Preferential Description Logics

Laura Giordano[1], Valentina Gliozzi[2], Nicola Olivetti[3], and Gian Luca Pozzato[2]

[1] Dip. di Informatica - Univ. Piemonte O. "A. Avogadro" - laura@mfn.unipmn.it
[2] Dip. di Informatica - Università di Torino - {gliozzi,pozzato}@di.unito.it
[3] LSIS-UMR CNRS 6168 Univ. "P. Cézanne" - nicola.olivetti@univ-cezanne.fr

Abstract. We extend the Description Logic $\mathcal{ALC}$ with a "typicality" operator $\mathbf{T}$ that allows us to reason about the prototypical properties and inheritance with exceptions. The resulting logic is called $\mathcal{ALC} + \mathbf{T}$. The typicality operator is intended to select the "most normal" or "most typical" instances of a concept. In our framework, knowledge bases may then contain, in addition to ordinary ABoxes and TBoxes, subsumption relations of the form "$\mathbf{T}(C)$ is subsumed by P", expressing that typical C-members have the property P. The semantics of a typicality operator is defined by a set of postulates that are strongly related to Kraus-Lehmann-Magidor axioms of preferential logic $\mathbf{P}$. We first show that $\mathbf{T}$ enjoys a simple semantics provided by ordinary structures equipped by a preference relation. This allows us to obtain a modal interpretation of the typicality operator. Using such a modal interpretation, we present a tableau calculus for deciding satisfiability of $\mathcal{ALC} + \mathbf{T}$ knowledge bases. Our calculus gives a nondeterministic-exponential time decision procedure for satisfiability of $\mathcal{ALC} + \mathbf{T}$. We then extend $\mathcal{ALC} + \mathbf{T}$ knowledge bases by a nonmonotonic completion that allows inferring defeasible properties of specific concept instances[1].

1 Introduction

The family of description logics (DLs) is one of the most important formalisms of knowledge representation. DLs are reminiscent of the early semantic networks and of frame-based systems. They offer two key advantages: a well-defined semantics based on first-order logic and a good trade-off between expressivity and complexity. DLs have been successfully implemented by a range of systems and they are at the base of languages for the semantic web such as OWL. A DL knowledge base (KB) comprises two components: (i) the TBox, containing the definition of concepts (and possibly roles), and a specification of inclusions relations among them, and (ii) the ABox containing instances of concepts and roles, in other words, properties and relations of individuals. Since the very objective of the TBox is to build a taxonomy of concepts, the need of representing prototypical properties and of reasoning about defeasible inheritance of such properties easily arises. The traditional approach is to handle defeasible inheritance by integrating some kind of nonmonotonic reasoning mechanism. This has

[1] This research has been partially supported by "*Progetto Lagrange - Fondazione CRT*" and by the project "*MIUR PRIN05: Specification and verification of agent interaction protocols*".

led to study nonmonotonic extensions of DLs [1–3, 5–7, 12]. However, finding a suitable nonmonotonic extension for inheritance reasoning with exceptions is far from obvious. Let us put forward some desiderata for such an extension:

1. The (nonmonotonic) extension must have a clear semantics and should be based on the same semantics as the underlying monotonic DL.
2. The extension should allow to specify prototypical properties in a natural and direct way.
3. The extension must be decidable, if so is the underlying monotonic DL and, possibly, computationally effective.

The nonmonotonic extensions proposed in the literature do not seem to fully satisfy all the above desiderata.

[1] proposes the extension of DL with Reiter's default logic. However, the same authors have pointed out that this integration may lead to both semantical and computational difficulties. Indeed, the unsatisfactory treatment of open defaults via Skolemization may lead to an undecidable default consequence relation. For this reason, [1] proposes a restricted semantics for open default theories, in which default rules are only applied to individuals explicitly mentioned in the ABox. Furthermore, Reiter's default logic does not provide a direct way of modeling inheritance with exceptions. This has motivated the study of extensions of DLs with prioritized defaults [12, 2].

A more general approach is undertaken in [6], where it is proposed an extension of DL with two epistemic operators. This extension allows one to encode Reiter's default logic as well as to express epistemic concepts and procedural rules. However, this extension has a rather complicated modal semantics, so that the integration with the existing systems requires significant changes to the standard semantics of DLs.

In [3] the authors propose an extension of DL with circumscription. One of motivating applications of circumscription is indeed to express prototypical properties with exceptions, and this is done by introducing "abnormality" predicates, whose extension is minimized. The authors provide decidability and complexity results based on theoretical analysis. However, they do not provide a calculus for their logic. Moreover, the use of circumscription to model inheritance with exceptions is not that straightforward, as we remark below.

In this work, we propose a novel approach to defeasible inheritance reasoning based on the typicality operator $\mathbf{T}$. The intended meaning is that, for any concept C, $\mathbf{T}(C)$ singles out the instances of C that are considered as "typical" or "normal". Thus assertions as "normally students do not pay taxes", or "typically users do not have access to confidential files" [3] are represented by $\mathbf{T}(\mathit{Student}) \sqsubseteq \neg \mathit{TaxPayer}$ and $\mathbf{T}(\mathit{User}) \sqsubseteq \neg\exists \mathit{hasAccess}.\mathit{ConfidentialFile}$.

Before entering in the technical details, let us sketch how we intend to use the typicality operator and what kind of inferential services we expect to profit. We assume that a KB comprises, in addition to the standard TBox and ABox, a set of assertions of the type $\mathbf{T}(C) \sqsubseteq D$ where D is a concept not mentioning $\mathbf{T}$. The reasoning system should be able to infer or propagate prototypical properties

of the concepts specified in the TBox, then to ascribe defeasible properties to individuals. For instance, let the KB contain:

$\mathbf{T}(Student) \sqsubseteq \neg TaxPayer$
$\mathbf{T}(Student \sqcap Worker) \sqsubseteq TaxPayer$
$\mathbf{T}(Student \sqcap Worker \sqcap \exists HasChild.\top) \sqsubseteq \neg TaxPayer$
$\mathbf{T}(Unemployed) \sqsubseteq \neg TaxPayer$

corresponding to the assertions: normally a student does not pay taxes, normally a working student pays taxes, but normally a working student having children does not pay taxes (because he is discharged by the government) etc...Observe that, if the same properties were expressed by ordinary inclusions, such as $Student \sqsubseteq \neg TaxPayer$ etc...we would simply get that there are not working students and so on, thus the KB would collapse. This collapse is avoided as we do not assume that $\mathbf{T}$ is monotonic, that is to say $C \sqsubseteq D$ does not imply $\mathbf{T}(C) \sqsubseteq \mathbf{T}(D)$. To continue with the example, if the TBox contains $PersonWithNoIncome \equiv Student \sqcup Unemployed$, then the system should be able to infer $\mathbf{T}(PersonWithNoIncome) \sqsubseteq \neg TaxPayer$. Suppose next that the ABox contains alternatively the following facts about $john$:

1. $student(john)$
2. $student(john), worker(john)$
3. $student(john), worker(john), \exists HasChild.\top(john)$

Then the reasoning system should be able to infer the expected (defeasible) conclusion about $john$ in each case: 1. $\neg TaxPayer(john)$, 2. $TaxPayer(john)$, 3. $\neg TaxPayer(john)$. Observe that setting up a similar specification of the KB by using default logic or circumscription is not that simple: with default logic [1, 2, 5–7, 12], one has to specify a priority on default application (or one has to find a smart encoding of defaults giving priority to more specific information); with circumscription [3], one has to introduce abnormality predicates, and then establish which predicates are minimized, fixed, or variable, and finally for the minimized ones what is their priority wrt minimization (a total or a partial order). As a further step, the system should be able to infer (defeasible) properties also of individuals implicitly introduced by existential restrictions, for instance, if the ABox further contains $\exists HasChild.Student(jack)$, it should conclude (defeasibly) $\exists HasChild.\neg TaxPayer(jack)$.

