to be able to deal with this kind of situations.

Definition 3.1 (feature). *Given a concept description D, a feature F_D is a function $F_D : \mathcal{W} \longmapsto \{0,1\}$ mapping interpretations to truth values. The feature F_D is said to be* active *in an interpretation $I \in \mathcal{W}$ when it evaluates to* 1.

In the following, we will employ the canonic interpretation $I_{\mathcal{A}}$ of an A-Box $\mathcal{A}$, where the set of individuals stand for themselves.

Example 3.1. The description $D \equiv \forall \mathsf{isMarriedTo}.\mathsf{Person}$ is active for the canonic interpretation $I_{\mathcal{A}}$ of the A-box presented in Ex. 2.1.

Active features may be employed as the input for efficient propositional learning systems, such as SNoW [11]. In a *learning from entailment* setting, the central point is the definition of efficient functions that are able to translate interpretations into sets of features [3], thus expressing relational qualities of the individuals that stand as instances of the target concepts.

Definition 3.2 (feature generating function). *Let I be a model for an A-box $\mathcal{A}$ and let $\mathcal{D}$ be a set of descriptions. A* feature generating function (FGF)*, denoted with χ, determines sets of features as follows: $\chi(I,\mathcal{D}) = \{F_D \mid D \in \mathcal{D},\ F_D(I) = 1\}$.*

Thus, the FGF χ performs a change of representation for I into the (subsumers of) description D. Now an interpretation can be regarded as a description graph in which each element involved is in the extension of some node. To restrict the range of possible interpretations, a particular one $I_{\mathcal{A}}$ can be regarded as the canonical model related to the A-box $\mathcal{A}$ [10] where each individual name stands for itself. This interpretation, in turn, can be represented as the very description graph $\mathcal{G}(\mathcal{A})$ (by Def. 2.4).

```
function χ_msc(I, D): Features
input:  I: interpretation,
        D: ALN description
output: Features: feature set

begin
Features ←
for each a ∈ Ind(A) do
        begin
        M_a ← msc^k_A(a)
        for each D ∈ D do
                if D ⊒ M_a then
                        Features ← Features ∧ F_D
        end
return Features
end
```

Fig. 2. A simple FGF algorithm for $\mathcal{ALN}$.

A non-standard inference service for DLs computes the *most specific concept* of an individual a with respect to an A-box $\mathcal{A}$, denoted $msc_{\mathcal{A}}(a)$, which is the most specific concept description (with respect to subsumption) whose extension contains a with respect to all of the models for $\mathcal{A}$ [8]. The generation of the features based on the *msc* may be performed through the simple algorithm reported in Fig. 2. For each individual, the *msc* with respect to the A-box must be computed[3]. This is similar to the most specific subsumer (*mss*) employed in [3], where the language admitted ground (and partially ground) descriptions. We preferred the *msc* since it is well-known and investigated in the KRR community and can be performed through *instance checking* which is supported by existing reasoners.

Example 3.2. Let a generating description be:
$D \equiv \mathsf{Male} \sqcap \leq 2.\mathsf{isMarriedTo} \sqcap \forall \mathsf{isMarriedTo}.\neg\mathsf{Male}.$
In the canonical interpretation of the A-box in Ex 2.1, the feature F_D is active. This holds for all the assertions subsumed by D, such as $msc(\mathsf{Bob})$.

This inference is not possible in all the DLs. When the A-boxes are cyclic only approximations of the *msc* can be computed. The problem arises when the DLs language is endowed with existential restrictions (such as FDL) or number restrictions (like $\mathcal{ALN}$). For example consider a very simple $\mathcal{ALN}$ A-box $\mathcal{A} = \{R(a,a), (\leq 1.R)(a)\}$. The *msc* for a: $\forall R.\cdots\forall R.(\leq 1.R \sqcap \geq 1.R)$ makes an infinite descending chain of descriptions. Something similar can be obtained also with numeric restrictions.

A solution could be recurring to a different semantics allowing for recursive descriptions. However, it has been shown that such a solution may compromise the tractability of the overall method, since in that case computing the *msc*'s has an exponential cost. Another possible wayout is to allow for approximated *msc*'s [8], for example up to a certain depth related to the maximum cycle in the A-box [12].

[3] In case of cyclic A-boxes, the k-approximation of the *msc* is considered.

