An ASP Translation for Non-Monotonic Reasoning on DL-Lite$_{\mathcal{R}}$ with Prototype Descriptions

Gabriele Sacco[1,2], Loris Bozzato[3,*] and Oliver Kutz[2]

[1]Fondazione Bruno Kessler, Via Sommarive 18, 38123 Trento, Italy
[2]Free University of Bozen-Bolzano, Piazza Università 1, 39100, Bolzano, Italy
[3]DiSTA - Università dell'Insubria, Via O. Rossi 9, 21100 Varese, Italy

Abstract

In Artificial Intelligence, defeasible reasoning has been studied as one of the key features of common-sense reasoning and consequently various kinds of non-monotonic logics have been developed to model it formally. We recently developed a non-monotonic logic in the Description Logic (DL) framework based on a combination of ideas from prototype theory, weighted DLs (aka "tooth logic"), and earlier work on justifiable exceptions. A central ingredient in the new framework is the notion of a prototype description, weighted characterisations of concepts denoting the typical features of its members. In this paper, we develop an initial ASP translation for this system which allows to reason on instance level queries in the preferred models of a knowledge base. In particular, under reasonable conditions on the form of the input knowledge base, we show that preference reasoning on answer sets can be encoded via standard ASP constructs. We show that the translation is complete with respect to the preferential semantics of our system.

Keywords

Non-monotonic logic, Typicality, Prototype theory, Description Logics, Answer Set Programming

1. Introduction

Defeasible reasoning is the capability of a system to reason with generalisations that may not be valid for all instances: that is, general statements may admit exceptions. Defeasible reasoning is a key feature of common-sense reasoning, thus its representation in logical formalisms via non-monotonic logics has been studied in Artificial Intelligence since the earlier years of the field [1, 2]. More recently, approaches for representing non-monotonicity have been proposed also in the context of Descriptions Logics (DL) [3, 4].

Following this direction, in [5] we introduced a formal approach for defeasible reasoning in DLs, namely *DLs with Prototype Descriptions*, which satisfies desiderata extracted from the studies on Generics [6] and insights from the philosophical and cognitive research on phenomena related to notions of exceptions and defeasibility. From a formal point of view, our work stems from a combination of ideas from prototype theory, weighted DLs (aka "tooth logic" [7]), and earlier work on defeasible DLs with Justifiable Exceptions [8, 9]. A central ingredient in the new framework is the notion of a *prototype description*, weighted characterisations of concepts denoting the typical features of its members. Intuitively, prototype descriptions provide a way to include into DL knowledge bases similarity-based numerical characterisations of prototype instances (for example, extracted and learned from data): in our framework, those (learned) weights together with the ontology axioms are then used to compute a "typicality score" of instances that is considered in defining a preference over DL models. In [10], we further presented different options to characterise the preference on models induced by the computation of the scores on prototypes.

In this paper, we turn our attention to reasoning methods for DLs with Prototype Descriptions. Following the direction of the previous work on Justified Exceptions [8], we present an Answer Set Programming (ASP) encoding for our formalism that enables instance-level reasoning (namely, instance

CILC 2025: 40th Italian Conference on Computational Logic, June 25–27, 2025, Alghero, Italy
*Corresponding author.
gsacco@fbk.eu (G. Sacco); loris.bozzato@uninsubria.it (L. Bozzato); oliver.kutz@unibz.it (O. Kutz)
0000-0001-5613-5068 (G. Sacco); 0000-0003-1757-9859 (L. Bozzato); 0000-0003-1517-7354 (O. Kutz)

checking) in presence of defeasible information in the case of knowledge bases in *DL-Lite*$_\mathcal{R}$ [11]. The ASP translation is an extension to the encoding proposed in [9] for reasoning on defeasible *DL-Lite*$_\mathcal{R}$ KBs with Justifiable Exceptions. In particular, we show that (by assuming some reasonable assumptions on the weights in the input knowledge base) the computation of inferences in preferred models can be encoded in standard ASP with weak constraints. Although the proposed encoding only allows us to reason on a basic form of model preference, it illustrates a simple way for implementing reasoning for DLs with prototype descriptions using readily available ASP tools. Moreover, since the rules defining the model preference are modular with respect to the encoding of models, the proposed ASP translation can be used as a base for encoding more complex preferences, possibly by adopting more flexible methods for ASP preferences [12].