Given the nonmonotonic character of the $\mathbf{T}$ operator, there is a difficulty with handling irrelevant information, for instance, given the KB as above, one should be able to infer as well:

$\mathbf{T}(Student \sqcap SportLover) \sqsubseteq \neg TaxPayer$
$\mathbf{T}(Student \sqcap Worker \sqcap SportLover) \sqsubseteq TaxPayer$

as $SportLover$ is irrelevant with respect to being a TaxPayer or not. For the same reason, the conclusion about $john$ being a TaxPayer or not should not be influenced by the addition of $SportLover(john)$ to the ABox. We refer to this problem as the problem of Irrelevance.

In this paper we lay down the base of an extension of DL with a typicality operator. Our starting point is a monotonic extension of the basic $\mathcal{ALC}$ with the **T** operator. The operator is supposed to satisfy a set of postulates that are essentially a reformulation of Kraus, Lehmann, and Magidor (KLM) axioms of preferential logic, namely the assertion $\mathbf{T}(C) \sqsubseteq P$ is equivalent to the conditional assertion $C \mathrel{|\!\sim} P$ of KLM preferential logic **P**. It turns out that the semantics of the typicality operator can be equivalently specified by considering a preference relation (a strict partial order) on individuals: the typical members of a concept C are just the most preferred individuals, or "most normal", of C according to the preference relation. The preference relation is the only additional ingredient that we need in our semantics. We assume that "most normal" members of a concept C always exist, whenever the concept C is non-empty. This assumption corresponds to the *Smoothness Condition* of KLM logics, or the well-known *Limit Assumption* in conditional logics. Taking advantage of this semantic setting, we can give a modal interpretation to the typicality operator: the modal operator $\Box$ has intuitively the same properties as in Gödel-Löb modal logic G of arithmetic provability. We then define a tableau system for this extension based on this modal interpretation, thereby obtaining a decision procedure and un upper complexity bound (NEXPTIME). In future research we will further consider whether this bound is optimal or not.

These are the main results of the paper, however the monotonic extension is not enough to perform inheritance reasoning of the kind described above: (i) we need a way of inferring defeasible properties of individuals, (ii) we need a way of handling Irrelevance.

In the paper we deal with (i) by defining a completion of an ABox: the idea is that each individual is assumed to be a typical member of the most specific concept to which it belongs. Such a completion allows to perform inferences as the ones 1.,2.,3. above. The paper outlines how we intend to cope with typicality of all instances and with Irrelevance. In particular, dealing with Irrelevance (ii) requires a nonmonotonic mechanism. The idea is to complete a KB with a set of default rules. The default rules are not used to express defeasible properties of concepts (as in default extension of DLs), but to propagate defeasible properties of a concept to its subsumed concepts, e.g. to infer $\mathbf{T}(\mathit{Student} \sqcap \mathit{SportLover}) \sqsubseteq \neg \mathit{TaxPayer}$ from $\mathbf{T}(\mathit{Student}) \sqsubseteq \neg \mathit{TaxPayer}$. Thus in our approach the nonmonotonic mechanism is only needed to handle Irrelevance, whose treatment by means of default rules will be the object of future work.

2 The logic $\mathcal{ALC}$ + T: the typicality operator T

We consider an alphabet of concept names $\mathcal{C}$, of role names $\mathcal{R}$, and of individuals $\mathcal{O}$. The language $\mathcal{L}$ of the logic $\mathcal{ALC}$ + **T** is defined by distinguishing *concepts* and *extended concepts* as follows: (Concepts) $A \in \mathcal{C}$ and $\top$ are *concepts* of $\mathcal{L}$; if $C, D \in \mathcal{L}$ and $R \in \mathcal{R}$, then $C \sqcap D, C \sqcup D, \neg C, \forall R.C, \exists R.C$ are *concepts* of $\mathcal{L}$. (Extended concepts) if C is a concept, then C and $\mathbf{T}(C)$ are *extended concepts*, and all the boolean combinations of extended concepts are extended concepts

of $\mathcal{L}$. A knowledge base is a pair (TBox,ABox). TBox contains subsumptions $C \sqsubseteq D$, where $C \in \mathcal{L}$ is either a concept or an extended concept $\mathbf{T}(C')$, and $D \in \mathcal{L}$ is a concept. ABox contains expressions of the form $C(a)$ and aRb where $C \in \mathcal{L}$ is an extended concept, $R \in \mathcal{R}$, and $a, b \in \mathcal{O}$.

In order to provide a semantics to the operator $\mathbf{T}$, we extend the definition of a model used in "standard" terminological logic $\mathcal{ALC}$:

Definition 1 (Semantics of T with selection function). *A model is any structure $\langle \Delta, I, f_{\mathbf{T}} \rangle$, where: Δ is the domain; I is the extension function that maps each extended concept C to $C^I \subseteq \Delta$, and each role R to a $R^I \subseteq \Delta^I \times \Delta^I$. I is defined in the usual way (as for $\mathcal{ALC}$) and, in addition, $(\mathbf{T}(C))^I = f_{\mathbf{T}}(C^I)$. $f_{\mathbf{T}} : Pow(\Delta) \rightarrow Pow(\Delta)$ is a function satisfying the following properties:*

$(f_{\mathbf{T}} - 1)$ $f_{\mathbf{T}}(S) \subseteq S$ $\quad$ $(f_{\mathbf{T}} - 2)$ if $S \neq \emptyset$, then also $f_{\mathbf{T}}(S) \neq \emptyset$

$(f_{\mathbf{T}} - 3)$ if $f_{\mathbf{T}}(S) \subseteq R$, then $f_{\mathbf{T}}(S) = f_{\mathbf{T}}(S \cap R)$ $\quad$ $(f_{\mathbf{T}} - 4)$ $f_{\mathbf{T}}(\bigcup S_i) \subseteq \bigcup f_{\mathbf{T}}(S_i)$

$(f_{\mathbf{T}} - 5)$ $\bigcap f_{\mathbf{T}}(S_i) \subseteq f_{\mathbf{T}}(\bigcup S_i)$

