A Formal Framework for Theory Learning using Description Logics

Jordi Alvarez
TALP Research Center
Universitat Politècnica de Catalunya
Mòdul C6, Campus Nord, 08034 Barcelona
jalvarez@lsi.upc.es

Abstract

This paper introduces a formal base in order to talk about theory learning from a Description Logics (DL) perspective. A probabilistic Description Logics is introduced; and its need is intuitively justified. Theory learning is defined using information theory concepts and the probabilistic DL framework. Once this has been done, a general theory learning environment is constructed on this theory. This environment is based on the application of a set of induction rules. New rules can be defined easily using a set of syntactic manipulators that modify concept expressions.

1 Introduction

Theory learning has been a main goal of machine learning from its beginning. Inductive Logic Programming (ILP) is the only area in machine learning that uses a representational formalism rich enough to make possible to learn an expressive theory.

Some work has been done also in the DL community. DL is an evolution of semantic networks and frame system that uses model theory to give a formal base to the representation and reasoning system [Donini et al., 1996]. Most DL learning stuff is related with the computation of the Least Common Subsumer (LCS), introduced in [Cohen et al., 1992] as an adaptation of Relative Least General Generalization to the DL field. See for example [Cohen and Hirsh, 1994a, Frazier and Pitt, 1996]. The work of [Kietz and Morik, 1994] by one hand, and [Cohen and Hirsh, 1994b] by another, try to acquire a whole theory (using the LCS computation as a subtask).

All this DL work, and most ILP one have been done from a concept learning perspective. In this paper I establish a formal framework that addresses the problem of theory learning as a whole, from a quite general point of view. In order to achieve that, model theory has been used in the reverse direction in which is used in DL formalization. Information theory measures are used in order to give a concrete definition for theory learning. This definition provides a way to devise algorithms that deal with theory learning as a whole.

The formalization introduced in this paper is based in probabilistic DL. Probabilistic DL has been boarded before in [Jaeger, 1994, Heinsohl, 1994, Koller et al., 1997]. In section 3, a new and simple approach to probabilistic DL is introduced.

A learning algorithm based on the introduced theory has been designed and a preliminary implementation performed as the YAYA system.

Although the theory introduced in this paper differs from ILP theory in some basic things (specially in the underlying representation being used), the YAYA learning system described later has a lot of commonalities with ILP systems. And the usage of DL representation formalism in combination with ILP techniques can be very interesting since one of the main goals of DL systems is to effectively and efficiently represent hierarchical knowledge and other knowledge patterns that usually appear in domain theories. Nevertheless, this has to be further explored.
<table>
<thead>
<tr>
<th>Constructor name</th>
<th>Syntax</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concept name</td>
<td>$A$</td>
<td>$A^2 \subseteq \Delta^2$</td>
</tr>
<tr>
<td>Top</td>
<td>$\top$</td>
<td>$\Delta^2$</td>
</tr>
<tr>
<td>Bottom</td>
<td>$\bot$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>Negation ($\neg$)</td>
<td>$\neg C$</td>
<td>$\Delta^2 \setminus C^2$</td>
</tr>
<tr>
<td>Conjunction</td>
<td>$C \land D$</td>
<td>$C^2 \land D^2$</td>
</tr>
<tr>
<td>Disjunction ($\lor$)</td>
<td>$C \lor D$</td>
<td>$C^2 \lor D^2$</td>
</tr>
<tr>
<td>Universal quantification ($\forall$)</td>
<td>$\forall R.C \quad {d_1 : \forall d_2 : (d_1,d_2) \in R \rightarrow d_2 \in C^2}$</td>
<td></td>
</tr>
<tr>
<td>Existential quantification ($\exists$)</td>
<td>$\exists R.C \quad {d_1 : \exists d_2 : (d_1,d_2) \in R^2 \land d_2 \in C^2}$</td>
<td></td>
</tr>
<tr>
<td>Number restrictions ($\leq$)</td>
<td>$\leq n S \quad {d_1 : #{d_2 : (d_1,d_2) \in S} \leq n}$</td>
<td></td>
</tr>
<tr>
<td>Number restrictions ($\geq$)</td>
<td>$\geq n S \quad {d_1 : #{d_2 : (d_1,d_2) \in S} \geq n}$</td>
<td></td>
</tr>
<tr>
<td>Loop</td>
<td>$[R]$</td>
<td>${d : (d,d)^2 \in R^2}$</td>
</tr>
</tbody>
</table>

Table 1: Syntax and semantics of YAYA concept language constructs. $A$ is a concept name, $C$ and $D$ are general concept expressions, $R$ (and $R_i$) can be any kind of role expression, $S$ must be a simple (non-composite) role, and $a_i$ are ABox individuals.

<table>
<thead>
<tr>
<th>Constructor name</th>
<th>Syntax</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Role name</td>
<td>$R$</td>
<td>$P^2 \subseteq \Delta^2 \times \Delta^2$</td>
</tr>
<tr>
<td>Inverse role</td>
<td>$R^-\quad {(d_1,d_2) : (d_2,d_1) \in R}$</td>
<td></td>
</tr>
<tr>
<td>Role composition</td>
<td>$R_1 \circ R_2 \quad {(d_1,d_2) : \exists e \in \Delta \land (d_1,e)^2 \in R_1^2 \land (e,d_2)^2 \in R_2^2}$</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Syntax and semantics for role constructors.

2 Description Logics

DL was born from the need to provide a formal and homogeneous base to semantic network and frame systems (we can find its origins in [Brachman and Schmolze, 1985]). This need is achieved by one hand by the clear distinction between intensional and extensional knowledge; and by other hand by the definition of the language used to express intensional language, commonly called Concept Language, and the definition of its semantics in a model theoretic way.

2.1 Concept Language

The concept language is usually a subset of first order predicate calculus; although its syntax is a bit different. Figure 1 shows the syntax and semantics for YAYA concept language concept-forming constructors. And figure 2 shows the syntax and semantics for YAYA concept language role-forming constructors. The reader can refer to [Donini et al., 1996] for a deep description of DL topics; and to [Alvarez, 2000b] for a detailed definition of YAYA description logics system.

There exist many DL systems with different concept languages. In this paper we will restrict to YAYA concept language; which is a constituted by the core language $\mathcal{ALCN} (\top, \bot, \land, \lor, \forall, \exists, \leq, \geq, \land, \lor, P \text{ constructors})$, plus the loop concept constructor, and inverse and composite roles. Obviously, definitions in section 3 and section 4 are valid for any other DL concept language.

From now on, we will use $\mathcal{L}$ to refer to the concept language in use and the set of terms in the concept language indistinctly. We will also use $\mathcal{LR}$ as the set of role expressions that can be constructed in $\mathcal{L}$.

A conceptual interpretation $\mathcal{I} = < \Delta^2, \mathcal{I}>$ consists of a non-empty set $\Delta^2$ (the domain); and a function $\mathcal{I}$ that maps every concept in $\mathcal{L}$ into a subset of $\Delta^2$, and every role expression in $\mathcal{LR}$ into a subset of $\Delta^2 \times \Delta^2$.

We have introduced the term conceptual interpretation in order to distinguish it from what will later be called specific interpretation. Description logics literature defines an only notion of interpretation. Our motivation to distinguish two slightly different notions of interpretation will be clear later, when the process of learning a theory
Then, an interpretation $I$ is a \textit{model} for a concept $C$ if $C^I$ is nonempty. A concept is \textit{satisfiable} if it has a model. A concept $C$ is \textit{subsumed} by another concept $D$ if $C^I \subseteq D^I$ for every interpretation $I$. Two concepts $C$ and $D$ are \textit{equivalent} if for every interpretation $I$, $C^I = D^I$.

### 2.2 Knowledge Bases

A knowledge base $\Sigma$ is defined as a pair constituted by a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$ ($\Sigma = < \mathcal{T}, \mathcal{A} >$).

#### 2.2.1 Intensional Knowledge

$\mathcal{T}$ contains the intensional knowledge and is constituted by a set of axioms. Different types of axioms do appear in the description logics literature (see [Donini et al., 1996] for details). Depending on the type of axioms allowed by a description logics system, it will be more or less expressive. YAYA only deals with general concept inclusion axioms, that have the following form:

$$C \leq D$$

where $C$ and $D$ are CL terms.

All other types of axiom can be reformulated as general concept inclusion axioms.

Different semantics can be used in order to define the semantics for TBoxes with general concept inclusions\(^2\). We use the so-called \textit{descriptive semantics}, following the approach of [Buchheit et al., 1993] (instead of the more restrictive least and greatest fix-point semantics [Baader, 1990, Nebel, 1991]).

Then, an interpretation $I$ satisfies the axiom $C \leq D$ if $C^I \subseteq D^I$. And an interpretation $I$ is a \textit{model} for a TBox $\mathcal{T}$ if it satisfies all the axioms in $\mathcal{T}$.

For our convenience, we define:

$$\text{Models}(\mathcal{T}) = \{I | I \text{ is a model for } \mathcal{T} \}$$

#### 2.2.2 Extensional Knowledge

The ABox $\mathcal{A}$ provides the extensional knowledge of a knowledge base $\Sigma$. The ABox provides information about a set of concrete objects called \textit{individuals}.

The notion of conceptual interpretation $I = < \Delta^I, -^I >$ is extended to that of \textit{specific interpretation} in order to cope with individuals. In a specific interpretation, the function $-^I$ also maps every individual for which $\mathcal{A}$ provides information to an element of $\Delta^I$.

The extensional knowledge is provided as a set of assertions. Every assertion is of the form:

\begin{itemize}
  \item \textbf{C(a)} \ \textit{concept membership assertion}
  \item \textbf{R(a,b)} \ \textit{role membership assertion}
\end{itemize}

where $C \in \mathcal{L}$, $R \in \mathcal{LR}$, and $a$, $b$ are individuals.

For purposes that will become clearer later we define a \textit{simple assertion} as a concept membership assertion of the form $A(a)$, or a role membership assertion of the form $S(a, b)$, where $A$ is a concept name, $S$ is a role name, and $a$, $b$ are individuals.

An interpretation $I = < \Delta^I, -^I >$ satisfies $C(a)$ if $a^I \in C^I$. And $I$ satisfies $R(a, b)$ if $(a^I, b^I) \in R^I$.

An ABox $\mathcal{A}$ is defined to be a set of assertions. Then, an interpretation $I$ is a \textit{model} of $\mathcal{A}$ if it satisfies all the assertions in $\mathcal{A}$. $\mathcal{A}$ is \textit{satisfiable} if it has a model.

\(^2\)Notice that general concept inclusions allow to have axioms stating for example that all mammals are son of another mammal ($\text{mammal} \subseteq \exists \text{parent.mammal}$) . . .
3 Probabilistic Description Logics

YAYA TBox allows probabilistic axioms of the form:

$$C \leq_{\alpha, \sigma} D$$ (3)

Where $C$ and $D$ are $\mathcal{L}$ concept expressions, $\alpha$ and $\sigma$ are real numbers such that $\alpha \in [0, 1]$, and $\sigma \geq 0$.