In the following sections, we first provide a brief recollection of DLs with Prototype Descriptions as defined in [10] and we introduce a simple model preference. We then introduce the rules and definitions of our ASP translation: by showing the correspondence between preferred models of the input knowledge base and optimal answer sets of its encoding, we prove the correctness of the translation. We close with discussions of related and future work.

2. DLs with Prototype Descriptions

In our formalism, we distinguish two parts in knowledge bases: *(i).* a DL knowledge base representing the knowledge of interest: the knowledge base can include defeasible TBox axioms about specific base concepts called *prototypes* and ABox assertions about prototypes instances and their features; *(ii).* an additional set containing *prototype descriptions*, weighted characterizations of prototypes, expressing the "degree of typicality" of the features of their istances. For example, in a scenario modelling knowledge about animals, we can have: *(i).* a DL knowledge base, with TBox axioms involving prototype Dog and ABox assertions about $pluto$ being a Dog and its features; *(ii).* a description of the prototypical Dog, expressing that, e.g., a typical Dog most likely has a collar and lives in a house.

In the following, we outline the formal syntax and semantics of such enriched KBs. The following definitions are independent from the DL language of the main knowledge base: we consider a fixed concept language $\mathcal{L}_\Sigma$ based on a DL signature Σ with disjoint and non-empty sets NC of *concept names*, NR of *role names*, and NI of *individual names*. We identify a subset of the concept names $\mathrm{NP} \subseteq \mathrm{NC}$ as denoting *prototype names*. For simplicity, we call *general concepts* the concepts composed only from concepts in $\mathrm{NC} \setminus \mathrm{NP}$.

The features associated with prototypes together with the degree of their importance are given in *prototype descriptions*.

Definition 1 (Positive prototype description). *Let $P \in \mathrm{NP}$ be a prototype name, let $C_1, \ldots, C_m$ be general concepts of $\mathcal{L}_\Sigma$ and let $\overline{w} = (w_1, \ldots, w_m) \in \mathbb{Q}^m$ be a weight vector of rational numbers, where for every $i \in \{1, \ldots, m\}$ we have $w_i > 0$. Then, the expression*

$$P(C_1 : w_1, \ldots, C_m : w_m)$$

is called a (positive) prototype description *for P.*

Note that this description of prototypes is also similar to the definition of concepts with tooth operators as defined in [13]. Intuitively, the weights associated with features can be combined to compute a score denoting the degree of typicality of an instance w.r.t. the prototype: for the current definition, weights are assumed to be positive and features are independent. Note that, to allow for a direct comparison across scores of different prototypes, these need to be normalised to a common value interval, possibly with a scoring function that does not depend on the number of features used in defining different prototypes.

In the knowledge part of the KB, we can use prototype names in DL axioms to describe properties of the instances of such concepts. Here we consider the case in which prototype names are only used as primitive concepts on the left-hand side of concept inclusions.

Definition 2 (Prototype axiom). *A concept inclusion of the type $P \sqsubseteq D$ is a* prototype axiom *of $\mathcal{L}_\Sigma$ if $P \in \mathrm{NP}$ and D is a general concept of $\mathcal{L}_\Sigma$.*

Intuitively, these axioms are not absolute and can be "overridden" by prototype instances (cf. *defeasible axioms* in [8]), also depending on the "degree of membership" of the individual to the given prototype (i.e., the satisfaction of its features).

As noted, we consider knowledge bases which can contain prototype axioms and which are enriched with an accessory KB, the PBox $\mathcal{P}$, providing prototype descriptions.

Definition 3 (Prototyped Knowledge Base, PKB). *A* prototyped knowledge base (PKB) *in language $\mathcal{L}_\Sigma$ is a triple $\mathfrak{K} = \langle \mathcal{T}, \mathcal{A}, \mathcal{P} \rangle$ where:*

- *$\mathcal{T} = T_P \uplus T_C$ is a DL TBox consisting of concept inclusion axioms of the form $C \sqsubseteq D$; $\mathcal{T}$ is partitioned into the sets T_P of prototype axioms and T_C of general concept inclusions based on general concepts;*

- *$\mathcal{A} = A_P \uplus A_C$ is a set of ABox assertions; $\mathcal{A}$ is partitioned into the sets A_P of prototype assertions (of the form $P(a)$ with $P \in \mathrm{NP}$ and $a \in \mathrm{NI}$) and A_C of ABox assertions for general concepts and roles;*

- *$\mathcal{P}$ is a set of prototype descriptions, exactly one for each prototype name $P \in \mathrm{NP}$ appearing in prototype TBox T_P.*

Note that a PKB $\langle \mathcal{T}, \mathcal{A}, \emptyset \rangle$ can be seen as simply a standard DL knowledge base. We present an example to clarify the notions and syntax introduced thus far.