Intuitively, given the extension of some concept C, $f_{\mathbf{T}}$ selects the *typical* instances of C. $(f_{\mathbf{T}} - 1)$ requests that typical elements of S belong to S. $(f_{\mathbf{T}} - 2)$ requests that if there are elements in S, then there are also *typical* such elements. The next properties constraint the behavior of $f_{\mathbf{T}}$ wrt $\cap$ and $\cup$ in such a way that they do not entail monotonicity. According to $(f_{\mathbf{T}} - 3)$, if the typical elements of S are in R, then they coincide with the typical elements of $S \cap R$, thus expressing a weak form of monotonicity (namely *cautious monotonicity*). $(f_{\mathbf{T}} - 4)$ corresponds to one direction of the equivalence $f_{\mathbf{T}}(\bigcup S_i) = \bigcup f_{\mathbf{T}}(S_i)$, so that it does not entail monotonicity. Similar considerations apply to the equation $f_{\mathbf{T}}(\bigcap S_i) = \bigcap f_{\mathbf{T}}(S_i)$, of which only the inclusion $\bigcap f_{\mathbf{T}}(S_i) \subseteq f_{\mathbf{T}}(\bigcap S_i)$ is derivable. $(f_{\mathbf{T}} - 5)$ is a further constraint on the behavior of $f_{\mathbf{T}}$ wrt arbitrary unions and intersections; it would be derivable if $f_{\mathbf{T}}$ were monotonic. We can prove the following proposition:

Proposition 1. $f_{\mathbf{T}}(S \cup R) \cap S \subseteq f_{\mathbf{T}}(S)$

We can give an alternative semantics for $\mathbf{T}$ based on a preference relation. The idea is that there is a global preference relation among individuals and that the typical members of a concept C, i.e. selected by $f_{\mathbf{T}}(C^I)$, are the minimal elements of C wrt this preference relation. Observe that this notion is global, that is to say, it does not compare individuals wrt a specific concept (something like y is more typical than x wrt concept C). In this framework, an object $x \in \Delta$ is a *typical instance* of some concept C, if $x \in C^I$ and there is no C-element in Δ *more typical* than x. The typicality preference relation is partial since it is not always possible to establish which object is more typical than which other. The following definition is needed before we provide the Representation Theorem.

Definition 2. *Given a relation $<$, which is a strict partial order (i.e. an irreflexive and transitive relation) over a domain Δ, for all $S \subseteq \Delta$, we define $Min_<(S) = \{x : x \in S \text{ and } \nexists y \in S \text{ s.t. } y < x\}$. We say that $<$ satisfies the* Smoothness Condition *iff for all $S \subseteq \Delta$, for all $x \in S$, either $x \in Min_<(S)$ or $\exists y \in Min_<(S)$ such that $y < x$.*

We are now ready to prove the Representation Theorem below, showing that given a model with a selection function, we can define *on the same domain* a preference relation $<$ such that, for all $S \subseteq \Delta$, $f_{\mathbf{T}}(S) = Min_<(S)$. Notice that, as a difference wrt related results (Theorem 3 in [11]), the relation is defined on the same domain Δ of $f_{\mathbf{T}}$. On the other hand, if $<$ is a preference relation satisfying the Smoothness Condition, then the operator defined as $f_{\mathbf{T}}(S) = Min_<(S)$ satisfies the postulates of Definition 1.

Theorem 1 (Representation Theorem). *Given any model $\langle \Delta, I, f_{\mathbf{T}} \rangle$, $f_{\mathbf{T}}$ satisfies postulates $(f_{\mathbf{T}} - 1)$ to $(f_{\mathbf{T}} - 5)$ above iff it is possible to define on Δ a strict partial order $<$, satisfying the Smoothness Condition, such that for all $S \subseteq \Delta$, $f_{\mathbf{T}}(S) = Min_<(S)$.*

Proof. ("*Only if*" direction) Given $f_{\mathbf{T}}$ satisfying postulates $(f_{\mathbf{T}} - 1)$ to $(f_{\mathbf{T}} - 5)$, we define $<$ as follows: for all $x, y \in \Delta$, we let $x < y$ if $\forall S \subseteq \Delta$, if $y \in f_{\mathbf{T}}(S)$ then $x \notin S$, and $\exists R \subseteq \Delta$ such that $S \subset R$ and $x \in f_{\mathbf{T}}(R)$. We prove that:

1. $<$ is irreflexive. Easily follows by the definition of $<$.
2. $<$ is transitive. Let (a) $x < y$ and (b) $y < z$. Let $z \in f_{\mathbf{T}}(S)$ for some S, then by definition of $<$, $y \notin S$, and $\exists R$ s.t. $S \subset R$ and $y \in f_{\mathbf{T}}(R)$. Furthermore, $x \notin R$ and $\exists Q : R \subset Q$ and $x \in f_{\mathbf{T}}(Q)$. From this we can conclude that $x \notin S$ (otherwise $x \in R$), and $S \subset Q$, hence $x < z$.
3. $f_{\mathbf{T}}(S) \subseteq Min_<(S)$. Let $x \in f_{\mathbf{T}}(S)$. Suppose $x \notin Min_<(S)$, i.e. for some $y \in S$, $y < x$. By definition of $<$, $y \notin S$, contradiction, hence $x \in Min_<(S)$.
4. $Min_<(S) \subseteq f_{\mathbf{T}}(S)$. Let $x \in Min_<(S)$. Then $x \in S$, i.e. $S \neq \emptyset$. By $(f_{\mathbf{T}} - 2)$, $f_{\mathbf{T}}(S) \neq \emptyset$. Suppose $x \notin f_{\mathbf{T}}(S)$. Consider $\bigcup R_i$ for all $R_i \subseteq \Delta$ s.t. $x \in f_{\mathbf{T}}(R_i)$. By $(f_{\mathbf{T}} - 5)$, we have $x \in f_{\mathbf{T}}(\bigcup R_i)$.
   Consider now $f_{\mathbf{T}}(\bigcup R_i \cup S)$. We can easily show that $f_{\mathbf{T}}(\bigcup R_i \cup S) \not\subseteq \bigcup R_i$ (otherwise, by $(f_{\mathbf{T}} - 3)$ $f_{\mathbf{T}}(\bigcup R_i \cup S) = f_{\mathbf{T}}(\bigcup R_i)$, and by Proposition 1, $f_{\mathbf{T}}(\bigcup R_i) \cap S \subseteq f_{\mathbf{T}}(S)$, which contradicts the fact that $x \in f_{\mathbf{T}}(\bigcup R_i)$ but $x \notin f_{\mathbf{T}}(S)$). Consider hence $y \in f_{\mathbf{T}}(\bigcup R_i \cup S)$ s.t. $y \notin \bigcup R_i$. By definition of $<$, $y < x$. Furthermore, by $(f_{\mathbf{T}} - 1)$ $y \in S$ (since $y \in \bigcup R_i \cup S$ and $y \notin \bigcup R_i$). It follows that $x \notin Min_<(S)$, contradiction, hence $Min_<(S) \subseteq f_{\mathbf{T}}(S)$.
5. $<$ satisfies the Smoothness Condition. Let $S \neq \emptyset$ and $x \in S$. If $x \in f_{\mathbf{T}}(S)$ then by point 3 we have $x \in Min_<(S)$. If $x \notin f_{\mathbf{T}}(S)$ we can reason as for point 4 to conclude that there is $y \in f_{\mathbf{T}}(\bigcup R_i \cup S)$ s.t. $y \notin \bigcup R_i$ (hence $y \in S$), and $y < x$. By Proposition 1, we have $y \in f_{\mathbf{T}}(S)$, hence by point 3 we conclude $y \in Min_<(S)$.