Axioms of this form will be called probabilistic concept inclusions. Axiom $C \leq_{\alpha, \sigma} D$ states that the conditional probability $P(D|C)$ follows a normal distribution centered in $\alpha$ and with a standard deviation of $\sigma$ in the set of interpretations satisfying it.

Then, the notion of satisfiability used for non-probabilistic DL is extended to that of $\beta$-satisfiability in the following way: given an interpretation $I = \langle \Delta^I, \mathcal{T} \rangle$, we say that $I$ $\beta$-satisfies the axiom $C \leq_{\alpha, \sigma} D$ if $P(D^I|C^I) \in N(\alpha, \sigma)$ with probability $\geq 1 - \beta$.

The notion of $T$-model is also extended to $\beta$-plausible $T$-model. A conceptual interpretation $I$ is a $\beta$-plausible model of $T$ if $I$ $\beta$-satisfies all the axioms in $T$.

Then, an important property about TBoxes with probabilistic axioms (probabilistic TBoxes) is that they preserve monotonicity from a model point of view; that is, given two probabilistic TBoxes $T_1$ and $T_2$, if $T_1 \subseteq T_2$ then $\text{Models}_{\beta}(T_1) \subseteq \text{Models}_{\beta}(T_2)$.

3.1 Axiom Estimation

Probabilistic tboxes allow to easily “extract” axioms from a set of interpretations. And that is what theory learning is about. In our probabilistic environment, given a theory $T$; then, for every two concepts $C$ and $D$ we can find an $\alpha$ and a $\sigma$ for which we can build an axiom $\Phi = C \leq_{\alpha, \sigma} D$ such that $\text{Models}(T \cup \Phi) = \text{Models}(T)$.

For every interpretation $I$ and every pair of concepts $C$ and $D$, we have a set of boolean variables, one for every element $d \in C^I$, stating whether $d \in D$ or not. Using the central limit theorem, for sufficiently large values of $|C^I|$, we can approximate $P(D^I|C^I)$ by a normal distribution.

Then, given a set of interpretations $M = \{I_1, I_2, \ldots, I_n\}$ constituted by randomly chosen models of a TBox $T$; and given two concepts $C$ and $D$, we can always compute $\alpha$ and $\sigma$ such that $\text{Models}(\langle C \leq_{\alpha, \sigma} D \rangle)$. This fact is quite important and will be refereed later in the description of the learning algorithm.

An approximation for $\alpha$ can be computed as:

$$\hat{\alpha} = \frac{\sum_{\mathcal{I} \in \mathcal{M}} P(D^I|C^I)}{|\mathcal{M}|}$$ (4)

Obviously, the set of interpretations that satisfy axiom $C \leq_{\alpha, \sigma} D$ is a subset of those satisfying $C \leq_{\alpha', \sigma'} D$ iff $\sigma < \sigma'$. So, for big $\sigma$, axioms of the form $C \leq_{\alpha, \sigma} D$ are not very helpful in order to define the behaviour of $\mathcal{M}$. So, we would like to obtain a $\hat{\sigma}$ such that $\hat{\sigma} \in \text{minimum possible value such that Models}(\langle \Phi \rangle) = \text{Models}(T)$. In order to do so, we can use a $\chi^2$ test. We know that:

$$\frac{\sum_{\mathcal{I} \in \mathcal{M}}(P(D^I|C^I) - \nu)^2}{\sigma^2} \in C^2,$$ (5)

where $C^2 = \frac{\chi^2}{|\mathcal{M}|-1}$.

A confidence interval has to be provided for $\sigma$ such that $P(\sigma < \hat{\sigma}) \leq \gamma$. Where $\gamma$ is a new parameter governing axiom estimation. We approximate $\sigma$ as:

$\text{Models}_{\beta}(T)$ is the set of $\beta$-plausible models of $T$. 

where \( C_{\gamma,M}^2 \) is such that the probability that (5) \( \geq C_{\gamma,M}^2 \) is \( 1 - \gamma \) (provided that (5) \( \in [0,1] \)).

Bigger \( \gamma \) imply a conservative policy in axiom estimation: low risk in generating an axiom that is not in \( \mathcal{T} \); and high probability that there exists a “better” axiom in order to define the behaviour of \( P(D|C) \). For lower \( \gamma \) we have a higher risk of generating a theory \( \mathcal{T}' \) such that \( Models(\mathcal{T}) \not\subseteq Models(\mathcal{T}') \).

Even though for some pairs of concepts \( C \) and \( D \) we will not be able to estimate axioms with low \( \hat{\sigma} \), all probabilistic axioms, even those providing little information, can be used as an important source of information to guide the search done by the learning process. Instead, if our system was restricted to non-probabilistic axioms, then: (a) the search had to be restricted to non-probabilistic axioms for the set of theory models \( M \) from which we are learning; or (b) we had to approximate probabilistic axioms as the closer non-probabilistic version of it. Both options suppose a drawback for the learning process.

Probabilistic DL has been boarded previously [Jaeger, 1994, Heinsohn, 1994, Koller et al., 1997]. Nevertheless, YAYA approach to probabilistic DL differs from both of them. From now on, we will refer to \( \beta \)-plausible models of a given \( \mathcal{T} \) for a fixed \( \beta \) simply as models of \( \mathcal{T} \).

4 Theory Learning

In this section we will formally define the process of theory learning from a pair \( < T_0, M > \); where \( T_0 \) is an initial theory we have, and \( M = \{ I_1, I_2, \ldots I_n \} \) is a set of conceptual interpretations. \( T_0 \) is a subset of the theory to be learned \( \mathcal{T}^* \) and is commonly referred as background knowledge, and the conceptual interpretations are supposed to be models of \( \mathcal{T}^* \).

The reasoning procedures performed by DL systems can be seen as the process of determining the set of models of a given knowledge base; and then check some properties over those models (that corresponds to the obtained result). In the same way, theory learning can be seen as the process of determining a TBox \( \mathcal{T}^* \) from a subset of its models.

Example Let’s consider an example: an adaptation of the arch learning task introduced by Winston. Figure 1 shows several arches, each one of them could correspond to a conceptual interpretation in \( M \). Elements in the domain correspond to physical objects. The conceptual interpretations contain a set of concepts and roles that are represented in a graphical way in figure 1.

![Figure 1: Three examples of arch. Each one of them corresponds to a model of the theory to be learned.](image)

The set of used concepts are: on_floor (the object is on the floor), flat_top (the object has a flat top), flat_bottom (it has a flat bottom), brick (it is a brick), wedge (it is a wedge), and
cylinder (it is a standing cylinder); and the set of roles: supports (relates an object to another object supported by the first one), and touches (relates two objects that are in contact and are at the same level (one object is not supported by the other)).

The theory to be learned (\(T^*\)) corresponds to a theory such that \(M\) is a subset of Models(\(T^*\)), and additionally, it contains as much redundant information in \(M\) as it can be captured with the used concept language. Then, if \(M\) is representative enough for the whole set of arches, and the concept language being used is expressive enough, \(T^*\) would correspond to an axiomatization for arches. The concept language used in this example is \(\mathcal{ALCN}\).

The initial theory \(\mathcal{T_0}\) could correspond to a set of axioms trying to capture physical world laws restricted to our representation environment. Both \(\mathcal{T_0}\) and a possible \(T^*\) are shown in table 3\(^4\).

<table>
<thead>
<tr>
<th>(A) name</th>
<th>(T) name</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\neg \text{on-floor} \in \text{supports} )</td>
<td>(\neg \text{archer})</td>
</tr>
<tr>
<td>(\text{supports} T \subseteq \neg \text{on-floor} )</td>
<td>(\text{on-floor} \subseteq \neg \text{archer} )</td>
</tr>
<tr>
<td>(\text{trick} \subseteq \neg \text{flat-left} )</td>
<td>(\text{trick} \subseteq \neg \text{flat-left} )</td>
</tr>
<tr>
<td>(\text{cylinder} \subseteq \neg \text{trick})</td>
<td>(\text{cylinder} \subseteq \neg \text{trick})</td>
</tr>
<tr>
<td>(\text{trick} \subseteq \neg \text{cylinder})</td>
<td>(\text{trick} \subseteq \neg \text{cylinder})</td>
</tr>
</tbody>
</table>

Table 3: \(\mathcal{T_0}\) and a possible \(T^*\) for the arch learning problem.

### 4.1 Model Representation

In order to be able to learn from conceptual interpretations, every interpretation \(\mathcal{I}\) should be represented in some way. As the interpretation provides extensional knowledge about the domain \(\Delta^I\), an easy way to represent an interpretation \(\mathcal{I}\) is through an ABox \(\mathcal{A}\).

A new ABox semantics will be defined in order to obtain an ABox that strictly represents \(\mathcal{I}\). In order to do so, we first define an equivalence relation between conceptual interpretations.

We say two conceptual interpretations \(\mathcal{I}_1\) and \(\mathcal{I}_2\) are equivalent if there exists a bijection \(\tau\) between \(\Delta^{I_1}\) and \(\Delta^{I_2}\) such that:

1. \(C^{I_1} = \{t | \tau(t) \in C^{I_2}\}\) for every \(C \in \mathcal{L}\), and
2. \(R^{I_1} = \{(s,t) | (\tau(s), \tau(t)) \in R^{I_2}\}\) for every \(R \in \mathcal{R}\), and

The ABox of an interpretation \(\mathcal{I}\) should be such that \(\Sigma = \langle \emptyset, \mathcal{A} \rangle\) will have as models all specific interpretations that are \(\mathcal{A}\)-extensions of conceptual interpretations equivalent to \(\mathcal{I}\); and no other interpretation should be a model of \(\mathcal{A}\). This assures us that only the domain objects may change in \(\mathcal{A}\) models; but the behaviour of objects in the domain and the relations among those objects are completely defined by \(\mathcal{A}\). \(\tau\) relates objects with the same behavior.

For computational and storage reasons, we define a new semantics called closed-world semantics. Closed-world semantics uses close world assumption in order to avoid the need to have negative concept assertions. We say that a specific interpretation \(\mathcal{I} = \langle \Delta^I, \tau \rangle\) is a closed-world model of \(\Sigma = \langle \mathcal{T}, \mathcal{A} \rangle\) if:

1. \(\mathcal{I}\) is a model of \(\Sigma\), and
2. For every concept name \(A \in \mathcal{L}\), and for every individual \(a\) that appears in \(\mathcal{A}\):
   - \(a^I \notin A^I\) if \(\langle \mathcal{T}, \mathcal{A} \rangle\) has a (standard) model \(\mathcal{T}_0\) in which \(a^{\mathcal{T}_0} \notin A^{\mathcal{T}_0}\)
3. For every role name \(P \in \mathcal{L}\), and for every pair of individuals \(a, b\) in \(\mathcal{A}\):
   - \((a, b) \notin P^I\) if \(\langle \mathcal{T}, \mathcal{A} \rangle\) has a model \(\mathcal{T}_0\) such that \((a^{\mathcal{T}_0}, b^{\mathcal{T}_0}) \notin P^{\mathcal{T}_0}\)

\(^4\)For the sake of simplicity, the arch model being used is quite simple (two supporting objects of the same height and a supported object). In the same direction, \(\mathcal{T}_0\) is also very simplistic, allowing some situations that cannot occur in our gravitative physical world: two objects supporting each other. Or one object supporting itself, for example.