Example 1. *Consider the following* prototyped knowledge base $\mathcal{K} = \langle \mathcal{T}, \mathcal{A}, \mathcal{P} \rangle$*:*

$$\begin{aligned}
\mathcal{T} = \{ & \mathtt{Dog} \sqsubseteq \mathtt{Trusted},\ \mathtt{Wolf} \sqsubseteq \neg\mathtt{Trusted}, \mathtt{Dog} \sqsubseteq \mathtt{hasLegs},\ \mathtt{Wolf} \sqsubseteq \mathtt{hasLegs} \},\\
\mathcal{A} = \{ & \mathtt{Dog(balto)}, \mathtt{Wolf(balto)},\ \mathtt{Dog(pluto)}, \mathtt{Wolf(alberto)},\ \mathtt{Dog(cerberus)},\\
& \mathtt{livesInWoods(balto)},\ \mathtt{hasLegs(balto)},\ \mathtt{isTamed(balto)},\\
& \mathtt{hasCollar(pluto)},\ \mathtt{hasLegs(pluto)},\ \mathtt{isTamed(pluto)},\\
& \mathtt{hasLegs(alberto)},\ \mathtt{Hunts(alberto)},\\
& \neg\mathtt{Trusted(cerberus)} \},\\
\mathcal{P} = \{ & \mathtt{Wolf(livesInWoods} : 10,\ \mathtt{hasLegs} : 4,\ \mathtt{livesInPack} : 8,\ \mathtt{Hunts} : 11),\\
& \mathtt{Dog(hasCollar} : 33,\ \mathtt{livesInHouse} : 22,\ \mathtt{hasLegs} : 11,\ \mathtt{isTamed} : 44) \}
\end{aligned}$$

Basically, $\mathcal{T}$ provides information about the prototype concepts `Dog` *and* `Wolf`*: using prototype axioms, we can state that (generally) dogs are trusted, but wolves are not. The ABox $\mathcal{A}$ provides information about prototype instances and their features: note that* `pluto` *and* `cerberus` *are dogs,* `alberto` *is a wolf, but* `balto` *is both a dog and a wolf. Finally, $\mathcal{P}$ provides the weights of features for prototypes* `Dog` *and* `Wolf`*: instances satisfying features with larger weights are interpreted as "more typical" members of the prototype concept. Note that different prototypes might give different importance to the same feature and can use different value ranges (see, for example, the different values for* `hasLegs` *in* `Wolf` *and* `Dog`*).*

Intuitively, we want to entail and justify the conclusion that `balto` *is a trusted dog which is a wolf, without being inconsistent, and that* `cerberus` *is an exceptional dog with respect to the property of dogs of being trusted. Moreover, in the case of the instances* `pluto` *and* `alberto` *no contradiction arises, thus we want that the axioms in $\mathcal{T}$ are applied to them normally.*

Note that the conflict regarding `balto` *has the same structure as the so called* Nixon diamond *[14].* $\diamond$

The semantics of PKBs is based on standard interpretations of the underlying DL $\mathcal{L}_\Sigma$. In fact, interpretations of a PKB are DL interpretations for its knowledge base part, as follows:

Definition 4 (Interpretation). *A pair $\mathcal{I} = \langle \Delta^\mathcal{I}, \cdot^\mathcal{I} \rangle$ is an* interpretation *for signature Σ with a non-empty domain, $\Delta^\mathcal{I}$, and with $a^\mathcal{I} \in \Delta^\mathcal{I}$ for every $a \in \mathrm{NI}$, $A^\mathcal{I} \subseteq \Delta^\mathcal{I}$ for every $A \in \mathrm{NC}$, $R^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$ for every $R \in \mathrm{NR}$, and where the extension of complex concepts is defined recursively as usual for language $\mathcal{L}_\Sigma$.*