The points above allow us to conclude.
("*If*" direction) Given a strict partial order $<$ satisfying the Smoothness Condition, we can define $f_{\mathbf{T}} : Pow(\Delta) \to Pow(\Delta)$ by letting $f_{\mathbf{T}}(S) = Min_<(S)$. It can be easily shown that $f_{\mathbf{T}}$ satisfies postulates $(f_{\mathbf{T}} - 1)$ to $(f_{\mathbf{T}} - 5)$. The proof is omitted due to space limitations. ■

Having the above Representation System, from now on, we will refer to the following semantics for $\mathcal{ALC} + \mathbf{T}$:

Definition 3 (Semantics of $\mathcal{ALC}+\mathbf{T}$). *A model $\mathcal{M}$ is any structure $\langle \Delta, <, I \rangle$, where Δ and I are defined as in Definition 1, and $<$ is a strict partial order over Δ satisfying the Smoothness Condition (see Definition 2 above). As a difference wrt Definition 1, the semantics of the* $\mathbf{T}$ *operator is: $(\mathbf{T}(C))^I = Min_<(C^I)$. For concepts (built from operators of $\mathcal{ALC}$), C^I is defined in the usual way.*

Definition 4 (Model satisfying a Knowledge Base). *Consider a model $\mathcal{M}$, as defined in Definition 3. We extend I so that it assigns to each individual a of $\mathcal{O}$ an element a^I of the domain Δ. Given a KB (TBox,ABox), we say that:*

- *$\mathcal{M}$ satisfies TBox if for all inclusions $C \sqsubseteq D$ in TBox, and all elements $x \in \Delta$, if $x \in C^I$ then $x \in D^I$.*
- *$\mathcal{M}$ satisfies ABox if: (i) for all $C(a)$ in ABox, we have that $a^I \in C^I$, (ii) for all aRb in ABox, we have that $(a^I, b^I) \in R^I$.*

$\mathcal{M}$ satisfies a knowledge base if it satisfies both its TBox and its ABox.

Notice that the meaning of $\mathbf{T}$ can be split into two parts: for any object x of the domain Δ, $x \in (\mathbf{T}(C))^I$ just in case (i) $x \in C^I$, and (ii) there is no $y \in C^I$ such that $y < x$. In order to isolate the second part of the meaning of $\mathbf{T}$ (for the purpose of the calculus that we will present in section 3) we introduce a new modality $\Box$ whose interpretation in $\mathcal{M}$ is defined as follows.

Definition 5. $(\Box C)^I = \{x \in \Delta \mid$ *for every* $y \in \Delta$, *if* $y < x$ *then* $y \in C^I\}$

The basic idea is simply to interpret the preference relation $<$ as an accessibility relation. By the Smoothness Condition, it turns out that the modality $\Box$ has the properties of Gödel-Löb modal logic of provability G. The Smoothness Condition ensures that typical elements of C^I exist whenever $C^I \neq \emptyset$, by preventing infinitely descending chains of elements. This condition therefore corresponds to the finite-chain condition on the accessibility relation (as in G). A similar correspondence has been presented in [9, 8] to interpret the preference relation in KLM logics. The following relation between $\mathbf{T}$ and $\Box$ holds:

Proposition 2. *For all $x \in \Delta$, we have $x \in (\mathbf{T}(C))^I$ iff $x \in C^I$ and $x \in (\Box\neg C)^I$*

Since we only use $\Box$ to capture the meaning of $\mathbf{T}$, in the following we will always use $\Box$ followed by a negated concept, as in $\Box\neg C$.

We can establish the following equation between our typicality operator and the nonmonotonic (conditional) inference operator $\mathrel{|\!\sim}$ (describing what can be *typically* derived from a given premise) by letting $C \mathrel{|\!\sim} D$ iff $\mathbf{T}(C) \sqsubseteq D$. It can be easily shown that there is a correspondence between the properties of $\mathbf{T}$ and the properties of $\mathrel{|\!\sim}$ in the system **P** described in [11].

3 Reasoning

In this section we present a tableau calculus for deciding the satisfiability of a knowledge base. Given a KB (TBox,ABox), any concrete reasoning system

should provide the usual reasoning services, namely *satisfiability of the KB*, *concept satisfiability*, *subsumption*, and *instance checking*. It is well known that the latter three services are reducible to the satisfiability of a KB.

We introduce a labelled tableau calculus for our logic $\mathcal{ALC}+\mathbf{T}$, which enriches the labelled tableau calculus for $\mathcal{ALC}$ presented in [4]. The calculus is called $T^{\mathcal{ALC}+\mathbf{T}}$ and it is based on the notion of *constraint system*. We consider a set of *variables* drawn from a denumerable set $\mathcal{V}$. $T^{\mathcal{ALC}+\mathbf{T}}$ makes use of labels, which are denoted with $x, y, z, \ldots$. Labels represent *objects*. An object is either a variable or an individual of the ABox, that is to say an element of $\mathcal{O} \cup \mathcal{V}$.

A *constraint* is a syntactic entity of the form $x \xrightarrow{R} y$ or $x : C$, where R is a role and C is either an extended concept or has the form $\Box\neg D$ or $\neg\Box\neg D$, where D is a concept. As we will define in Definition 6, the ABox of an $\mathcal{ALC}$ + **T**-knowledge base can be translated into a set of constraints by replacing every membership assertion $C(a)$ with the constraint $a : C$ and every role aRb with the constraint $a \xrightarrow{R} b$. A tableau is a tree whose nodes are pairs $\langle S \mid U \rangle$, where:

- S contains constraints (or *labelled* formulas) of the form $x : C$ or $x \xrightarrow{R} y$;
- U contains formulas of the form $C \sqsubseteq D^L$, representing subsumption relations $C \sqsubseteq D$ of the TBox. L is a list of labels. As we will discuss later, this list is used in order to ensure the termination of the tableau calculus.

A node $\langle S \mid U \rangle$ is also called a *constraint system*. A branch is a sequence of nodes $\langle S_1 \mid U_1 \rangle, \langle S_2 \mid U_2 \rangle, \ldots, \langle S_n \mid U_n \rangle \ldots$, where each node $\langle S_i \mid U_i \rangle$ is obtained by its immediate predecessor $\langle S_{i-1} \mid U_{i-1} \rangle$ by applying a rule of $T^{\mathcal{ALC}+\mathbf{T}}$, having $\langle S_{i-1} \mid U_{i-1} \rangle$ as the premise and $\langle S_i \mid U_i \rangle$ as one of its conclusions. A branch is closed if one of its nodes is an instance of (Clash), otherwise it is open. We say that a tableau is closed if all its branches are closed.