\(^5\)The term \(\mathcal{A}\)-extension is used here in the sense that a conceptual interpretation is converted to a concrete interpretation by adding a mapping from \(\mathcal{A}\) individuals to \(\Delta^I\).
The second and third points are only defined for concept names because the rest of the work is done by the semantics of \( \mathcal{L} \) constructors.

Note that with this definition, consistent knowledge bases may have no closed-world model. Take for example, \( \Sigma = (\mathcal{T}, \mathcal{A}) \), such that \( \mathcal{T} = \{ \neg A \leq B \} \), and \( \mathcal{A} = \{ \langle R(a, b) \rangle \} \). Then, \( \Sigma \) has a model \( I_0 \) in which \( a^{I_0} \notin A^{I_0} \), and another one \( I_1 \) in which \( a^{I_1} \notin B^{I_1} \). But it has no model \( I \) in which \( a^I \in \neg A^I \land a^I \in \neg B^I \).

From this point on, we will use the term closed-world ABox for ABoxes for which we assume closed-world semantics; and closed-world knowledge bases for knowledge bases with a closed-world ABox. And we will refer to the set of models of a closed-world knowledge base \( \Sigma = (\mathcal{T}, \mathcal{A}) \) as \( \text{Models}_{cw}(\Sigma) \).

Taking all this into account, we can compute the representation of a conceptual interpretation \( \mathcal{I} \) as a closed-world ABox \( \mathcal{A} \) in the following way:

1. For every object \( o_i \in \Delta^I \) we define a new individual \( a_i \). We also define a function \( \tau^R \) that maps every object \( o_i \) to its corresponding individual \( a_i \).
2. For every object \( o \in \Delta^I \) and every concept name \( A \in \mathcal{L} \) add the assertion \( A(o^{\mathcal{I}}) \) if \( o \in A^I \).
3. For every pair of objects \( (o_1, o_2) \in \Delta^I \) and every role name \( S \in \mathcal{L}R \) add the assertion \( R(o_1^{\mathcal{I}}, o_2^{\mathcal{I}}) \) if \( (o_1, o_2) \in R^I \).

We will refer to the closed-world ABox computed from a conceptual interpretation \( \mathcal{I} \) using this procedure as: \( \text{Representation}(\mathcal{I})^6 \). Note that \( \text{Representation}(\mathcal{I}) \) is constituted only by simple assertions; and that \( \mathcal{I} \in \text{Models}_{cw}(\mathcal{A}) \), since \( \mathcal{A} \) is being built from \( \mathcal{I} \). Table 4 shows the closed-world ABox resulting from applying this procedure to figure 1 first arch.

### 4.2 Quantifying Description Logics Information

The next step in order to formalize the notion of theory learning is the quantification of the amount of information a given theory \( \mathcal{T} \) is able to predict in \( \text{Models}(\mathcal{T}) \). Then, if we have learned \( \mathcal{T} \) from \( \mathcal{T}_0, \mathcal{M} \) we will be able to state how much redundant information in \( \mathcal{M} \) elements is represented in \( \mathcal{T} \).

Before proceeding, a new kind of ABox assertion is defined: role non-membership assertion \( \neg R(a, b) \), where \( R \in \mathcal{L}R \), and \( a, b \) are individuals. We extend also the notion of simple assertion to this new kind of assertions. An interpretation \( \mathcal{I} \) satisfies the assertion \( \neg R(a, b) \) iff \( (a, b) \notin R^I \). Then, we define an extended ABox as an ABox allowing role-nonmembership assertions; and an extended simple ABox as an extended ABox containing only simple assertions.

As will be shown (and used) later, any interpretation can be represented by a set of boolean variables. Then, the introduction of this kind of assertion allows to establish a correspondence between assertions and boolean variable values (every assertion corresponds to a boolean variable value, and every boolean variable value has a corresponding assertion)\(^7\). Instead, there was the choice to introduce role negation in the concept language; but different to concept negation, concept languages with role negation are not widely used. So, I have preferred not to impose such an expressivity restriction in the concept language. This new type of assertions and extended ABoxes will only be used in this section in order to quantify description logics information.

Then, we define the \( \text{ExtendedRepresentation} \) of a conceptual interpretation \( \mathcal{I} \) as the only extended simple ABox \( \mathcal{A}_E \) such that \( \text{Models}(\mathcal{T}, \mathcal{A}_E) = \text{Models}_{cw}(\text{Representation}(\mathcal{I})) \). Table 4 shows the \( \text{ExtendedRepresentation} \) of the first arch of figure 1.

The \textit{entropy} of a conceptual interpretation \( \mathcal{I} \), \( \text{entropy}(\mathcal{I}) \), is defined as the number of assertions in \( \text{ExtendedRepresentation}(\mathcal{I}) \).

---

\(^6\)The given algorithm is restricted to interpretations with a finite domain.

\(^7\)Note for example that in order to represent the first arch in figure 1 when no intensional knowledge is available it is necessary to use role non-membership assertions in order to state that there is no pair of objects that touches laterally (see also table 4).
Table 4: This table shows the closed-world representation, and the extended representation of arch 1 in figure 1.

The entropy of $\mathcal{I}$ has a direct interpretation as the number of boolean variables necessary to represent $\mathcal{I}$. If no knowledge is available, for every concept name $A$, and every element $d \in \Delta^2$ we need a boolean random variable stating whether $d \in A^2$ (true) or whether $d \notin A$ (false). Additionally, for every role name $R$, and every element pair $(d_1, d_2) \in \Delta^2 \times \Delta^2$ we need a boolean random variable stating whether $(d_1, d_2) \in R^2$ or not. When no knowledge is available (and then we suppose that the probability for true is the same that the probability for false, and is equal to $\frac{1}{2}$), the entropy for any of these boolean random variables is $-\frac{1}{2} \log_{2} \frac{1}{2} - \frac{1}{2} \log_{2} \frac{1}{2} = 1$.

Every assertion in $\mathcal{A}_E$ represents the value of one of these boolean variables. So, the notion of entropy that has been introduced here is a direct adaptation of Shannon measure.

**Example**  The entropy of the arch 1 of figure 1 is 36, as it can be checked by counting the number of assertions in its extended representation (see table 4). In fact, for this simple arch model, it can easily be seen that every arch model can be represented with 36 boolean variables, since the number of domain elements will always be three, and there are six different concept names and two different role names.

The minimum expression of a given extended simple ABox $\mathcal{A}_E$ with respect to a TBox $\mathcal{T}$, $\text{Minimum}_{\mathcal{T}}(\mathcal{A})$, is defined as an extended simple ABox $\mathcal{A}_0$ such that $\text{Models}(\mathcal{T}, \mathcal{A}_0) = \text{Models}(\mathcal{T}, \mathcal{A})$ and $|\mathcal{A}_0| = \text{minimum}(\{|\mathcal{A}_E| : \text{Models}(\mathcal{A}) = \text{Models}(\mathcal{A}_E)\})$.

$\text{Minimum}_{\mathcal{T}}(\mathcal{A})$ determines the minimum number of assertions that can be used to represent the extensional knowledge in $\mathcal{A}$ given the intensional knowledge in $\mathcal{T}$.

Then, we can define the amount of redundant information in an extended simple ABox $\mathcal{A}$ with respect to a TBox $\mathcal{T}$ as:

$$\text{Redundant}_{\mathcal{T}}(\mathcal{A}) = |\mathcal{A}| - |\text{Minimum}_{\mathcal{T}}(\mathcal{A})|.$$

**Example**  Going on with our arch example; and given the initial TBox $\mathcal{T}_0$ in table 3, we can compute $\text{Minimum}_{\mathcal{T}_0}(\text{ExtendedRepresentation}(\text{arch 1}))$. The result is shown in table 5. Then, it follows that $\text{Redundant}_{\mathcal{T}_0}(\text{er}_{\text{arch 1}}) = 19$.

Table 5: This table shows $\text{Minimum}_{\mathcal{T}_0}(\text{ExtendedRepresentation}(\text{arch 1}))(\text{min}_{\text{arch 1}})$. It contains only 17 assertions from the 36 initially present in $\text{ExtendedRepresentation}(\text{arch 1}) (\text{er}_{\text{arch 1}})$. The set of models for $<\mathcal{T}_0, \text{min}_{\text{arch 1}}>\text{ is equal to the set of models for }<\mathcal{T}_0, \text{er}_{\text{arch 1}}>$. That is, the assertions in $\text{er}_{\text{arch 1}} \setminus \text{min}_{\text{arch 1}}$ can be inferred from $<\mathcal{T}_0, \text{min}_{\text{arch 1}}>$.

Now we are able to talk about the amount of information encoded in a TBox $\mathcal{T}$ from a model theoretic point of view. For this purpose, we define:

- The amount of information provided by $\mathcal{T}$ with respect to a model $\mathcal{I}$:

\[\text{Note that there can be several } \mathcal{A}_E \text{ that minimize } |\mathcal{A}_E|. \text{ Minimum}_{\mathcal{T}}(\mathcal{A}) \text{ is one of them.}\]
\[ \text{Inf}_{\text{T}}(T) = \text{Redundant } T(\text{ExtendedRepresentation}(I)) \]

- The amount of information provided by \( T \) with respect to a set of conceptual interpretations \( M = \{I_1, I_2, \ldots, I_n\} \) that are models of \( T \):
  \[ \text{Inf}_{M}(T) = \sum_{\tau \in \text{T}} \text{Inf}_{\tau}(T) \]

4.3 Learning a TBox

Let’s go back to our problem. We want to learn a TBox \( T^* \) from a pair \(< T_0, M >\) constituted by an initial theory \( T_0 \) such that \( T_0 \subseteq T^* \), and a set of \( T^* \)-models \( M = \{I_1, I_2, \ldots, I_n\} \).

Then, we define the process of learning a TBox \( T^* \) from a pair \(< T_0, M >\) as the process of acquiring a set of axioms \( \{\Phi_1, \Phi_2, \ldots, \Phi_m\} \) such that:

1. \( T^* = T_0 \cup \{\Phi_1, \Phi_2, \ldots, \Phi_m\} \)
2. \( M \subseteq \text{Models}(T^*) \)
3. \( \forall \text{TBox } T : T^* \subseteq T \land M \subseteq \text{Models}(T) \Rightarrow \text{Inf}_{M}(T) = \text{Inf}_{M}(T^*) \)
4. \( \forall \text{TBox } T : T_0 \subseteq T \land T \subseteq T^* \Rightarrow \text{Inf}_{M}(T) < \text{Inf}_{M}(T^*) \)

The first condition defines \( T^* \) as the union of \( T_0 \) and the set of acquired axioms. The second one states that the set of conceptual interpretations from which we learn should be models of \( T^* \). The third point states that all TBoxes including \( T^* \) and having \( M \) as models provide no new information. And finally the fourth point states that no subset of \( T^* \) that has \( T_0 \) as a subset provides as much information as \( T^* \) provides.

This definition does not imply uniqueness. It only states clearly from a model theoretic point of view, which conditions must hold \( T^* \).