Note that we are not giving a DL interpretation to the prototype description expressions in $\mathcal{P}$. However, we need to introduce additional semantic structure to manage exceptions to prototype axioms in T_P, exploiting the prototype description expressions in $\mathcal{P}$. We consider the notion of axiom instantiation as defined in [8]: intuitively, for an axiom $\alpha \in \mathcal{L}_\Sigma$ the *instantiation* of α with $e \in \mathrm{NI}$, written $\alpha(e)$, is the specialisation of α to e.[1]

Definition 5 (Exception assumptions and clashing sets). *An* exception assumption *is a pair $\langle \alpha, e \rangle$ where $\alpha \in T_P$ is a prototype axiom, $e \in \mathrm{NI}$ is an individual name appearing in $\mathcal{A}$ and such that $\alpha(e)$ is an axiom instantiation of α.*

A clashing set *for $\langle \alpha, e \rangle$ is a satisfiable set $S_{\langle \alpha, e \rangle}$ of ABox assertions s.t. $S_{\langle \alpha, e \rangle} \cup \{\alpha(e)\}$ is unsatisfiable.*

Intuitively, an exception assumption $\langle P \sqsubseteq D, e \rangle$ states that we assume that e is an exception to the prototype axiom $P \sqsubseteq D$ in a given interpretation. Then, the fact that a clashing set $S_{\langle P \sqsubseteq D, e \rangle}$ for $\langle P \sqsubseteq D, e \rangle$ is verified by such an interpretation gives a "justification" of the validity of the assumption of overriding in terms of ABox assertions. This intuition is reflected in the definition of models: we first extend interpretations with a set of exception assumptions.

Definition 6 ($\boldsymbol{\chi}$-interpretation). *A* $\boldsymbol{\chi}$-interpretation *is a structure $\mathcal{I}_{\boldsymbol{\chi}} = \langle \mathcal{I}, \chi \rangle$ where $\mathcal{I}$ is an interpretation and χ is a set of exception assumptions.*

Then, $\boldsymbol{\chi}$-models for a PKB $\mathfrak{K}$ are those $\boldsymbol{\chi}$-interpretations that verify "strict" axioms in $T_C \cup \mathcal{A}$ and defeasibly apply prototype axioms in T_P (excluding the exceptional instances in χ).

Definition 7 ($\boldsymbol{\chi}$-model). *Given a PKB $\mathfrak{K}$, a $\boldsymbol{\chi}$-interpretation $\mathcal{I}_{\boldsymbol{\chi}} = \langle \mathcal{I}, \chi \rangle$ is a* $\boldsymbol{\chi}$-model *for $\mathfrak{K}$ (denoted $\mathcal{I}_{\boldsymbol{\chi}} \models \mathfrak{K}$), if the following holds:*

(i) for every $\alpha \in T_C \cup \mathcal{A}$ of $\mathcal{L}_\Sigma$, $\mathcal{I} \models \alpha$;

(ii) for every $\alpha = P \sqsubseteq D \in T_P$, if $\langle \alpha, d \rangle \notin \chi$, then $\mathcal{I} \models \alpha(d)$.

Two DL interpretations $\mathcal{I}_1$ and $\mathcal{I}_2$ are NI-*congruent*, if $c^{\mathcal{I}_1} = c^{\mathcal{I}_2}$ holds for every $c \in \mathrm{NI}$. This extends to $\boldsymbol{\chi}$-interpretations $\mathcal{I}_{\boldsymbol{\chi}} = \langle \mathcal{I}, \chi \rangle$ by considering interpretations $\mathcal{I}$. Intuitively, we say that a $\boldsymbol{\chi}$-interpretation is justified if all of its exception assumptions have a clashing set that is verified by the interpretation.

Definition 8 (Justifications). *We say that $\langle \alpha, e \rangle \in \chi$ is* justified *for a $\boldsymbol{\chi}$-model $\mathcal{I}_{\boldsymbol{\chi}}$, if some clashing set $S_{\langle \alpha, e \rangle}$ exists such that, for every $\mathcal{I}'_{\boldsymbol{\chi}} = \langle \mathcal{I}', \chi \rangle$ of $\mathfrak{K}$ that is* NI-*congruent with $\mathcal{I}_{\boldsymbol{\chi}}$, it holds that $\mathcal{I}' \models S_{\langle \alpha, e \rangle}$.*

A $\boldsymbol{\chi}$-model $\mathcal{I}_{\boldsymbol{\chi}}$ of a PKB $\mathfrak{K}$ is justified, *if every $\langle \alpha, e \rangle \in \chi$ is justified in $\mathfrak{K}$.*

We consider the notion of logical consequence from justified $\boldsymbol{\chi}$-models (i.e. axioms that are valid in all $\boldsymbol{\chi}$-models that are justified): we write $\mathfrak{K} \models_J \alpha$ if $\mathcal{I}_{\boldsymbol{\chi}} \models \alpha$ for every justified $\boldsymbol{\chi}$-model $\mathcal{I}_{\boldsymbol{\chi}}$ of $\mathfrak{K}$. There can be more than one justified model, in particular for different valid combinations of exception assumptions and justifications. As will be shown in examples, this allows reasoning by cases: scores defined over prototype descriptions' values allow to define a preference over such cases.