Given a KB, we define its *corresponding constraint system* as follows:

Definition 6 (Corresponding constraint system). *Given an* $\mathcal{ALC}$ + **T**-*knowledge base* $(TBox, ABox)$*, we define its* corresponding constraint system $\langle S \mid U \rangle$ *as follows:* $S = \{a : C \mid C(a) \in ABox\} \cup \{a \xrightarrow{R} b \mid aRb \in ABox\}$ *and* $U = \{C \sqsubseteq D^{\emptyset} \mid C \sqsubseteq D \in TBox\}$.

Definition 7 (Model satisfying a constraint system). *Let* $\mathcal{M}$ *be a model as defined in Definition 4. We define a function* α *which assigns to each variable of* $\mathcal{V}$ *an element of* Δ*, and assigns every individual* $a \in \mathcal{O}$ *to* $a^I \in \Delta$*.* $\mathcal{M}$ *satisfies* $x : C$ *under* α *if* $\alpha(x) \in C^I$ *and* $x \xrightarrow{R} y$ *under* α *if* $(\alpha(x), \alpha(y)) \in R^I$*. A constraint system* $\langle S \mid U \rangle$ *is satisfiable if there is a model* $\mathcal{M}$ *and a function* α *such that* $\mathcal{M}$ *satisfies under* α *every constraint in* S *and that, for all* $C \sqsubseteq D \in U$ *and for all* x *occurring in* S*, we have that if* $\alpha(x) \in C^I$ *then* $\alpha(x) \in D^I$.

Proposition 3. *Given an* $\mathcal{ALC}$ + **T**-*knowledge base, it is satisfiable if and only if its corresponding constraint system is satisfiable.*

Therefore, in order to check the satisfiability of (TBox,ABox), we build its corresponding constraint system $\langle S \mid U \rangle$, and then we use $T^{\mathcal{ALC}+\mathbf{T}}$ to check the

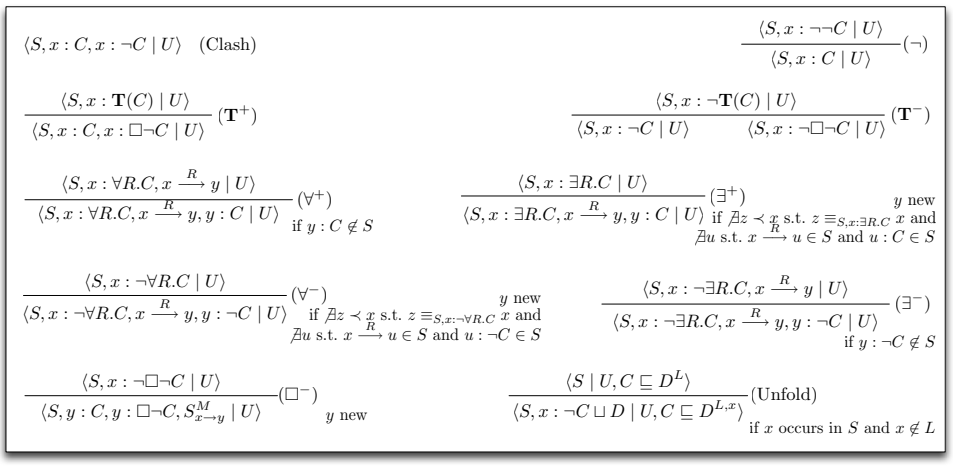

Fig. 1. The calculus $T^{\mathcal{ALC}+\mathbf{T}}$. To save space, we omit the standard rules for $\sqcup$ and $\sqcap$.

satisfiability of $\langle S \mid U\rangle$. In order to check a constraint system $\langle S \mid U\rangle$ for satisfiability, our calculus $T^{\mathcal{ALC}+\mathbf{T}}$ adopts the usual technique of applying the rules until either a contradiction is generated (Clash) or a model satisfying $\langle S \mid U\rangle$ can be obtained from the resulting constraint system.

In order to take into account the TBox, we use a technique of *unfolding*, similar to the one described in [4]. Given a node $\langle S \mid U\rangle$, for each subsumption $C \sqsubseteq D^L \in U$ and for each label x that appears in the tableau, we add to S the constraint $x : \neg C \sqcup D$. As mentioned above, each formula $C \sqsubseteq D$ is equipped by the list L of labels in which it has been unfolded in the current branch. This is needed in order to avoid multiple unfolding of the same subsumption by using the same label, generating non-termination in a proof search.

Before introducing the rules of $T^{\mathcal{ALC}+\mathbf{T}}$ we need some more definitions. First, as in [4], we assume that labels are introduced in a tableau according to an ordering $\prec$, that is to say if y is introduced in the tableau, then $x \prec y$ for all labels x that are already in the tableau.

Given a tableau node $\langle S \mid U\rangle$ and an object x, we define $\sigma(\langle S \mid U\rangle, x) = \{C \mid x : C \in S\}$. Furthermore, we say that two labels x and y are *S-equivalent*, written $x \equiv_S y$, if they label the same set of concepts, i.e. $\sigma(\langle S \mid U\rangle, x) = \sigma(\langle S \mid U\rangle, y)$. Intuitively, S-equivalent labels can represent the same element in the model built by the rules of $T^{\mathcal{ALC}+\mathbf{T}}$. Last, we define $S^M_{x\to y} = \{y : C, y : \Box\neg C \mid x : \Box\neg C \in S\}$.

The rules of $T^{\mathcal{ALC}+\mathbf{T}}$ are presented in Figure 1. Rules $(\exists^+)$, $(\forall^-)$, and $(\Box^-)$ are called *dynamic* since they introduce a new variable in their conclusions. The other rules are called *static*. We do not need any extra rule for the positive occurrences of the $\Box$ operator, since these are taken into account by the computation of $S^M_{x\to y}$. The side conditions on the rules $(\exists^+)$ and $(\forall^-)$ are introduced in order to ensure a terminating proof search, by implementing the standard *blocking* technique described below. The rules of $T^{\mathcal{ALC}+\mathbf{T}}$ are applied with the following

standard strategy: 1. apply a rule to a variable $x \in \mathcal{V}$ only if no rule is applicable to a variable $y \in \mathcal{V}$ such that $y \prec x$; 2. apply dynamic rules ($(\Box^-)$ first) only if no static rule is applicable. This strategy ensures that the variables are considered one at a time according to the ordering $\prec$. Consider an application of a dynamic rule to a variable x of a constraint system $\langle S \mid U \rangle$. For all $\langle S' \mid U' \rangle$ obtained from $\langle S \mid U \rangle$ by a sequence of rule applications, it can be easily shown that (i) no rule can be applied in $\langle S' \mid U' \rangle$ to a variable y s.t. $y \prec x$ and (ii) $\sigma(\langle S \mid U \rangle, x) = \sigma(\langle S' \mid U' \rangle, x)$. The calculus so obtained is sound and complete wrt to the semantics described in Definition 7 (we omit the proof to save space).