In order to talk about axioms individually, the following definitions are introduced:

- \( T \) is \emph{consistent} with respect to \( M = \{I_1, I_2, \ldots, I_n\} \) iff \( M \subseteq \text{Models}(T) \).
- An axiom \( \Phi \) is \emph{consistent} with respect to a pair \(< T, M >\) iff \( T \cup \{\Phi\} \) is consistent with respect to \( M \).
- An axiom \( \Phi \) is \emph{interesting} with respect to a pair \(< T, M >\) iff \( \Phi \) is consistent with respect to \( T, M > \), and \( \text{Inf}_{M}(T \cup \{\Phi\}) > \text{Inf}_{M}(T) \).

The two latter definitions can also be extended to sets of axioms \( \{\Phi_1, \Phi_2, \ldots, \Phi_m\} \).

Up to here we have defined the problem of learning a theory \( T^* \) from a set of models \( M \). Nevertheless, we have not said a word about procedures to find \( T^* \) or, if not possible, to find a theory as \emph{similar as possible} to \( T^* \). In any case, the procedure has to perform a search in the space of TBoxes. This is, of course, intractable.

**Example** Given the initial theory in table 3 (\( T_0 \), and arch 1 from figure 1; let’s check whether axiom \( \Phi_9 \) is interesting with respect to \(< T_0, \{\text{arch } 1\} > \).

Minimum \( T_0 \cup \{\Phi_9\} \{\text{er, arch}_1\} \) corresponds to \{on\_floor(a), on\_floor(c), brick(a), brick(b), brick(c), supports(a, b), supports(c, b), \neg supports(b, b)\}. The amount of redundant information is then 28, strictly greater than it was for \( T_0 \). So, \( \Phi_9 \) is interesting for \(< T_0, \{\text{arch } 1\} > \).

Now, doing the same for \( \Phi_{10} \) with respect to \(< T_0 \cup \{\Phi_9\}, \{\text{arch } 1\} > \) we obtain that Minimum \( T_0 \cup \{\Phi_9, \Phi_{10}\} \{\text{er, arch}_1\} \) corresponds to \{brick(a), brick(b), brick(c), supports(a, b), supports(c, b), \neg supports(b, b)\}. So, \( \Phi_{10} \) is interesting for \(< T_0 \cup \{\Phi_9\}, \{\text{arch } 1\} > \).
4.4 Devising a procedure for theory learning

A proposal of a general procedure for approximate theory learning can be found in the next section. The procedure, although it has been based on this theory, does not guarantee any theoretic bound on the obtained theory. Much work has still to be done in this direction. Nevertheless, the procedure has been successfully used in the adaptation of some typical Inductive Logic Programming benchmarks. These paragraphs are intended to provide some reflections that are used by the procedure introduced in the next section.

Now, in order to provide a method for theory learning, we can benefit from the fact that our reasoning system is model-monotonic in order to convert the search in the space of TBoxes into a search in the space of \( \mathcal{L} \) concept inclusion axioms. Then, given a TBox \( \mathcal{T} \) such that \( \mathcal{T}_0 \subseteq \mathcal{T} \subseteq \mathcal{T}^* \):

1. If an axiom \( \Phi \) is found interesting for \( \mathcal{T} \), it is because \( \Phi \in \mathcal{T}^* \), or because \( \mathcal{T}^* \) provides the information in \( \Phi \) through other axioms not in \( \mathcal{T} \).

2. If an axiom \( \Phi \) is found to be not interesting for \( \mathcal{T} \), we have that \( \Phi \notin \mathcal{T} \) or \( \Phi \notin \mathcal{T}^* \).

This provides evidence for the known fact that the difficulty of learning a theory strongly depends on the theory itself, and on the set of examples used to learn it.

The problem of learning a theory for any pair \( \langle \mathcal{T}, \mathcal{M} \rangle \) is still intractable. So, we must perform some kind of tractable heuristic search. This search will not provide the exact solution but a TBox consistent with respect to \( \langle \mathcal{T}, \mathcal{M} \rangle \) that is as close as possible to \( \mathcal{T}^* \).

5 The YAYA learning procedure

This section explains how all the previous theory has been used to provide the base for a tractable learning procedure. First, the search space used by the YAYA learning process is defined. Second, a set of generic \( \mathcal{L} \) concept expression manipulators are introduced. These manipulators are intended to provide a general \( \mathcal{L} \) manipulation framework in which a learning process can be easily defined by using them. Third, a practical approach to \( \mathcal{T} \) information computation is introduced. And finally, a set of induction rules constituting YAYA learning process are introduced. Both information computation procedures and induction rules use concept manipulators.

YAYA provides a learning procedure that is an heuristically guided greedy search over the space of TBox axioms. Greedy methods have been widely used by the machine learning community. Although greedy search does not guarantees, in general, to find the best solution to a problem; it provides a tractable procedure to find a solution.

Greedy search extremely depends on local minima. The number of local minima usually depends on the number of dimensions of the search space: it is probable that the more dimensions the search space has, the less local minima it has. YAYA induction rule set has been designed so to overcome the drawbacks of greedy search as much as possible.

A YAYA learning state is defined as \( \langle \mathcal{T}, \mathcal{CN}, \mathcal{RN}, \mathcal{FC} \rangle \). Where \( \mathcal{T} \) is a TBox such that \( \mathcal{T}_0 \subseteq \mathcal{T} \), \( \mathcal{CN} \) is a set of concept names plus \( \top \), \( \mathcal{RN} \) is a set of role names, and \( \mathcal{FC} \) is a set of forbidden and redundant \( \mathcal{L} \) expressions (they help to importantly reduce the search space in some situations). None of the axioms in \( \mathcal{T} \) contains a concept name that is not in \( \mathcal{CN} \) nor a role name that is not in \( \mathcal{RN} \). Concept expressions in \( \mathcal{FC} \) cannot appear in a \( \mathcal{T} \)-axiom as a subconcept; and they can only appear in the left hand side of \( \mathcal{T} \) axioms. The sets \( \mathcal{CN} \) and \( \mathcal{RN} \) restrict the set of concept names and role names that can take part in \( \mathcal{T} \) axioms. \( cn \) and \( rn \) are subsets of the set of concept and role names appearing in \( \mathcal{M} \).

The initial learning state is: \( \langle \mathcal{T}_0, \{ \top \}, \{ S | S \text{ is a role name in } \mathcal{M} \text{ or its inverse } \}, \emptyset \rangle \). So, YAYA learning system implements a top-down approach. As has been stated above, for any two concepts, and a set of models \( \mathcal{M} \), we are able to find a probabilistic axiom \( \Phi_1 = C \leq_{a, \sigma} D \) such that \( \mathcal{M} \sqsubseteq \text{Models}(\{ \Phi_1 \}) \). This means that due to the use of probabilistic knowledge, YAYA learning system does not need any specialization step, as intermediate theories will never be overly specific.
YAYA learning algorithm performs a greedy search. The search is performed by a set of induction rules that modify the search state. Induction rules have been designed in order to overcome the drawbacks of greedy algorithms. They are applied according to a set of heuristics based information theory measures that can be computed efficiently and allow to compute an approximation to the measures defined in section 4.3. The description of these measures can be found in section 5.2.

5.1 Concept Generation

The set of concepts that can take part in axioms generated by the YAYA learning procedure are restricted by the set of induction rules provided by YAYA. In order to distinguish between the language supported by the reasoning system and the language the learning system is able to generate we will refer to the second one as \( \mathcal{L}_G \). It holds that \( \mathcal{L}_G \subset \mathcal{L} \). The \( \mathcal{L}_G \) constructors are: \( \wedge, \vee, \neg, \exists, \leq, \geq, [R], \circ \) and role inversion. Moreover, the way in which these constructors can be combined is also restricted by the set of induction rules.

The YAYA learner explores new axioms from the set of current axioms in \( \mathcal{T} \). New axioms are generated by manipulating the concepts that take part in axioms in \( \mathcal{T} \). There are several different concept manipulation procedures that are used by YAYA to obtain a concept from another one. Before explaining them, we need some additional definitions:

- We define a simple concept as a concept containing only constructors in \( \{ A, \top, \bot, \wedge, \vee, \leq_n, \geq_n \} \).
- \( \text{parent}_{\mathcal{T}, \mathcal{CN}}(A, B) \) \( \triangleq \forall I \in \text{Models}(\mathcal{T}) : A^I \subseteq B^I \), where \( A, B \in \mathcal{CN} \)
- \( \text{direct\_parent}_{\mathcal{T}, \mathcal{CN}}(A, B) \) \( \triangleq \text{parent}(A, B) \land \forall A' \in \mathcal{CN} : \exists I \in \text{Models}(\mathcal{T}) : A'^I \subset A^I \lor B^I \subset A'^I \), where \( A, B \in \mathcal{CN} \).
- We will say a concept name \( A \) is the son (direct-son) of another concept name \( B \) if \( B \) is the parent (direct-parent) of \( A \).

Now we are able to define concept manipulators:

- \( \text{append} : \mathcal{L}_G \times \mathcal{L}_G \rightarrow \mathcal{L}_G \)
  \( \text{append}(C, D) \) computes a concept subsumed by \( C \) in which \( D \) has been situated at the end of the chain of existentials of \( C \).

  \[
  \text{append}(G, C) = G \cap C \\
  \text{append}(G \cap \exists R.C, D) = G \cap \exists R. \text{fix}(C) \cap \text{append}(C, D)
  \]

- \( \text{concept\_names} : \mathcal{L}_G \rightarrow 2^{\mathcal{CN}} \)
  \( \text{concept\_names}(C) \) returns a set with the set of concept names appearing in \( C \).

  \[
  \text{concept\_names}(A) = \{ A \} \\
  \text{concept\_names}(\top) = \top \\
  \text{concept\_names}(G \cap C) = \text{concept\_names}(G) \cup \text{concept\_names}(C) \\
  \text{concept\_names}(\exists R.C) = \text{concept\_names}(C) \\
  \text{concept\_names}(\leq_n R) = \text{concept\_names}(\geq_n R) = \emptyset
  \]

- \( \text{contained\_concepts} : \mathcal{L}_G \rightarrow 2^{\mathcal{C}} \)
  \( \text{contained\_concepts}(C) \) computes all the concept expressions that appear in \( C \).

  \[
  \text{contained\_concepts}(C) = \{ D_1 | \exists D_2 : D_1 \in \text{prefix}(D_2) \land D_2 \in \text{suffix}(C) \}
  \]
• **direct generalizations**: $\mathcal{L}_G \rightarrow 2^\mathcal{C}_G$

  direct generalizations($C$) corresponds, for a given $T$ and a given $\mathcal{C}_N$, to the set of concepts in which only one concept name in $C$ has been substituted by one of its direct parents.

  \[
  \text{direct generalizations}(A) = \{A\} \cup \{B | \text{direct parent}_{T,\mathcal{C}_N}(A, B)\} \\
  \text{direct generalizations}(G^-) = \{G^-\} \\
  \text{direct generalizations}(C \cap D) = \{E \cap D | E \in \text{direct generalizations}(D)\} \cup \{C \cap D | E \in \text{direct generalizations}(C)\} \\
  \text{direct generalizations}(\exists R. C) = \{\exists R. D | D \in \text{direct generalizations}(C)\}
  \]