The next part of the semantics takes care of defining a preference over exceptions in case of conflicts between different prototype axioms. The main intuition of prototype descriptions is that each individual which is an instance of a prototype is associated with a score which denotes the "degree of typicality" of the individual with respect to the concept described by the prototype. As in [13], such a degree is computed from the prototype features that are satisfied by the instances and their score. Ideally, the prototype score of an individual allows us to determine preferences over models: for an individual, axioms on prototypes with higher score are preferred to the ones on lower scoring prototypes; thus the measure needs to be comparable across different prototypes.

Given the set of prototypes $P \in \mathcal{P}$, a family of *prototype score functions* $\{f_P\}_{P \in \mathcal{P}}$ is composed by functions $f_P : \mathrm{NI} \to \mathbb{R}$ for each prototype name $P \in \mathcal{P}$ such that every function of the family has range in a fixed interval $[x, ..., y] \in \mathbb{R}$.

[1]As in [8], $\alpha(e)$ can be formally specified via the FO-translation of α.

Ideally, these families of functions can then be used to define preferences over models: different preference criteria can be defined, in particular, by using the results of score functions on the exceptional individuals in the exception assumptions' sets χ of $\boldsymbol{\chi}$-interpretations.

In general, a *preference* between exception assumption sets is some relation between sets χ for $\mathfrak{K}$, denoted $\chi_1 > \chi_2$. Given two $\boldsymbol{\chi}$-interpretations $\mathcal{I}^1_{\boldsymbol{\chi}} = \langle \mathcal{I}^1, \chi_1 \rangle$ and $\mathcal{I}^2_{\boldsymbol{\chi}} = \langle \mathcal{I}^2, \chi_2 \rangle$, we then say that $\mathcal{I}^1_{\boldsymbol{\chi}}$ is *preferred* to $\mathcal{I}^2_{\boldsymbol{\chi}}$ (denoted $\mathcal{I}^1_{\boldsymbol{\chi}} > \mathcal{I}^2_{\boldsymbol{\chi}}$) if $\chi_1 > \chi_2$.

Finally, we define the notion of PKB model as a minimal justified model for the PKB.

Definition 9 (PKB model). *An interpretation $\mathcal{I}$ is a* PKB model *of $\mathfrak{K}$ (denoted, $\mathcal{I} \models \mathfrak{K}$) if*

- *$\mathfrak{K}$ has some justified $\boldsymbol{\chi}$-model $\mathcal{I}_\chi = \langle \mathcal{I}, \chi \rangle$.*
- *there exists no justified $\mathcal{I}'_{\boldsymbol{\chi}} = \langle \mathcal{I}', \chi' \rangle$ that is preferred to $\mathcal{I}_{\boldsymbol{\chi}}$.*

The consequence from PKB models of $\mathfrak{K}$ (denoted $\mathfrak{K} \models \alpha$) characterises the "preferred" consequences of the PKB, on the basis of the degree of typicality of instances.

The above definitions are provided for any score function and preference: in the following we will define one specific way of instantiating these definitions (which we later use as a counterpart for the preference in the ASP encoding). For another way of computing scores and preferences, see [10].

A simple score function can be defined by considering the features that are inferable from the KB in a single model:

Definition 10 (Model dependent prototype score). *Given a prototype definition $P(C_1 : w_1, ..., C_m : w_m)$, we define the* score function $score^P_{\mathcal{I}_{\boldsymbol{\chi}_j}} : \mathrm{NI} \to \mathbb{R}$ *for prototype P and a justified $\boldsymbol{\chi}$-model $\mathcal{I}_{\boldsymbol{\chi}_j}$ as:*

$$score^P_{\mathcal{I}_{\boldsymbol{\chi}_j}}(a) = \sum_{\mathcal{I}_{\boldsymbol{\chi}_j} \models C_i(a)} w_i$$

This measure, however, depends on the value interval over which the prototype weights have been defined: in order to compare the score of an individual with scores relative to other prototypes, this value needs to be normalised. We do so by computing the maximum score max_P and minimum score min_P for all prototypes.