Theorem 2 (Soundness and Completeness of $T^{\mathcal{ALC}+\mathbf{T}}$). *Given a constraint system $\langle S \mid U \rangle$, it is unsatisfiable iff it has a closed tableau.*

Let us conclude this section by analyzing termination and complexity of $T^{\mathcal{ALC}+\mathbf{T}}$. In general, non-termination in labelled tableau calculi can be caused by two different reasons: 1. some rules copy their principal formula in the conclusion(s), and can thus be reapplied over the same formula without any control; 2. dynamic rules may generate infinitely-many labels, creating infinite branches.

Concerning the first source of non-termination (point 1), the only rules copying their principal formulas in their conclusions are $(\forall^+)$, $(\exists^-)$, (Unfold), $(\forall^-)$, and $(\exists^+)$. However, the side conditions on these rules avoid multiple applications on the same formula. Indeed, (Unfold) can be applied to a constraint system $\langle S \mid U, C \sqsubseteq D^L \rangle$ by using the label x only if it has not yet been applied to x in the current branch (i.e. x does not belong to L). Concerning $(\forall^+)$, the rule can be applied to $\langle S, x : \forall R.C, x \xrightarrow{R} y \mid U \rangle$ only if $y : C$ does not belong to S. When $y : C$ is introduced in the branch, the rule will not further apply to $x : \forall R.C$. The same for $(\exists^-)$, $(\exists^+)$, and $(\forall^-)$.

Concerning the second source of non-termination (point 2), we can prove that we only need to adopt the standard loop-checking machinery known as *blocking*, which ensures that the rules $(\exists^+)$ and $(\forall^-)$ do not introduce infinitely-many labels on a branch. Thanks to the properties of $\Box$, no other additional machinery is required to ensure termination. Indeed, we can show that the interplay between rules $(\mathbf{T}^-)$ and $(\Box^-)$ does not generate branches containing infinitely-many labels. Let us discuss the termination in more detail.

Without the side conditions on the rules $(\exists^+)$ and $(\forall^-)$, the calculus $T^{\mathcal{ALC}+\mathbf{T}}$ does not ensure a terminating proof search. Indeed, given a constraint system $\langle S \mid U \rangle$, it could be the case that $(\exists^+)$ is applied to a constraint $x : \exists R.C \in S$, introducing a new label y and the constraints $x \xrightarrow{R} y$ and $y : C$. If an inclusion $\mathbf{T}(\exists R.C) \sqsubseteq D$ belongs to U, then (Unfold) can be applied by using y, thus generating a branch containing $y : \neg\mathbf{T}(\exists R.C)$, to which $(\mathbf{T}^-)$ can be applied introducing $y : \neg\Box\neg(\exists R.C)$. An application of $(\Box^-)$ introduces a new variable z and the constraint $z : \exists R.C$, to which $(\exists^+)$ can be applied generating a new label u. (Unfold) can then be re-applied on $\mathbf{T}(\exists R.C) \sqsubseteq D$ by using u, incurring a loop. In order to prevent this source of non termination, we adopt the standard technique of *blocking*: the side condition of the $(\exists^+)$ rule says that this rule can be applied to a node $\langle S, x : \exists R.C \mid U \rangle$ only if there is no z occurring in S such

that $z \prec x$ and $z \equiv_{S,x:\exists R.C} x$. In other words, if there is an "older" label z which is equivalent to x wrt $S, x : \exists R.C$, then $(\exists^+)$ is not applicable, since the condition and the strategy imply that the $(\exists^+)$ rule has already been applied to z. In this case, we say that x is *blocked* by z. The same for $(\forall^-)$.

As mentioned, another possible source of infinite branches could be determined by the interplay between rules $(\mathbf{T}^-)$ and $(\Box^-)$. This cannot occur, i.e. the interplay between these two rules does not generate branches containing infinitely-many labels. Intuitively, the application of $(\Box^-)$ to $x : \neg\Box\neg C$ adds $y : \Box\neg C$ to the conclusion, so that $(\mathbf{T}^-)$ can no longer consistently introduce $y : \neg\Box\neg C$. This is due to the properties of $\Box$ (no infinite descending chains of $<$ are allowed). More in detail, if (Unfold) is applied to $\mathbf{T}(C) \sqsubseteq D$ by using x, an application of $(\mathbf{T}^-)$ introduces a branch containing $x : \neg\Box\neg C$; when a new label y is generated by an application of $(\Box^-)$ on $x : \neg\Box\neg C$, we have that $y : \Box\neg C$ is added to the current constraint system. If (Unfold) and $(\mathbf{T}^-)$ are also applied to $\mathbf{T}(C) \sqsubseteq D$ on the new label y, then the conclusion where $y : \neg\Box\neg C$ is introduced is closed, by the presence of $y : \Box\neg C$. By this fact, we do not need to introduce any loop-checking machinery on the application of $(\Box^-)$.

Theorem 3 (Termination of $T^{\mathcal{ALC}+\mathbf{T}}$). *Let $\langle S \mid U\rangle$ be a constraint system, then any tableau generated by $T^{\mathcal{ALC}+\mathbf{T}}$ is finite.*

Since the calculus $T^{\mathcal{ALC}+\mathbf{T}}$ is sound and complete (Theorem 2), and since an $\mathcal{ALC} + \mathbf{T}$-knowledge base is satisfiable iff its corresponding constraint system is satisfiable (Proposition 3), from Theorem 3 above it immediately follows that:

Theorem 4 (Decidability). *Checking whether a given $\mathcal{ALC} + \mathbf{T}$-knowledge base is satisfiable is a decidable problem.*

Let us conclude this section with a complexity analysis of the calculus $T^{\mathcal{ALC}+\mathbf{T}}$:

Theorem 5 (Complexity). *Given an $\mathcal{ALC}+\mathbf{T}$-knowledge base, checking whether it is satisfiable can be solved in nondeterministic exponential time.*

Proof. We first show that the number of labels generated on a branch is at most exponential in the size of KB. Let n be the size of a KB. Given a constraint system $\langle S \mid U\rangle$, the number of extended concepts appearing in $\langle S \mid U\rangle$, including also all the ones appearing as a subformula of other concepts, is $O(n)$. As there are at most $O(n)$ concepts, there are at most $O(2^n)$ variables labelling distinct sets of concepts. Hence, there are $O(2^n)$ non-blocked variables in S.

Let m be the maximum number of direct successors of each variable $x \in S$, obtained by applying dynamic rules. m is bound by the number of $\exists R.C$ concepts $(O(n))$ plus the number of $\neg\forall R.C$ concepts $(O(n))$ plus the number of $\neg\Box\neg C$ concepts $(O(n))$. Therefore, there are at most $O(2^n \times m)$ *variables* in S, where $m \leq 3n$. The number of *individuals* in the ABox is bound by n too, and each individual has at most m direct successors. The number of *labels* in S is then bound by $O((2^n + n) \times m)$, and hence by $O(2^{2n})$.

For a given label x, the concepts labelled by x introduced in the branch (namely, all the possible subconcepts of the initial constraint system, as well

as all boxed subconcepts) are $O(n)$. According to the standard strategy, after all static rules have been applied to a label x in phase 1, no other concepts labelled by x can be introduced later on a branch. Hence, the labelled concepts introduced on the branch is $O(n)$ for each label, and the number of all labelled concepts on the branch is $O(n \times 2^{2n})$. Therefore, a branch can contain at most an exponential number of applications of tableau rules.