• **direct specializations**: $\mathcal{L}_G \rightarrow 2^\mathcal{C}_G$

  direct specializations($C$) corresponds, for a given $T$ and a given $\mathcal{C}_N$, to the set of concepts in which only one concept name in $C$ has been substituted by one of its direct parents.

  \[
  \text{direct specializations}(A) = \{A\} \cup \{B | \text{direct parent}_{T,\mathcal{C}_N}(A, B)\} \\
  \text{direct specializations}(G^-) = \{G^-\} \\
  \text{direct specializations}(C \cap D) = \{E \cap D | E \in \text{direct specializations}(D)\} \cup \{C \cap D | E \in \text{direct specializations}(C)\} \\
  \text{direct specializations}(\exists R. C) = \{\exists R. D | D \in \text{direct specializations}(C)\}
  \]

• **existential chain**: $\mathcal{L}R \rightarrow \mathcal{L}_G$

  existential chain($CR$) converts the sequence of simple roles in $R$ to a chain of existentials. So, existential chain($R_1 \circ R_2 \circ R_3 \ldots R_n$) = $\exists R_1 \exists R_2 \exists R_3 \ldots \exists R_n$

  \[
  \text{existential chain}(R) = \exists R. \top \\
  \text{existential chain}(R \circ CR) = \exists R. \text{existential chain}(CR)
  \]

• **generalizations**: $\mathcal{L}_G \rightarrow 2^\mathcal{C}_G$

  generalizations($C$) corresponds, for a given $T$ and a given $\mathcal{C}_N$, to the set of concepts in which none, one or more of the concept names in $C$ have been substituted by one of its parents.

  \[
  \text{generalizations}(A) = \{A\} \cup \{B | \text{parent}_{T,\mathcal{C}_N}(A, B)\} \\
  \text{generalizations}(G^-) = \{G^-\} \\
  \text{generalizations}(C \cap D) = \{E \cap D | E \in \text{generalizations}(D)\} \cup \{C \cap D | E \in \text{generalizations}(C)\} \\
  \text{generalizations}(\exists R. C) = \{\exists R. D | D \in \text{generalizations}(C)\}
  \]

• **infix**: $\mathcal{L}_G \rightarrow \mathcal{L}_G$

  infix($C$) is the most specific concept subsuming $C$ in which there does not appear the $\exists$ constructor.

  \[
  \text{infix}(G) = G \\
  \text{infix}(C \cap D) = \text{infix}(C) \cap \text{infix}(D) \\
  \text{infix}(\exists R. C) = \top
  \]

• **invert**: $\mathcal{L}_G \rightarrow \mathcal{L}_G$

  invert($C$) is the concept $C$ seen from the end of the chain of existentials. For example, invert($G_1 \cap \exists R_1 G_2 \cap \exists R_2 G_3 \cap \ldots \exists R_{n-1} G_n$) = $G_n \cap \exists R_{n-1}^- G_{n-1} \cap \exists R_{n-2}^- G_{n-2} \ldots \exists R_1^- G_1$.  

12
invert(G) = G
invert(G ∩ ∃R.C) = G_1 ∩ ∃R_1.G_2 ∩ ∃R_2.G_3 ∩ ... G_n ∩ ∃R^-G,
where G_1 ∩ ∃R_1.G_2 ∩ ∃R_2.G_3 ∩ ... G_n is invert(C)

• obtain_role : L_G → L_R
  obtain_role(C) gets the structure of the concept (stated in the existentials in C) as a composite role. For example obtain_role(G_1 ∩ ∃R_1.G_2 ∩ ∃R_2.G_3 ∩ ... G_n) = R_1 ∩ R_2 ∩ ... ∩ R_n-1.
obtain_role(∃R_1.C_1 ∩ ∃R_2.C_2) is undefined (as it would require role conjunction, not present in L_R). obtain_role(G) is also undefined.

  obtain_role(G ∩ ∃R.C) = R ∩ obtain_role(C)

• prefixes : L_G → 2^L_G
  prefixes(C) is the set of concepts that are prefixes of C. In general, if C = G_1 ∩ ∃R_1.G_2 ∩ ∃R_2.G_3 ∩ ... G_n, then prefixes(C) = \{ G_1, G_1 ∩ ∃R_1.G_2, ... \}.

  prefixes(G) = \{ G \}
  prefixes(C ∩ D) = specific(\{ E | E ∈ generalizations(prefixes(C)) \})
  \cup \{ E | E ∈ generalizations(prefixes(D)) \}
  prefixes(∃R.C) = \{ ⊤ \} ∪ \{ ∃R.D | D ∈ prefixes(C) \}

• same_as : L_G × L_G → L_G
  Given two concepts C and D that should correspond to existential chains, same_as(C, D) compute a concept stating that the last individual in the existential chain of C is the same than the last individual in the existential chain of D.

  same_as(G_1, G_2) = ⊤
  same_as(G_1, G_2 ∩ ∃R.C) = \{ obtain_role(∃R.C) \}
  same_as(G_1 ∩ ∃R_1, C_1.G_2 ∩ ∃R_2.C_2) = \{ obtain_role(∃R_1.C_1) ∩ obtain_role(∃R_2.C_2) \}

• specializations : L_G → 2^{L_G}
  specializations(C) corresponds, for a given T and a given CN, to the set of concepts in which none, one or more of the concept names in C have been substituted by one of its sons.

  specializations(A) = \{ A \} ∪ \{ B | parent_T.CN(B, A) \}
  specializations(G^-) = \{ G^- \}
  specializations(C ∩ D) = \{ E | E ∈ specializations(D) \}
  \cup \{ C | E ∈ specializations(C) \}
  specializations(∃R.C) = \{ ∃R.D | D ∈ specializations(C) \}

• suffixes : L_G → 2^{L_G}
  suffixes(C) is the set of concepts that are suffixes of C. In general, if C = G_1 ∩ ∃R_1.G_2 ∩ ∃R_2.G_3 ∩ ... G_n, then suffixes(C) = \{ G_n, G_{n-1} ∩ ∃R_{n-1}.G_n, ... \}.

  suffixes(G) = \{ G \}
  suffixes(C ∩ D) = specific(\{ E | E ∈ generalizations(suffixes(C)) \})
  \cup \{ E | E ∈ generalizations(suffixes(D)) \}
  suffixes(∃R.C) = \{ ∃R.C \} ∪ suffixes(C)

where A and B are concept names in CN, G, G_1 ... G_n are simple concepts in L_G, G^- is a simple concept with no concept names, C, D, E and F are L_G concepts, R is a role name or its inverse, and CR is a role expression.

In addition, we say that C is T-subsumed by D iff (C ⊆ D) ∨ (C ≤_1_0 D ∈ T).

13
5.2 Practical Information Quantification

In order to test whether an axiom $\Phi$ is interesting for a TBox $\mathcal{T}$, we need to compute the amount of information provided by $\mathcal{T} \cup \{\Phi\}$. In this section we will describe the computational procedures used by YAYA for this purpose.

The procedures explained here do not allow to compute the information provided by a whole TBox. Instead, they provide a measure of the amount of information provided by an axiom $\Phi_1$ given another axiom $\Phi_2$ and with respect to $\mathcal{M}$. So, we can compare $\Phi$ to all the axioms in a TBox $\mathcal{T}$; and detect some situations in which $\Phi$ does not provide new information (and thus, it is not interesting). Both procedures introduced in this section are applicable only in certain situations.

This two procedures do not guarantee that all non-interesting axioms will be flagged as non-interesting. But it has been empirically tested that a lot of non-interesting axioms can be detected, crucially reducing the search space.

5.2.1 Axiom Specialization

Given two axioms, $\Phi_1 = C_1 \leq_{\alpha_1, \sigma_1} D_1$, and $\Phi_2 = C_2 \leq_{\alpha_2, \sigma_2} D_2$, we say that $\Phi_1$ is a specialization of $\Phi_2$ iff $C_1 \leq C_2 \land D_2 \leq D_1$. The term specialization is used in the sense that, given an interpretation satisfying both axioms: (1) $\Phi_1$ provides information about a subset of the elements for which $\Phi_2$ provides information; and (2) The amount of information provided for $d \in C^T_1$ by $\Phi_1$ is a subset of the amount of information provided by $\Phi_2$.

The following points have been taken into account in order to define the measure:

1. Both axioms $\Phi_1$ and $\Phi_2$ are compared with respect to the information provided with respect to $D_1$. $D_1$ is not necessarily a concept name; so, according to section 4.3, we should compute the entropy in $D_1$ by accessing the concept names appearing in $D_1$ and the semantics of $D_1$. This is too cost effective and unnecessary as both axioms are compared with respect to the same concept $D_1$.

So, $D_1$ will be used as an atomic piece of knowledge. The boolean variables in section 4.3 will be defined here with respect to $D_1$.

2. We want our measure to be independent from the number of elements in $\Delta^T$. So, we will compute the entropy per domain element.

3. We want to provide a measure for the whole set of models of $\mathcal{T}$; not for a concrete model of it.

In order to achieve that, we first adapt Shannon entropy measure to a boolean variable:

$$H(p) = -p \cdot \log p - (1 - p) \cdot \log(1 - p)$$

Given $\Phi = C \leq_{\alpha, \sigma} D$, we know that $P(D|C)$ in $\text{Models}\{\Phi\}$ follows a normal distribution $N(\alpha, \sigma)$. So, we can “normalize” the previous definition in order to give a definition of the entropy measure for an axiom:

$$H(\Phi) = \int_{p=0}^{p=1} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}(\frac{p-\alpha}{\sigma})^2} \cdot H(p) dp$$

$H(\Phi)$ is the mean for all $\mathcal{I} \in \text{Models}(\mathcal{T})$ of the boolean variable $<d \in D, d \notin D^T>$ for a random $d \in C^T$.

Then, we define the relative amount of information provided by $\Phi_1 = C_1 \leq_{\alpha_1, \sigma_1} D_1$ given a generalization of it $\Phi_2 = C_2 \leq_{\alpha_2, \sigma_2} D_2$, and with respect to $\mathcal{M}$ as:

$$\text{RelInf}_{\mathcal{M}}(\Phi_1|\{\Phi_2\}) = H(\Phi_2) - \frac{H(\Phi_1) + H(\Phi_2)}{2}$$

where:
1. \( \Phi_3 = \neg C_1 \cap C_2 \leq_{\alpha_3, \sigma_3} D_1 \); and \( \alpha_3, \sigma_3 \) have been estimated from \( \mathcal{M} \).

2. \( \Phi'_2 = C_2 \leq_{\alpha'_2, \sigma'_2} D_2 \) would be the axiom estimated from \( \mathcal{M} \) with respect to \( C_2 \) and \( D_2 \) if \( |C_1^T| = |(\neg C_1)^T \cap C_2^T| \). \( \alpha'_2 \) and \( \sigma'_2 \) can be easily computed as:
   \[
   (a) \alpha'_2 = \frac{\alpha + \alpha_1}{2}
   
   (b) \sigma'_2 = \sqrt{\frac{\sigma^2 + \sigma_2^2 + (\alpha - \alpha'_1)^2 + (\alpha - \alpha'_2)^2}{2}}
   \]

   There are two important comments on the previous definition. First, a new axiom is introduced. This is done in order to take into account information provided by \( \Phi_1 \) about elements \( d \in C_2^T \) and \( d \notin C_1^T \). With respect to \( \Phi_2 \), all the elements \( d \in C_2^T \) are in \( D_1^T \) with probability \( \alpha_2 \) (as \( D_1^T \subseteq D_2^T \)). When \( \Phi_1 \) is taken into account, the probability that an element \( d \in C_1^T \) is also in \( D_1^T \) is modified. Because of that, and in order to maintain the probability distribution established by \( \Phi_2 \), the probability that an element \( d \in (\neg C_1)^T \cap C_2^T \) is also in \( D_1^T \) should also change.