The maximum score max_P denotes the score of the maximum value of $score^P$ obtainable from the weights of consistent subset of features of P. Formally, given a prototype description $P(C_1 : w_1, ..., C_m : w_m)$, let $\boldsymbol{S}_P$ be the set of sets $S_P \subseteq \{C_1, \ldots, C_n\}$ s.t. $S_P \cup \mathfrak{K}$ is consistent. Then: $max_P = \max(\sum_{C_i \in S_P} w_i \mid S_P \in \boldsymbol{S}_P)$.

The minimal score min_P denotes the sum of the weights for "unavoidable" features, namely those that are strictly implied by the membership to the prototype concept. Formally: $min_P = \sum_{\mathfrak{K} \models_J P \sqsubseteq C_i} w_i$. Now, a normalised score function $nscore^P_{\mathcal{I}_{\boldsymbol{\chi}}}$ can be derived from $score^P_{\mathcal{I}_{\boldsymbol{\chi}}}$ as:

$$nscore^P_{\mathcal{I}_{\boldsymbol{\chi}}}(a) = \frac{score^P_{\mathcal{I}_{\boldsymbol{\chi}}}(a) \; - \; min_P}{max_P \; - \; min_P}.$$

By this normalisation we obtain a family of prototype score functions with range in the same interval $[0, 1]$, allowing for comparison of prototype scores on the same individual.

Simple model-dependent preference. For the definition of a preference based on such scores, we consider the case in which the scores are equal across all of the justified $\boldsymbol{\chi}$-models of $\mathcal{K}$:

Definition 11 (Stable score). *Given a set $M = \{\mathcal{I}_{\boldsymbol{\chi}} \mid \mathcal{I}_{\boldsymbol{\chi}}$ is a justified $\boldsymbol{\chi}$-model of $\mathcal{K}\}$, $score_P(a)$ is a* stable score *iff for every $\mathcal{I}^i_{\boldsymbol{\chi}}, \mathcal{I}^j_{\boldsymbol{\chi}} \in M$, it holds that $score^P_{\mathcal{I}^i_{\boldsymbol{\chi}}}(a) = score^P_{\mathcal{I}^j_{\boldsymbol{\chi}}}(a)$*

Intuitively, this condition of stability of scores occurs in the case in which the computation of the tipicality scores does not depend on other prototype axioms: this condition is verified, for example, in knowledge bases where axioms and assertions involving features are not dependent on the satisfaction of prototype axioms, but only on the "strict" part of the KB.

Let us consider the case in which *all of the prototypes in $\mathcal{K}$ have stable scores*: in this case, we can propose a notion of preference based on model dependent scores, which comes as a simplification of the model dependent preference introduced in [10]:

Definition 12 (Preference SimpleMDP). *$\chi_1 > \chi_2$ if, for every $\langle P \sqsubseteq D, e\rangle \in \chi_1 \setminus \chi_2$ such that there exists a $\langle Q \sqsubseteq E, f\rangle \in \chi_2 \setminus \chi_1$, it holds that $nscore_P(e) < nscore_Q(f)$.*

Basically, SimpleMDP prefers justified $\boldsymbol{\chi}$-models where the *exceptions* appear for elements of the *lower* scoring prototypes.

Example 2. *Considering the PKB reported in the example above, assume to have two PKB interpretations $\mathcal{I}^1$ and $\mathcal{I}^2$ associated respectively with the following two sets of clashing assumptions:*

$$\chi_1 = \{\langle \mathtt{Wolf} \sqsubseteq \neg\mathtt{Trusted}, \mathtt{balto}\rangle, \langle \mathtt{Dog} \sqsubseteq \mathtt{Trusted}, \mathtt{cerberus}\rangle\}$$
$$\chi_2 = \{\langle \mathtt{Dog} \sqsubseteq \mathtt{Trusted}, \mathtt{balto}\rangle, \langle \mathtt{Dog} \sqsubseteq \mathtt{Trusted}, \mathtt{cerberus}\rangle\}$$