The satisfiability of a KB can thus be solved by defining a procedure which nondeterministically generates an open branch of exponential size (in the size of KB). The problem is in NEXPTIME. ■

4 Reasoning about Typicality

Logic $\mathcal{ALC} + \mathbf{T}$ allows one to reason monotonically about typicality. In $\mathcal{ALC} + \mathbf{T}$ we can consistently express, for instance, the fact that three different concepts, as *student*, *working student* and *working student with children*, have a different status as taxpayers.

What about the typical properties of an individual *john* that we know being a working student, and having children? Of course, if we know that *john* is a typical instance of the concept $\mathit{Student} \sqcap \mathit{Worker} \sqcap \exists \mathit{HasChild}.\top$, i.e. if the ABox contains the assertion $(*)$ $\mathbf{T}(\mathit{Student} \sqcap \mathit{Working} \sqcap \exists \mathit{HasChild}.\top)(\mathit{john})$, then, in $\mathcal{ALC} + \mathbf{T}$, we can conclude that $\neg \mathit{TaxPayer}(\mathit{john})$. However, in absence of (*), we cannot derive $\neg \mathit{TaxPayer}(\mathit{john})$.

In general, we would like to infer that individuals have the properties which are typical of the most specific concept to which they belong. To this purpose, we define a completion of the knowledge base which adds to the ABox, for each individual a occurring in the ABox, the assertion that a is a typical instance of the most specific concept C to which it belongs. Although in general ABoxes can contain typicality assertions about individuals, in practice we assume that typicality assertions are automatically generated by the system by means of the completion, and are not inserted by the user. From now on, we therefore assume that the initial ABox of a KB does not contain any typicality assertion.

Definition 8 (Completion of a Knowledge Base). *The KB (TBox,ABox') is the completion of the KB (TBox,ABox), if ABox' is obtained from ABox by adding to it, for all individual names a in the ABox, the assertion* $\mathbf{T}(C_1 \sqcap \ldots \sqcap C_j)(a)$*, where* $C_1, \ldots, C_j$ *are all the concepts* C_i *such that: (1)* C_i *is a subconcept of any concept occurring in (TBox,ABox); (2)* C_i *does not contain* $\mathbf{T}$*; (3) a is an instance of* C_i*, i.e.* $C_i(a)$ *is derivable in* $\mathcal{ALC}$ *from (TBox,ABox).*

For instance, assuming that $\mathit{Student}(\mathit{john})$, $\mathit{Worker}(\mathit{john})$ and $\exists \mathit{HasChild}.\top(\mathit{john})$ are the only assertions concerning *john* derivable from the KB, the completion above would add $\mathbf{T}(\mathit{Student} \sqcap \mathit{Worker} \sqcap \exists \mathit{HasChild}.\top)(\mathit{john})$ to the ABox, as $\mathit{Student} \sqcap \mathit{Worker} \sqcap \exists \mathit{HasChild}.\top$ is the most specific concept of which *john* is an instance. From this, we can conclude in $\mathcal{ALC} + \mathbf{T}$ that *john* does not pay taxes.

The completion adds $\mathbf{T}(C_1 \sqcap \ldots \sqcap C_j)(a)$ by considering each $C_i(a)$ *derivable* in $\mathcal{ALC}$ from the KB, rather than considering only $C_i(a)$ in the ABox. This is needed, for instance, to infer that *john* does not pay taxes from the KB containing $\mathit{Professor} \sqsubseteq \forall \mathit{HasChild}.\mathit{Student}$, $\mathit{Professor}(\mathit{paul})$, and $\mathit{HasChild}(\mathit{paul}, \mathit{john})$.

As a matter of fact, if we had in the ABox the information that *john* is a *TaxPayer*, this would not cause an inconsistent completion of the KB. Indeed, in such a case, $\mathit{Student} \sqcap \mathit{Worker} \sqcap \exists \mathit{HasChild}.\top \sqcap \mathit{TaxPayer}$ would be the most specific concept of which *john* is an instance, so that the assertion $\mathbf{T}(\mathit{Student} \sqcap \mathit{Worker} \sqcap \exists \mathit{HasChild}.\top \sqcap \mathit{TaxPayer})(\mathit{john})$ would be added in the completion of the KB. This does not allow to infer that $\neg \mathit{TaxPayer}(\mathit{john})$. Hence, no inconsistency arises. However, it could be the case that the KB obtained by the completion is inconsistent, even if the initial KB is consistent. For instance, the KB containing $\mathbf{T}(C) \sqsubseteq \forall R.E$, $\mathbf{T}(D) \sqsubseteq \forall R.\neg E$, $C(a)$, $D(b)$, $R(a, c)$, and $R(b, c)$ is consistent, whereas its completion, including also $\mathbf{T}(C)(a)$ and $\mathbf{T}(D)(b)$, is not. In this case, we keep the initial KB unaltered, instead of the one obtained by the completion.

Notice that the completion of the ABox only introduces $O(n)$ new formulas $a : \mathbf{T}(C_1 \sqcap \ldots \sqcap C_j)$, one for each named individual a in the ABox. Furthermore, the size of each formula $\mathbf{T}(C_1 \sqcap \ldots \sqcap C_j)$ is $O(n^2)$ as $C_1, \ldots, C_j$ are all distinct subformulas of the initial formula ($O(n)$), and each C_i has size $O(n)$. Hence, after the completion construction, the size of the KB is polynomial in n. Moreover, for each individual a ($O(n)$) and for each concept C ($O(n)$), we have to check whether $C(a)$ is derivable in $\mathcal{ALC}$ from the KB, which is a problem in EXPTIME. Hence, the completion construction requires exponential time and produces a KB of size polynomial in the size of the original one:

Theorem 6. *The problem of deciding satisfiability of the knowledge base after completion is in* NEXPTIME *in the size of the original KB.*

As mentioned, given a consistent KB, its completion could be inconsistent. In this case, we choose to keep the original KB. As an alternative, we could consider all maximal consistent KBs (extensions) that can be generated by adding, for all individuals, the relative most-specific concept assumptions. We could then perform either a skeptical or a credulous reasoning with respect to such extensions.

Preferential logic allows to deal with some forms of inheritance among concepts, by the property of cautious monotonicity (which comes from the semantic property ($f_{\mathbf{T}} - 3$)): if $\mathbf{T}(C) \sqsubseteq D$ and $\mathbf{T}(C) \sqsubseteq E$, then $\mathbf{T}(C \sqcap D) \sqsubseteq E$. Coming back to the example above, if we knew that all students typically have a teacher, i.e. $\mathbf{T}(\mathit{Student}) \sqsubseteq \exists \mathit{HasTeacher}.\top$, and that *john* is a student and has a teacher ($\mathit{Student}(\mathit{john})$ and $\exists \mathit{HasTeacher}.\top(\mathit{john})$ are in the ABox) then, by the completion construction above, we would get $\mathbf{T}(\mathit{Student} \sqcap \exists \mathit{HasTeacher}.\top)(\mathit{john})$, and, by cautious monotonicity, we would conclude that *john* does not pay taxes.