   Adding axiom \( \Phi_1 \) to \( \{ \Phi_2 \} \) implicitly adds also axiom \( \Phi_3 \). Or what is the same, \( \Phi_3 \) can be deduced from \( \{ \Phi_1, \Phi_2 \} \) in our probabilistic DL system.

   Imagine we have induced from a model (our world for example), that (1) the 95 per cent of the birds fly; but (2) only the 20 per cent of the antarctic birds fly. Computing entropy reduction only from (1) and (2) is not enough. Doing so, we would conclude that having axioms (1) and (2) provides less information than having only axiom (1). Why? Because we have not taken into account that our knowledge about non-antarctic birds has been modified by axiom (2).

   Another important thing is that \( RelInfo \) gives the same weight to the set \( |C_1^T| \) and to the set \( |(\neg C_1)^T \cap C_2^T| \). The modification tries to better capture the different behaviour exhibited by \( C_1^T \) and \( (\neg C_1)^T \cap C_2^T \) without taking into account the known \( |P(C_1|C_2) \). This has been shown to work better for the performed experiments.

### 5.2.2 Axiom Decomposition

The most common case in a relational learning system that is not detected by \( RelInfo \) is what we have called \textit{axiom decomposition}. Given the following two axioms:

\[
\Phi_1 = G_1 \cap \exists R_1 G_2 \leq_{1,0} \exists R_1 G_2 \cap \exists R_2 G_3, \text{ and}
\]

\[
\Phi_2 = G_2 \leq_{1,0} \exists R_2 G_3
\]

We have that axiom \( \Phi_1 \) is redundant given axiom \( \Phi_2 \). But \( \Phi_1 \) is not a specialization of \( \Phi_2 \), so, \( RelInfo \) cannot be applied. Another measure named \textit{DecompInfo} is provided for this purpose. \textit{DecompInfo} states whether the information provided by an axiom \( \Phi_1 \) about a suffix of another axiom \( \Phi_2 \) makes this axiom redundant (provided that both concepts in \( \Phi_2 \) have a common prefix).

In order to compute whether axiom \( \Phi_1 \) is not interesting for \( T \) we look for axioms \( \Phi_2 \) in \( T \) that provide information about the suffix of \( \Phi_1 \) that has one less existential than \( \Phi_1 \). Then, parameters \( \alpha_1 \) and \( \sigma_1 \) are estimated for \( \Phi_1 \) from information provided by \( \Phi_2 \) and information provided by \( \mathcal{M} \) about \( \Phi_1 \) one-existential prefix (this information is directly computed from \( \mathcal{M} \)). Two estimated values are obtained: \( \alpha'_1 \) and \( \sigma'_1 \), meaning that axiom \( \Phi'_1 = C_1 \leq_{\alpha'_1, \sigma'_1} D_1 \) can be deduced from \( \Phi_2 \) plus prefix information.

Then, \textit{DecompInfo} computes the information provided by \( \Phi_1 \) given that we have \( \Phi'_1 \).

Going to the formulas, given the two axioms:

\[
\Phi_{hi1} = C_1 \leq_{\alpha_1, \sigma_1} D_1, \text{ and}
\]

\[
\Phi_{hi2} = C_2 \leq_{\alpha_2, \sigma_2} D_2
\]

where:

\[
C_1 = \text{in} \text{fix}(C_1) \cap \exists R_{cl_1} C_1^T \\
D_1 = \text{in} \text{fix}(D_1) \cap \exists R_{d_1} D_1^T \\
C_1 \leq_{\alpha_1, \sigma_1} (\exists R_{cl_1} C_1^T) \cap [R_{cl_1} \circ R_{cl_1}]^9
\]

\[9\] These three restrictions about \( \Phi_1 \) and \( \Phi_2 \) generalize the idea of same prefix for \( C_1 \) and \( D_1 \) used to introduce \textit{DecompInfo} to a more general idea of equivalent prefix.

15
$C'_1$ is $\mathcal{T}$-subsumed by $C_2$
$D_2$ is $\mathcal{T}$-subsumed by $D'_1$

We first estimate a set of axioms from $\mathcal{M}$. These axioms capture all information necessary about the $C_1$ prefix with one existential in order to perform the computations. Axioms estimated are: $\Phi_{\text{pre}_0} \ldots \Phi_{\text{pre}_n}$, where:

$$\Phi_{\text{pre}_i} = \text{inf}(C_1) \leq \alpha_{\text{pre}_i}, \sigma_{\text{pre}_i} \geq_i R_i \cdot \text{inf}(C'_1) \cap \leq_i R_i \cdot \text{inf}(C'_1), \text{ for } i = 1 \ldots n^{10}$$

where $\alpha_{\text{pre}_i}$ and $\sigma_{\text{pre}_i}$ have been estimated from $\mathcal{M}$; and $n$ is the greatest integer such that $\alpha_{\text{pre}_n} > 0$.

Then, we define:

$$\alpha'_1 = \sum_{i=1}^{n} (1 - (1 - \alpha_2)^i) \cdot \alpha_{\text{pre}_i} \quad (10)$$

$$\sigma'_1 = \sqrt[1-n]{\sum_{i=1}^{n} \alpha_{\text{pre}_i} \cdot \sigma_{\text{pre}_i}^2} \quad (11)$$

$$\Phi'_1 = C_1 \leq \alpha'_1, \sigma'_1, D_1 \quad (12)$$

And finally:

$$\text{DecompInfo}(\Phi_1, \Phi_2) = H(\Phi'_1) - H(\Phi_1) \quad (13)$$

5.3 Induction Rules

The set of induction rules provided by YAYA learning system has been designed to reach as much as possible interesting axioms from an initially reduced set of axioms (if $\mathcal{T}_0 = \emptyset$, the starting point is axiom $\top \leq_{1,0} \top$).

As it has been said before, there is no need to generalize the theory in any moment. Then, all induction rules correspond to ILP specialization operators.

Due to the big amount of rules\footnote{Although small is beautiful, the number of rules is necessarily big due to the big amount of $\mathcal{L}$ constructors and the differentiation of the antecedent and the consequent of an axiom (that results in providing different rules to treat the antecedent and the consequent).}, I have preferred to maintain them classified into different classes. In this classification the terms specialization and generalization have been used in a different sense of that used for ILP operators.

Specialization rules specialize the antecedent or the consequent of an axiom. They can be justified intuitively from an information theoretic point of view: given axiom $\Phi$ in $\mathcal{T}$, it is probable that $\Phi$ is interesting because $\mathcal{T}^+$ contains a specialization of it, say $\Phi^*$. Then, information in $\Phi^*$ is what makes $\Phi$ interesting.

Generalization rules generalize the antecedent or the consequent of an axiom. Given an axiom $\Phi^* \in \mathcal{T}^+$, we know that it is probable that given a specialization of it, $\Phi$, $\Phi$ is interesting for $\mathcal{T}^+ \setminus \{\Phi^*\}$. So, when an axiom is found to be interesting, it is possible that a generalization of it is still more interesting.

Besides these rules, YAYA provides some general exploration rules, and some environment updating rules. Before describing YAYA induction rules we define some functions that will be used in the rule definitions:
• \(\text{forbidden-concept}(C) = \exists D_1, D_2 | D_1 \in \text{contained-concepts}(C) \land D_2 \in \mathcal{F}^C \land D_1 \sqsubseteq D_2\)

• \(C \sqsubseteq_T D\), iff \(((C \sqsubseteq D) \land (\exists D' : D \sqsubseteq_{\alpha, \sigma} D' \in T) \lor (C \sqsubseteq_{1,0} D \in T)\)

• We say that concept \(C\) is hierarchically related to concept \(D\) in \(T\) iff \(C \sqsubseteq_T D \lor D \sqsubseteq_T C\)

In order to provide examples for a few rules, let’s introduce one of the domains for which YAYA has been tested: that of semantic parsing used in [Zelle and Mooney, 1993] (although the experiment was different from that in [Zelle and Mooney, 1993]). The domain consists of a set of over 1400 sentences of the form: agent action [patient] [accomp] [instrument]; where agent, patient, action, accomp, and instrument correspond to individuals instance of a word sense from the WordNet lexical ontology [Miller, 1990]\(^{12}\). Sentence examples are: “the girl ate the chicken with the fork” (where fork is the instrument), or “the man ate the chicken with the pasta” (where pasta is the accomp). Figure 2 shows the part of the taxonomy of WordNet senses that appear in sentences corresponding to the verb eat.

![Figure 2: A part of the WordNet sense taxonomy that is used in the semantic parsing experiment (italic names correspond to senses directly appearing in the sentences).](image)

Now let’s see the YAYA induction rules. For every rule, it is stated how the learning state is modified; and in which conditions the rule applies. The example domain is used only to provide examples for some of the rules.

**cut consequent** \((\rightarrow, \sqsubseteq, \leq, \land)\) \(< T, \mathcal{C}^N, \mathcal{R}^N, \mathcal{F}^C >\rightarrow< T \cup \{\Phi\}, \mathcal{C}^N, \mathcal{R}^N, \mathcal{F}^C >\)

1. \(C \leq_{\alpha, \sigma} D \in T\)
2. \(D = \text{append}(D', \exists R, G)\), where \(G\) is a simple concept.
3. \(\Phi = C \leq_M D'\)\(^{13}\) is interesting with respect to \(< T, M >\)\(^{14}\).
4. \(\neg \text{forbidden-concept}(D')\)

This is a consequent generalization rule.

**equivalence rule** \((\rightarrow, =)\) \(< T, \mathcal{C}^N, \mathcal{R}^N, \mathcal{F}^C >\rightarrow< T \cup \{\Phi\}, \mathcal{C}^N, \mathcal{R}^N, \mathcal{F}^C >\)

1. \(C \leq_{1,0} D \in T\)
2. \(\forall I \in M : D^I = C^I\)

\(^{12}\)WordNet senses are introduced as concept names and a set of axioms providing the WordNet taxonomy. The way in which the WordNet ontology has been integrated into the YAYA DL system is explained in [Alvarez, 2000a].

\(^{13}\)We will denote by \(C \leq_{\alpha, \sigma} D\) the axiom estimated from \(M\) for \(C\) and \(D\).