$$
\begin{aligned}
\mathit{ABox}\ \mathcal{A} = \{\ & \mathsf{Person}(\mathsf{Meg}), \neg\mathsf{Male}(\mathsf{Meg}), (\forall \mathsf{isMarriedTo}.\bot)(\mathsf{Meg}), \mathsf{Parent}(\mathsf{Meg},\mathsf{Bob}), \mathsf{Parent}(\mathsf{Meg},\mathsf{Pat}),\\
& \mathsf{Person}(\mathsf{Bob}), \mathsf{Male}(\mathsf{Bob}), \mathsf{Parent}(\mathsf{Bob},\mathsf{Ann}),\\
& \mathsf{Person}(\mathsf{Pat}), \mathsf{Male}(\mathsf{Pat}), (\forall \mathsf{isMarriedTo}.\bot)(\mathsf{Pat}), \mathsf{Parent}(\mathsf{Pat},\mathsf{Gwen}),\\
& \mathsf{Person}(\mathsf{Gwen}), \neg\mathsf{Male}(\mathsf{Gwen}), (\forall \mathsf{isMarriedTo}.\bot)(\mathsf{Gwen}),\\
& \mathsf{Person}(\mathsf{Ann}), \neg\mathsf{Male}(\mathsf{Ann}), \mathsf{Parent}(\mathsf{Ann},\mathsf{Sue}), \mathsf{isMarriedTo}(\mathsf{Ann},\mathsf{Tom}),\\
& \mathsf{Person}(\mathsf{Sue}), \neg\mathsf{Male}(\mathsf{Sue}),\\
& \mathsf{Person}(\mathsf{Tom}), \mathsf{Male}(\mathsf{Tom})\}
\end{aligned}
$$

BK descriptions for feature generation $\mathcal{T}_{BK}$

$$
\begin{aligned}
\mathsf{Single} &\equiv \mathsf{Person} \sqcap \leq 0.\mathsf{isMarriedTo};\\
\mathsf{Mother} &\equiv \neg\mathsf{Male} \sqcap \forall \mathsf{Parent}.\mathsf{Person} \sqcap \geq 1.\mathsf{Parent};\\
\mathsf{GrandParent} &\equiv \mathsf{Person} \sqcap \forall \mathsf{Parent}.(\mathsf{Person} \sqcap \forall \mathsf{Parent}.\mathsf{Person} \sqcap \geq 1.\mathsf{Parent}) \sqcap \geq 1.\mathsf{Parent}
\end{aligned}
$$

Positive (resp. negative) instances for the target concept: $I^+ = \{\mathsf{Meg}, \mathsf{Gwen}\}$ ($I^- = \{\mathsf{Ann}, \mathsf{Pat}\}$).
Generated examples: $P = \{p_1, p_2\}$ and $N = \{n_1, n_2\}$ where

$$
\begin{aligned}
p_1 &= \mathsf{Single} \wedge \mathsf{Mother} \wedge \mathsf{GrandParent} && \text{(for Meg)}\\
p_2 &= \mathsf{Single} \wedge \mathsf{Mother} && \text{(for Gwen)}\\
n_1 &= \neg\mathsf{Single} \wedge \mathsf{Mother} && \text{(for Ann)}\\
n_2 &= \mathsf{Single} \wedge \neg\mathsf{Mother} && \text{(for Pat)}
\end{aligned}
$$

whose easy generalization is: $\mathsf{Single} \wedge \mathsf{Mother}$.

Fig. 3. Toy example: a kinship learning problem.

Example 3.3 (Kinship learning problem). Fig, 3 contains an ABox, $\mathcal{A}$, and a background TBox, $\mathcal{T}_{BK}$, with some descriptions employed for feature generation that are typical in a kinship learning problem. In particular, the description Single concerns an individual that is *non-married person*, Mother describes an individual that is *a non-male (female) parent of at least one person*, and GrandParent regards an individual that is *a person that is parent of at least a person that is parent of at least another person*.

Now, if Meg and Gwen are deemed as positive instances for the target concept, and Ann and Pat as negative instances, the corresponding examples for the propositional learning problem are to be generated; e.g., to generate features related to Gwen, one has to check whether $msc_{\mathcal{A}}(\mathsf{Gwen}) = (\mathit{Person} \sqcap \neg\mathsf{Male} \sqcap \forall \mathsf{isMarriedTo}.\bot)$ is subsumed by some BK description, obtaining the example p_2 reported in Fig. 3. After the feature generation phase, it is easy to see how the two positive examples may be generalized in order to induce a consistent propositional description.

4 Applicability

We intend to discuss the efficiency of the feature generation method in this setting and its applicability. The overall algorithm including the FGF as a preprocessing phase would act as follows. Each interpretation is processed using a set of generating descriptions as a background knowledge of *types*. After the preliminary feature generation phase, a vector of active features is generated per interpretation which is then passed to the learning algorithm. As mentioned before, an algorithm that can work in variable-length vector of features is more suitable [5] (there could be an unlimited number).

Just like the original method, the adaptation to $\mathcal{ALN}$ presented here is tractable. Indeed, similarly to that paradigm, the following result holds:

Theorem 4.1 (FGF complexity). *Let I be an interpretation and let D be an $\mathcal{ALN}$ generating description. There is a FGF χ_{msc}, based on the msc operator, that is capable of generating all of the active features in polynomial time.*

Proof. Let I be an interpretation that is the canonical model of an A-box $\mathcal{A}$. The algorithm presented in Fig. 2 computes all the active features by finding the msc of each individual and performing a subsumption query. When the input description D subsumes such an msc the corresponding feature can be considered as being active.