5 Extensions

We consider this work as a first step. We plan to extend our approach in the following directions.

Inheritance with exceptions. Once the completion of a KB has been defined as above, the problem of inferring the typical properties of an individual is reduced to the problem of inferring the properties of the most specific concept to which it belongs. This can be done by reasoning on the typical properties of concepts in the TBox. Although Preferential Description Logic allows to capture - through cautious monotonicity - some form of inheritance of typical properties among concepts, there are cases in which cautious monotonicity is not strong enough to derive the intended conclusions. For instance, if we know that *jack* is a student who is a sport lover, we cannot conclude that *jack* is not a tax payer, as we do not have the property that typical students (or all students) are sport lovers, and hence cautious monotonicity is not applicable. Here we are faced with the problem of *Irrelevance*. Since the property of being a sport lover is irrelevant with respect to the property of paying taxes, we would like to infer that also $\mathbf{T}(\mathit{Student} \sqcap \mathit{SportLover}) \sqsubseteq \neg \mathit{TaxPayer}$, and therefore that *jack* is not a tax payer. In order to allow this form of inheritance among concepts, we can introduce a default rule of the following type:

$$\frac{\mathbf{T}(\mathit{Student}) \sqsubseteq \neg \mathit{TaxPayer} \qquad : \mathbf{T}(\mathit{Student} \sqcap \mathit{SportLover}) \not\sqsubseteq \mathit{TaxPayer}}{\mathbf{T}(\mathit{Student} \sqcap \mathit{SportLover}) \sqsubseteq \neg \mathit{TaxPayer}} (IRR)$$

By the default rule above, if typical students are not tax payers, and *it is consistent* to assume that typical students who are sport lovers are not tax payers, then we could conclude that typical sport lover students are not tax payers. With this rule, the typical properties of a more general concept C are considered one by one, and are inherited by a more specific concept $(C \sqcap D)$ if it is consistent to do so. In order to deal with default rules like this one, we need to integrate our calculus with a standard mechanism to reason about defaults.

Reasoning on the typicality of all instances. The completion of a knowledge base, as defined above, only applies to individuals explicitly named in the ABox. However, we would like to reason on the typical properties of all individuals. Assume, for instance, that the ABox contains the assertions: $\exists \mathit{HasChild}.\mathit{Worker}(\mathit{bill})$ and $\forall \mathit{HasChild}.\mathit{Student}(\mathit{bill})$. Thus, *bill* has a child who is a student and is working. We want to be able to infer that *bill* has a child who is a tax payer. To this purpose, we need to assume that the *bill*'s child is a typical working student.

To reason about the typicality of all individuals, we would need to assume that all individuals generated during the tableau construction are *typical* instances of the most specific concept to which they belong. To this purpose, we could think of applying the completion construction of Definition 8 to all generated individuals. The completion construction would be applied only when all relevant formulas $y : C_1, \ldots, y : C_j$ with label y have already been introduced in the branch. According to the strategy described in section 3, the completion should be performed only after the application of the rules to all x s.t. $x \prec y$.

Extension to other DLs. We want to study the extension of our approach to more expressive description logics. For instance, we plan to extend $\mathcal{ALCNR}$ considered in [4] with our typicality operator $\mathbf{T}$, and consider which is the complexity corresponding to this extension. Finally, we want to study the extension of the language of concepts by allowing arbitrary occurrences of the operator $\mathbf{T}$.

6 Conclusions

We have proposed an extension of $\mathcal{ALC}$ for reasoning about typicality in Description Logic framework. The resulting logic is called $\mathcal{ALC} + \mathbf{T}$. We have proposed a calculus for deciding the satisfiability of a general knowledge base in $\mathcal{ALC} + \mathbf{T}$. The calculus, called $T^{\mathcal{ALC}+\mathbf{T}}$, is analytic, terminating, and allows us to decide the satisfiability of a knowledge base in $\mathcal{ALC} + \mathbf{T}$ in nondeterministic exponential time. The calculus is reminiscent of the tableaux calculi for KLM logics presented in [9, 8]. We have then shown how to complete the ABox by means of typicality assumptions, in order to infer prototypical properties of the individuals explicitly mentioned in the ABox. We have argued how to apply a similar completion also to individuals implicitly mentioned in the ABox, in order to infer their properties. Finally, we have sketched how to reason about the inheritance of typical properties from more general to more specific concepts handling with irrelevant information, by using appropriate default rules.

KLM logics are related to probabilistic reasoning. A probabilistic extension of DLs has been proposed in [10]. In particular, the notion of conditional constraint in [10] allows typicality assertions to be expressed (with a specified probability). We plan to compare in details this probabilistic approach to ours elsewhere.

References

1. F. Baader and B. Hollunder. Embedding defaults into terminological knowledge representation formalisms. *J. Autom. Reasoning*, 14(1):149–180, 1995.
2. F. Baader and B. Hollunder. Priorities on defaults with prerequisites, and their application in treating specificity in terminological default logic. *J. Autom. Reasoning*, 15(1):41–68, 1995.
3. P. A. Bonatti, C. Lutz, and F. Wolter. Description logics with circumscription. In *Proc. of KR*, pages 400–410, 2006.
4. M. Buchheit, F. M. Donini, and A. Schaerf. Decidable reasoning in terminological knowledge representation systems. *J. Artif. Int. Research (JAIR)*, 1:109–138, 1993.
5. F. M. Donini, M. Lenzerini, D. Nardi, W. Nutt, and A. Schaerf. An epistemic operator for description logics. *Artif. Intell.*, 100(1-2):225–274, 1998.
6. F. M. Donini, D. Nardi, and R. Rosati. Description logics of minimal knowledge and negation as failure. *ACM Trans. Comput. Log.*, 3(2):177–225, 2002.
7. T. Eiter, T. Lukasiewicz, R. Schindlauer, and H. Tompits. Combining answer set programming with description logics for the semantic web. In *KR*, 141-151, 2004.
8. L. Giordano, V. Gliozzi, N. Olivetti, and G. L. Pozzato. Analytic Tableaux Calculi for KLM Rational Logic R. In *Proc. of JELIA 2006, LNAI* 4160, pp. 190-202.
9. L. Giordano, V. Gliozzi, N. Olivetti, and G. L. Pozzato. Analytic Tableaux for KLM Preferential and Cumulative Logics. In *LPAR 2005, LNAI* 3835, 666-681.
10. R. Giugno and T. Lukasiewicz. P-$\mathcal{SHOQ}$(D): A Probabilistic Extension of $\mathcal{SHOQ}$(D) for Probabilistic Ontologies in the Semantic Web. In *JELIA'02,* 86-97.
11. S. Kraus, D. Lehmann, and M. Magidor. Nonmonotonic reasoning, preferential models and cumulative logics. *Artificial Intelligence*, 44(1-2):167–207, 1990.
12. U. Straccia. Default inheritance reasoning in hybrid kl-one-style logics. In *Proc. of IJCAI*, pages 676–681, 1993.