\(^{14}\)Computing whether an axiom is interesting for \(T\) with respect to \(M\) is approximated using the information theory measures described in section 5.2.
3. \( \Phi = D \leq_{1,0} C \) is interesting for \( < \mathcal{T}, \mathcal{M} > \)

**expand antecedent** \( (\rightarrow_{\exists \leq} ) < \mathcal{T}, \mathcal{CN}, \mathcal{RN}, \mathcal{FC} > \rightarrow < \mathcal{T} \cup \{ \Phi \}, \mathcal{CN}, \mathcal{RN}, \mathcal{FC} > \)

1. \( C \leq D' \in \mathcal{T} \) and \( D' \) is hierarchically related to \( D \) in \( \mathcal{T} \).
2. \( \text{append}(C_1, G) = C \), where \( G \) is a simple concept.
3. \( C_2 = (G \cap \exists R.A, \text{where } R \in \mathcal{RN} \) and \( A \in \mathcal{CN} \).
4. \( \Phi = \text{append}(C_1, C_2) \leq_{\mathcal{M}} D \) is interesting with respect to \( < \mathcal{T}, \mathcal{M} > \).
5. \( \neg \text{forbidden_concept} \left( \text{append}(C_1, C_2) \right) \)

This rule is an antecedent specialization rule. Given axiom \( C \leq D \), it specializes \( C \) by adding one more existential to the chain of existentials in \( D \).

**Example** Imagine \( \Phi = \text{eat-nI} \leq_{0.2,0.7} \exists \text{agent animal-nI} \in \mathcal{T} \). The application of this rule would produce axiom \( \Phi' = \text{eat-nI} \cap \exists \text{patient animal-nI} \leq_{1,0} \exists \text{agent animal-nI} \) (taken that \( \text{animal-nI} \in \mathcal{CN} \)).

**expand consequent** \( (\rightarrow_{\leq, \exists} ) < \mathcal{T}, \mathcal{CN}, \mathcal{RN}, \mathcal{FC} > \rightarrow < \mathcal{T} \cup \{ \Phi \}, \mathcal{CN}, \mathcal{RN}, \mathcal{FC} > \)

1. \( C' \leq_{\alpha, \sigma} D \in \mathcal{T} \) and \( C' \) is hierarchically related to \( C \) in \( \mathcal{T} \).
2. \( \text{append}(D_1, G) = C \), where \( G \) is a simple concept.
3. \( D_2 = (G \cap \exists R.A, \text{where } R \in \mathcal{RN} \) and \( A \in \mathcal{CN} \).
4. \( \Phi = C \leq_{\mathcal{M}} \text{append}(D_1, D_2) \) is interesting with respect to \( < \mathcal{T}, \mathcal{M} > \).
5. \( \neg \text{forbidden_concept} \left( \text{append}(D_1, D_2) \right) \)

This rule is a consequent specialization rule. Given axiom \( C \leq D \), it tries to specialize \( D \) by adding one more existential to the chain of existentials in \( D \).

**feature rule** \( (\rightarrow_{\leq_{\alpha, \sigma}} ) < \mathcal{T}, \mathcal{CN}, \mathcal{RN}, \mathcal{FC} > \rightarrow < \mathcal{T} \cup \{ C \leq_{\alpha, \sigma} D' \}, \mathcal{CN}, \mathcal{RN}, \mathcal{FC} > \)

1. \( C \leq_{\alpha, \sigma} D \in \mathcal{T} \)
2. \( \text{append}(C_1, G \cap \exists R.C_2) = C \), where \( \mathcal{T} \in \text{suffixes}(C_1) \land G \nexists_{\leq R} \)
3. \( D' = \text{append}(C_1, G \cap \leq \? R \cap \exists R.C_2) \)
4. \( \forall I \in \mathcal{M} : (C \cap D)' = (C \cap D') \)

This is a consequent specialization rule. Its use tries to specialize the consequent of an axiom \( \Phi \) by adding \( \leq_{1} \) restrictions to every one of the existentials taking part in \( \Phi \).

**forbid same-as long paths** \( (\rightarrow_{\mathcal{FC} \mid \emptyset} ) < \mathcal{T}, \mathcal{CN}, \mathcal{RN}, \mathcal{FC} > \rightarrow < \mathcal{T}, \mathcal{CN}, \mathcal{RN}, \mathcal{FC} \cup \{ CR \} > \)

1. \( \{ C \leq_{\alpha, \sigma} [R], [R] \leq_{\alpha, \sigma} C \} \cap \mathcal{T} \neq \emptyset \)
2. \( [R_1 \circ R_2] \) and \( R \) are equivalent, and the number of simple roles in \( R_1 \) is greater than the number of simple roles in \( R_2 \).
3. \( CR = \text{existentialChains}(R_1) \land CR \notin \mathcal{FC} \)
4. \( \Phi = CR \leq_{1,0} [R_1 \circ R_2] \) is consistent with respect to \( < \mathcal{M}, \mathcal{T} > \).
5. \( \mathcal{T}' = \mathcal{T} \cup \{ \Phi \} \) if \( \Phi \) is interesting with respect to \( < \mathcal{T}, \mathcal{M} > \); and \( \mathcal{T} = \mathcal{T}' \) otherwise.

This rule is an environment updating rule, and a general exploration rule at the same time. It checks whether the longer existential chain extracted from a loop construct is redundant. If it is, it adds the corresponding axiom to \( \mathcal{T} \) and the existential chain to \( \mathcal{FC} \).

**generalize antecedent** \( (\rightarrow_{\uparrow_{\leq}} ) < \mathcal{T}, \mathcal{CN}, \mathcal{RN}, \mathcal{FC} > \rightarrow < \mathcal{T} \cup \{ \Phi_2 \}, \mathcal{CN}, \mathcal{RN}, \mathcal{FC} > \)

1. \( \Phi_1 = C \leq_{\alpha, \sigma} D' \in \mathcal{T} \) and \( D' \) is hierarchically related to \( D \) in \( \mathcal{T} \).
2. \( C = \text{append}(C_1, A \cap G \cap C_2) \)
3. \( C' = \text{append}(C_1, B \cap G \cap C_2) \), where \( B \in \text{directParents}(A) \)
4. \( \Phi_2 = C' \leq_{\mathcal{M}} D \), and \( \Phi_2 \) is interesting for \( < \mathcal{T}, \mathcal{M} > \).

\(^{15}\)WordNet distinguishes between food senses and animal senses. Notice for example, that chicken-n2 in figure 2 corresponds to food; while chicken-n1 (not appearing in the figure) corresponds to animal.
This is a antecedent generalization rule. It is specially useful when large hierarchies of concept names are present in $\mathcal{T}_0$.

Example Imagine $\Phi = \text{person-nI} \leq_{1,0} \exists \text{agent}^{-} \text{eat} \in \mathcal{T}$. As there are other sentences in which the agent is an animal, the antecedent would be generalized to $\text{life-form-nI}$, obtaining:

$\Phi = \text{life-form-nI} \leq_{1,0} \exists \text{agent}^{-} \text{eat}$

generalize antecedent concept names ($\rightarrow_{\mathcal{N}^{*}}$)

$\langle \mathcal{T}, \mathcal{C}^{*}, \mathcal{R}^{*}, \mathcal{F}^{*} \rangle \rightarrow \langle \mathcal{T}', \mathcal{C}^{*}, \mathcal{R}^{*}, \mathcal{F}^{*} \rangle$

1. $\Phi_1 = C \leq_{a,\sigma} D' \in \mathcal{T}$ and $D'$ is hierarchically related to $D$ in $\mathcal{T}$.
2. $\Phi_2 = E \leq_{a,\sigma} D$ such that $E \in \text{generalizations}(C) \land E \not\in \mathcal{M} : (E \land D)^{2} = (C \land D)^{2}$
3. $\not\exists \Phi' = F \leq_{a,\sigma} D$ such that $F \in \text{generalizations}(E) \land F \not\in \mathcal{M} : (F \land D)^{2} = (C \land D)^{2}$
4. $\Phi_3 = E \subseteq_{\mathcal{M}} C$
5. $\mathcal{T}' = \mathcal{T} \cup \{ \Phi_2, \Phi_3 \}$
6. $\mathcal{C}^{*} = \mathcal{C} \cup \text{concept names}(E)$

This rule is an antecedent generalization rule and an environment updating rule at the same time (see the example in the specialize counterpart).

generalize consequent ($\rightarrow_{\mathcal{N}^{*}}$)

$\langle \mathcal{T}, \mathcal{C}^{*}, \mathcal{R}^{*}, \mathcal{F}^{*} \rangle \rightarrow \langle \mathcal{T}', \mathcal{C}^{*}, \mathcal{R}^{*}, \mathcal{F}^{*} \rangle$

1. $\Phi_1 = C' \leq_{a,\sigma} D \in \mathcal{T}$ and $C'$ is hierarchically related to $C$ in $\mathcal{T}$.
2. $D = \text{append}(D_1, A \cap G \cap D_2)$
3. $D' = \text{append}(D_1, B \cap G \cap D_2)$, where $B \in \text{direct parents}(A)$
4. $\Phi_2 = C \leq_{a,\sigma} D'$, and $\Phi_2$ is interesting for $\langle \mathcal{T}, \mathcal{M} \rangle$.

This is an antecedent generalization rule. It is specially useful when large hierarchies of concept names are present in $\mathcal{T}_0$.

generalize consequent concept names ($\rightarrow_{\mathcal{N}^{*}_{\downarrow}}$)

$\langle \mathcal{T}, \mathcal{C}^{*}, \mathcal{R}^{*}, \mathcal{F}^{*} \rangle \rightarrow \langle \mathcal{T}', \mathcal{C}^{*}, \mathcal{R}^{*}, \mathcal{F}^{*} \rangle$

1. $\Phi_1 = C' \leq_{a,\sigma} D \in \mathcal{T}$ and $C'$ is hierarchically related to $C$ in $\mathcal{T}$.
2. $\Phi_2 = C \leq_{a,\sigma} E$ such that $E \in \text{generalizations}(D) \land E \neq D \land \forall I \in \mathcal{M} : (E \land C)^{2} = (D \land C)^{2}$
3. $\not\exists \Phi' = C \leq_{a,\sigma} F$ such that $F \in \text{generalizations}(E) \land F \neq E \land \forall I \in \mathcal{M} : (F \land C)^{2} = (D \land C)^{2}$
4. $\Phi_3 = E \subseteq_{\mathcal{M}} D$
5. $\mathcal{T}' = \mathcal{T} \cup \{ \Phi_2, \Phi_3 \}$
6. $\mathcal{C}^{*} = \mathcal{C} \cup \text{concept names}(E)$

This rule is a consequent generalization rule and an environment updating rule at the same time.

more than rule ($\rightarrow_{\geq}$)

$\langle \mathcal{T}, \mathcal{C}^{*}, \mathcal{R}^{*}, \mathcal{F}^{*} \rangle \rightarrow \langle \mathcal{T} \cup \{ C \leq_{a,\sigma} D' \}, \mathcal{C}^{*}, \mathcal{R}^{*}, \mathcal{F}^{*} \rangle$

1. $C \leq_{a,\sigma} D \in \mathcal{T}$
2. $\text{append}(C_1, G \cap \exists R.C_2) = C$, where $\mathcal{T} \in \text{suffixes}(C_1) \land G \not\subseteq \exists_1 R$
3. $n > 1 \land n = \text{max}(m \forall I \in \mathcal{M} : (C \land D)^{2} = (C \land \text{append}(C_1, G \cap \geq_m R \cap \exists R.C_2))^2$
4. $D' = \text{append}(C_1, G \cap \geq_n R \cap \exists R.C_2)$