*We have now two $\boldsymbol{\chi}$-interpretations corresponding to $\langle \mathcal{I}^1, \chi_1\rangle$ and $\langle \mathcal{I}^2, \chi_2\rangle$. Assuming that they are also $\boldsymbol{\chi}$-*models, *we can check if the two are also* justified. *Since, the clashing assumptions have the following clashing sets, respectively* $\{\mathtt{Wolf}(\mathtt{balto}), \mathtt{Trusted}(\mathtt{balto}), \mathtt{Dog}(\mathtt{cerberus}), \neg\mathtt{Trusted}(\mathtt{cerberus})\}$ *for the clashing assumptions in χ_1 and* $\{\mathtt{Dog}(\mathtt{balto}), \neg\mathtt{Trusted}(\mathtt{balto}), \mathtt{Dog}(\mathtt{cerberus}), \neg\mathtt{Trusted}(\mathtt{cerberus})\}$ *for those in χ_2, they are both justified.*

In order to decide which model is preferred, we need to compute the prototype scores for balto *and for* cerberus*: note that in our PKB scores are stable. We have: $score_{\mathtt{Wolf}}(\mathtt{balto}) = 14$, $score_{\mathtt{Dog}}(\mathtt{balto}) = 55$, $score_{\mathtt{Dog}}(\mathtt{cerberus}) = 11$. Then we need to normalise them, getting $nscore_{\mathtt{Dog}}(\mathtt{balto}) \approx 0.4$, $nscore_{\mathtt{Wolf}}(\mathtt{balto}) \approx 0.3$, $nscore_{\mathtt{Dog}}(\mathtt{cerberus}) = 0$. Consequently $score_{\mathtt{Wolf}}(\mathtt{balto}) < score_{\mathtt{Dog}}(\mathtt{balto})$ and, since $\langle \mathtt{Dog} \sqsubseteq \mathtt{Trusted}, \mathtt{cerberus}\rangle$ is present in both χ_1 and χ_2 so it does not influence the preference order, then we can conclude that $\chi_1 > \chi_2$. This means that the preferred model, i.e. the PKB model, is $\mathcal{I}^1$ where* balto *is an exception to* $\mathtt{Wolf} \sqsubseteq \neg\mathtt{Trusted}$ *and* cerberus *is an exception to* $\mathtt{Dog} \sqsubseteq \mathtt{Trusted}$. *Consequently, it holds that* $\mathfrak{K} \models \mathtt{Trusted}(\mathtt{balto})$ *and* $\mathfrak{K} \models \neg\mathtt{Trusted}(\mathtt{cerberus})$.

Moreover, we note that for pluto *and* alberto *we can standardly infer* $\mathtt{Trusted}(\mathtt{pluto})$ *and* $\neg\mathtt{Trusted}(\mathtt{alberto})$. *The reason is that the exception assumptions are referred to specific individuals, and since there are no contradicting assertions for* pluto *and* alberto*, there are no clashing sets that justify their assumptions as exceptions. Therefore, axioms in $\mathcal{T}$ apply to them standardly.* ◇

3. Translation to ASP with Preferences

3.1. ASP Translation

Defeasible reasoning on PKBs can be encoded by means of an ASP translation: the base of such encoding is the ASP translation for *DL-Lite*$_{\mathcal{R}}$ with defeasible axioms presented in [9].

If we consider input PKBs in *DL-Lite*$_{\mathcal{R}}$ [11], the translation process can be largely defined as in [9]: the goal of the encoding is to obtain a Datalog representation of the input PKB that can then be used to reason on instance checking queries.

In the following, we adopt the model dependent scoring function: for simplicity, we further assume that the scores are *stable*, already *normalized* across prototype descriptions and only *integers*. Although these assumptions can be seen as limiting, they allow us to show that some simple preference reasoning can already be obtained using standard ASP[2] constructs: these assumptions can be relaxed, for example

[2]In particular, we use weak constraints and aggregates to encode the preference rules: we adopt the syntax of the `clingo` ASP solver for such constructs.