Now, the description $M = msc_{\mathcal{A}}(a)$ can be recursively constructed as follows:

$$\mathsf{prim}(M) = \bigsqcap_{C(a)\in\mathcal{A}} C$$
$$\mathsf{val}(M) = \bigsqcap_{R\in N_R} \bigsqcap_{R(a,b)\in\mathcal{A}} \mathsf{msc}_{\mathcal{A}}(b)$$
$$\mathsf{max}(M) = \mathsf{min}(M) = |\{b \in \mathsf{Ind}(\mathcal{A}) \mid R(a,b) \in \mathcal{A}\}|$$

It is easy to see that computing the msc is linear in the depth of the A-box graph.

The algorithm is dominated by the construction of the msc and by the subsumption which are both polynomial in $\mathcal{ALN}$ (provided that no cycle occurs in the A-box).

In the original framework on using DLs for feature generation [3], the simple language FDL was adopted which is roughly equivalent with $\mathcal{EL}$ [10] with a *concrete domain* [4] for expressing attributes as datatype properties. This DL is basically endowed with two constructors: conjunction and qualified existential restriction ($\exists R.C$). As previously discussed, theoretically also this setting may suffer of the presence of cycles in the graph representing the interpretation.

As regards the problem of cyclic A-boxes, the characterization of concept descriptions is terms of regular languages should be exploited [8]. Besides, a change of semantics should be made in order to take into account cyclic definitions. However, the computation of *msc*'s would not be tractable unless recurring to approximations [12].

Learning directly $\mathcal{ALN}$ representations may compromise the effectiveness of the whole process. Indeed the standard generalizing operator for such description, the *least common subsumer* (*lcs*) applied to *msc*'s [13], is known to yield poorly predictive generalizations. In our case, the exploitation of a tractable feature generation method allows the application of efficient algorithms for propositional representations which can handle cases with very large number of features [5, 11].

5 Conclusions and Future Work

In the line of the paradigm for feature generation employing a DLs knowledge base as a collection of relational types, we have shown a method where a standard DLs language like $\mathcal{ALN}$ is adopted. This extends the applicability to different features with respect the original paradigm, namely universal and numeric restrictions, maintaining the tractability of feature generation process. Moreover this can also be considered a base for future extensions of the method toward even more expressive languages in the DLs family.

This work could be extended towards more expressive languages endowed with union and full negation in order to support completely the semantic concept models mentioned before [6]. This would allow for the exploitation of available pieces of knowledge encoded in DLs to be used as a sort of background knowledge in the manner indicated in the paper. Besides, the adoption of DLs languages with concrete domains [4] may help to constrain more the search space, thus augmenting the efficiency of the learning process. The next step will include investigation on the employment of feature construction techniques in order to automatize the setup of the generating features, e.g. concept learning algorithms applicable to DLs descriptions.

References

[1] Valiant, L.G.: Robust logics. In: Proceedings of the 31st Annual ACM Symposium on the Theory of Computing. (1999) 642–651

[2] Kramer, S., Lavrač, N., Džeroski, S.: Propositionalization approaches to relational data mining. In Džeroski, S., Lavrač, N., eds.: Relational Data Mining. Springer (2001)

[3] Cumby, C.M., Roth, D.: Learning with feature description logics. In Matwin, S., Sammut, C., eds.: Inductive Logic Programming, 12th International Conference, ILP2002. Volume 2583 of LNCS., Springer (2002) 32–47

[4] Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P., eds.: The Description Logic Handbook. Cambridge University Press (2003)

[5] Blum, A.: Learning boolean functions in an infinite attribute space. Machine Learning **9** (1992) 373–386

[6] Calvanese, D., Lenzerini, M., Nardi, D.: Unifying class-based representation formalisms. Journal of Artificial Intelligence Research **11** (1999) 199–240

[7] Molitor, R.: Structural subsumption for $\mathcal{ALN}$. Technical Report LTCS-98-03, LuFg Theoretical Computer Science, RWTH Aachen, Germany (1998)

[8] Baader, F., Küsters, R.: Computing the least common subsumer and the most specific concept in the presence of cyclic $\mathcal{ALN}$ concept descriptions. In Herzog, O., Güenter, A., eds.: Proceedings of the 22th Annual German Conference on Artificial Intelligence. Volume 1504 of LNAI., Springer (1998) 129–140

[9] Donini, F.M., Lenzerini, M., Nardi, D., Nutt, W.: The complexity of concept languages. Information and Computation **134** (1997) 1–58

[10] Küsters, R., Molitor, R.: Approximating most specific concepts in description logics with existential restrictions. In Baader, F., Brewka, G., Eiter, T., eds.: Proceedings of the Joint German/Austrian Conference on Artificial Intelligence, KI/ÖGAI01. Volume 2174 of LNCS., Springer (2001) 33–47

[11] Carleson, A., Cumby, C., Rosen, J., Roth, D.: The SNoW learning architecture. Technical Report UIUCDCS-R-99-2101, CS Dept., University of Illinois at Urbana-Champaign (1999)

[12] Mantay, T.: Commonality-based ABox retrieval. Technical Report FBI-HH-M-291/2000, Department of Computer Science, University of Hamburg, Germany (2000)

[13] Cohen, W.W., Borgida, A., Hirsh, H.: Computing least common subsumers in description logic. In: Proceedings of the 10th National Conference on Artificial Intelligence, AAAI92, MIT-Press (1992)