This is a consequent specialization rule. Its use tries to specialize the consequent of an axiom $\Phi$ by adding $\geq$ restrictions as restrictive as possible to every one of the existentials taking part in $\Phi$.

pre-expand rule ($\rightarrow_{\exists_{\geq}}$)

$\langle \mathcal{T}, \mathcal{C}^{*}, \mathcal{R}^{*}, \mathcal{F}^{*} \rangle \rightarrow \langle \mathcal{T} \cup \{ \Phi \}, \mathcal{C}^{*}, \mathcal{R}^{*}, \mathcal{F}^{*} \rangle$

1. $C' \leq_{a,\sigma} D \in \mathcal{T}$ and $C'$ is hierarchically related to $C$ in $\mathcal{T}$ or $C \leq D' \in \mathcal{T}$ and $D'$ is hierarchically related to $D$ in $\mathcal{T}$.
2. \( E = A \cap \exists R. \top \), where \( A \in \mathcal{C} \mathcal{N} \) and \( R \in \mathcal{R} \mathcal{N} \).
3. \( \Phi = \text{append}(E, C) \leq_M \text{append}(E, D) \) is interesting with respect to \( < \mathcal{T}, \mathcal{M} > \).
4. \( \neg \text{forbidden concept}(\text{append}(E, C)) \)

This rule is a general exploration rule. Given axiom \( \Phi^* \in \mathcal{T}^* \) such that \( \Phi^* = C^* \leq_{a^*, \sigma^*} D^* \); it is possible that axiom \( \Phi = C \leq_{a, \sigma} D \), where \( C \in \text{suffix}(C^*) \) and \( D \in \text{suffix}(D^*) \); is also interesting because of the subset of elements taking part in \( \Phi \) that also take part in \( \Phi^* \).

This rule tries to detect axioms in \( \mathcal{T}^* \) from axioms that correspond to a “suffix” if it.

**same-as consequent rule** \((\rightarrow_{\leq_{\mathcal{O}}}, \leq_{\mathcal{O}})\) \( < \mathcal{T}, \mathcal{C} \mathcal{N}, \mathcal{R} \mathcal{N}, \mathcal{F} \mathcal{C} > \rightarrow < \mathcal{T} \cup \{ \Phi \}, \mathcal{C} \mathcal{N}, \mathcal{R} \mathcal{N}, \mathcal{F} \mathcal{C} >

1. \( \{ C \leq_{a, \sigma} D_1, C' \leq_{a, \sigma} D_2 \} \in \mathcal{T} \land C' \leq_{\mathcal{O}} C \)
2. \( \Phi = C' \leq_{\mathcal{O}} \text{same-as}(D_1, D_2) \) is interesting with respect to \( < \mathcal{T}, \mathcal{M} > \).

This rule is a consequent specialization rule. It tries to find out relations between related objects. If two axioms have the same antecedent, it is possible that given an individual \( s \) holding both axioms, related individuals that make \( s \) satisfy the axioms are related among them.

**same-as axiom rule** \((\rightarrow_{\mathcal{O}^0}, \leq_{\mathcal{O}})\) \( < \mathcal{T}, \mathcal{C} \mathcal{N}, \mathcal{R} \mathcal{N}, \mathcal{F} \mathcal{C} > \rightarrow < \mathcal{T} \cup \{ \Phi \}, \mathcal{C} \mathcal{N}, \mathcal{R} \mathcal{N}, \mathcal{F} \mathcal{C} >

1. \( C' \leq_{a, \sigma} D \in \mathcal{T} \) and \( C' \) is hierarchically related to \( C \) in \( \mathcal{T} \).
2. \( \Phi = C \leq_{\mathcal{O}} \text{same-as}(C, D) \) is interesting with respect to \( < \mathcal{T}, \mathcal{M} > \).

This is also a consequent specialization rule.

**specialize antecedent** \((\rightarrow_{\mathcal{O}^0}, \leq_{\mathcal{O}})\) This rule is equivalent to the *generalize antecedent*; but instead of generalizing concept names, it specializes them.

**specialize antecedent concept names** \((\rightarrow_{\mathcal{O}^0}, \leq_{\mathcal{O}^0})\) This rule is equivalent to the *generalize antecedent concept names*; but instead of generalizing concept names, it specializes them.

**Example** Let’s take \( \Phi = \text{eat-n1} \cap \exists \text{patient animal-n1} \leq_{1, 0} \exists \text{agent animal-n1} \). As all eaten animals are eaten by carnivores, this rule will specialize the *animal-n1* concept name to *carnivore-n1*; adding axiom \( \text{eat-n1} \cap \exists \text{patient animal-n1} \leq_{1, 0} \exists \text{agent carnivore-n1} \), and adding *carnivore-n1* to *C*.

**specialize axiom** \((\rightarrow_{\mathcal{O}^0}, \leq_{\mathcal{O}})\) \( < \mathcal{T}, \mathcal{C} \mathcal{N}, \mathcal{R} \mathcal{N}, \mathcal{F} \mathcal{C} > \rightarrow < \mathcal{T} \cup \{ \Phi_2 \}, \mathcal{C} \mathcal{N}, \mathcal{R} \mathcal{N}, \mathcal{F} \mathcal{C} >

1. \( \Phi_1 = C' \leq D \in \mathcal{T} \) and \( C' \) is hierarchically related to \( C \) in \( \mathcal{T} \) or \( \Phi_1 = C \leq D' \in \mathcal{T} \) and \( D' \) is hierarchically related to \( D \) in \( \mathcal{T} \).
2. \( C = \text{append}(C_1, A_C \cap G \cap C_2) \)
3. \( D = \text{append}(D_1, A_D \cap G \cap D_2) \)
4. \( C'' = \text{append}(C_1, B_C \cap G \cap C_2) \), where \( B_C \in \text{generalizations}(A_C) \)
5. \( D'' = \text{append}(D_1, B_D \cap G \cap D_2) \), where \( B_D \in \text{generalizations}(A_D) \)
6. \( \Phi_2 = C'' \leq_M D'' \) is interesting for \( < \mathcal{T}, \mathcal{M} > \)

This is a specialization rule. Sometimes, specializing only the antecedent or the consequent in a given axiom is not enough to detect an interesting specialization of it. This rule tries to fix this problem by specializing both at the same time.

**specialize consequent** \((\rightarrow_{\mathcal{O}^0}, \leq_{\mathcal{O}})\) This rule is equivalent to the *generalize consequent*; but instead of generalizing concept names, it specializes them.

**specialize consequent concept names** \((\rightarrow_{\mathcal{O}^0}, \leq_{\mathcal{O}^0})\) This rule is equivalent to the *generalize consequent concept names*; but instead of generalizing concept names, it specializes them.

In order to try to minimize the search performed by the induction rules application; rule applications are performed in a given order that depends on the rule being applied:

---

\(^{16}\)In fact, this rule is not a consequent specialization rule since \( D \) is not necessarily a specialization of \( D_1 \) or \( D_2 \). Its qualification as a specialization rule is because the conceptual idea behind the rule is that of a specialization one. The reason because it is not formally a specialization rule comes from the poor expressivity of the loop construct, that does not allow to qualify individuals taking part in the cycle. This also applies to the *same-as axiom rule*. 
1. cut consequent rule is applied if possible.
2. forbid same-as long path rule is applied if possible.
3. Applications involving generalization rules are applied.
4. The rest of applications are performed.

The order in which induction rule applications with the same priority are applied is established by a set of heuristics that combine several factors such as: minimizing the complexity of the axiom(s) that take part in the application; maximizing the information provided by the axiom(s) taking part in the application; and maximizing the amount of information that possibly could be provided by the axioms resulting from the rule application.

6 Experiments

A set of experiments have been done for several domains: semantic parsing [McClelland and Kawamoto, 1986, Zelle and Mooney, 1993], family relationships [Quinlan, 1990], and king-rook-king illegal chess positions [Quinlan, 1990].

The experiments have shown promising results, and helped to design and tune the set of induction rules. The results described below were obtained with a previous version of YAYA [Alvarez, 1998] in which an only ABox is used as the training set and probabilistic axioms only have a probability attached to them.

6.1 Family Relationships

The classical two-family benchmark used in [Quinlan, 1990] has been tested with YAYA. The set of role names used are: mother-of, father-of, wife-of, son-of, daughter-of, and their inverses. Axioms defining a role from the rest of them are learned for all the role names. For example, axioms for the mother-of role are as follows:

\[
\exists \text{wife} \circ o f \exists \text{son} \circ o f \cdots \text{T} \leq 1 [\text{mother} \circ o f \circ \text{son} \circ o f \circ \text{wife} \circ o f] \\
\exists \text{mother} \circ o f \circ \text{T} \leq 1 [\text{mother} \circ o f \circ \text{son} \circ o f \circ \text{wife} \circ o f] \\
\exists \text{wife} \circ o f \circ \exists \text{daughter} \circ o f \circ \text{T} \leq 1 [\text{mother} \circ o f \circ \text{daughter} \circ o f \circ \text{wife} \circ o f] \\
\exists \text{mother} \circ o f \circ \text{T} \leq 1 [\text{mother} \circ o f \circ \text{daughter} \circ o f \circ \text{wife} \circ o f]
\]

6.2 KRK Illegal Configurations

A set of tests for KRK configurations has been done. For every test, a set of 300 KRK random configurations have been automatically generated using a Lisp program. The used representation is equivalent to the one used for ILP predicate-style representation. Two concept names: legal and illegal have been introduced. All legal configurations are instances of legal; and illegal ones are instances of illegal. For every configuration, 17 individuals are introduced: 1 for the configuration, 8 for the table rows, and 8 for the table columns. Rows and columns are related through the $lt$ and $adj$ roles. Six more role names are introduced in order to define the positions of the rook and both kings.

YAYA learning procedure has been applied. The following axioms (among others) have been learned for all the performed tests:

\[
[CWR \circ CBK^-] \leq 0.94 \text{ illegal} \\
[RWR \circ RBK^-] \leq 0.94 \text{ illegal} \\
[CWK \circ ADJ \circ CBK^-] \cap [RWK \circ ADJ \circ RBK^-] \leq 1 \text{ illegal}
\]

These axioms provide a definition of illegal that covers the 99 per cent of illegal randomly generated configurations.
6.3 Semantic Parsing

A variation of the experiment described in [McClelland and Kawamoto, 1986, Zelle and Mooney, 1993] has been performed. A subset of about 250 simple sentences have been used as the test set. These sentences correspond to simple sentences as: the man eats the pasta with the fork. Words in the sentences have been related to word senses in the WordNet ontology. WordNet word senses are represented in YAYA as concept names and a set of axioms providing the WordNet taxonomy.

The goal of the learning experiment was to learn selectional restrictions among different senses taking part in a sentence. For example, learning that when an animal is eaten, the agent should be a carnivore. YAYA was able to learn all the restrictions in the test sentences. The set of sentences contained about 15 different sentence patterns with around 60 different WordNet senses.

References


\footnote{The complete text contains about 1400 sentences. Only those for the verb eat and move have been used.}