Table 1
Normal form for axioms in $\mathcal{K}$ from $\mathcal{L}_\Sigma$ (cfr. [9, Table 2])

Strict axioms: for $A, B \in \mathrm{NC}$, $R, S \in \mathrm{NR}$, $a, b \in \mathrm{NI}$:

$$A(a) \qquad R(a,b) \qquad A \sqsubseteq B \qquad A \sqsubseteq \neg B \qquad \exists R \sqsubseteq A_{\exists R} \qquad A_{\exists R} \sqsubseteq \exists R$$

$$R \sqsubseteq S \qquad \mathrm{Dis}(R,S) \qquad \mathrm{Inv}(R,S) \qquad \mathrm{Irr}(R)$$

Prototype axioms: for $P \in \mathrm{NP}$, $B \in \mathrm{NC}$: $\quad P \sqsubseteq B$

Table 2
DL-Lite$_\mathcal{R}$ input, deduction and output rules (cfr. [9, Table 3])

***DL-Lite*$_\mathcal{R}$ input translation $I_{dlr}(S)$**

(idlr-nom)	$a \in \mathrm{NI} \mapsto \{\mathtt{nom}(a)\}$	(idlr-subex)	$\exists R \sqsubseteq B \mapsto \{\mathtt{subEx}(R,B)\}$
(idlr-cls)	$A \in \mathrm{NC} \mapsto \{\mathtt{cls}(A)\}$	(idlr-supex)	$A \sqsubseteq \exists R \mapsto \{\mathtt{supEx}(A,R,aux^\alpha)\}$
(idlr-rol)	$R \in \mathrm{NR} \mapsto \{\mathtt{rol}(R)\}$	(idlr-subr)	$R \sqsubseteq S \mapsto \{\mathtt{subRole}(R,S)\}$
(idlr-inst)	$A(a) \mapsto \{\mathtt{insta}(a,A)\}$	(idlr-dis)	$\mathrm{Dis}(R,S) \mapsto \{\mathtt{dis}(R,S)\}$
(idlr-triple)	$R(a,b) \mapsto \{\mathtt{triplea}(a,R,b)\}$	(idlr-inv)	$\mathrm{Inv}(R,S) \mapsto \{\mathtt{inv}(R,S)\}$
(idlr-subc)	$A \sqsubseteq B \mapsto \{\mathtt{subClass}(A,B)\}$	(idlr-irr)	$\mathrm{Irr}(R) \mapsto \{\mathtt{irr}(R)\}$

***DL-Lite*$_\mathcal{R}$ deduction rules P_{dlr}**

(pdlr-instd)	$\mathtt{instd}(x,z) \leftarrow \mathtt{insta}(x,z).$
(pdlr-tripled)	$\mathtt{tripled}(x,r,y) \leftarrow \mathtt{triplea}(x,r,y).$
(pdlr-subc)	$\mathtt{instd}(x,z) \leftarrow \mathtt{subClass}(y,z), \mathtt{instd}(x,y).$
(pdlr-supnot)	$\neg\mathtt{instd}(x,z) \leftarrow \mathtt{supNot}(y,z), \mathtt{instd}(x,y).$
(pdlr-subex)	$\mathtt{instd}(x,z) \leftarrow \mathtt{subEx}(v,z), \mathtt{tripled}(x,v,x').$
(pdlr-supex)	$\mathtt{tripled}(x,r,x') \leftarrow \mathtt{supEx}(y,r,x'), \mathtt{instd}(x,y).$
(pdlr-subr)	$\mathtt{tripled}(x,w,x') \leftarrow \mathtt{subRole}(v,w), \mathtt{tripled}(x,v,x').$
(pdlr-dis1)	$\neg\mathtt{tripled}(x,u,y) \leftarrow \mathtt{dis}(u,v), \mathtt{tripled}(x,v,y).$
(pdlr-dis2)	$\neg\mathtt{tripled}(x,v,y) \leftarrow \mathtt{dis}(u,v), \mathtt{tripled}(x,u,y).$
(pdlr-inv1)	$\mathtt{tripled}(y,v,x) \leftarrow \mathtt{inv}(u,v), \mathtt{tripled}(x,u,y).$
(pdlr-inv2)	$\mathtt{tripled}(y,u,x) \leftarrow \mathtt{inv}(u,v), \mathtt{tripled}(x,v,y).$
(pdlr-irr)	$\neg\mathtt{tripled}(x,u,x) \leftarrow \mathtt{irr}(u), \mathtt{const}(x).$
(pdlr-nsubc)	$\neg\mathtt{instd}(x,y) \leftarrow \mathtt{subClass}(y,z), \neg\mathtt{instd}(x,z).$
(pdlr-nsupnot)	$\mathtt{instd}(x,y) \leftarrow \mathtt{supNot}(y,z), \neg\mathtt{instd}(x,z).$
(pdlr-nsubex)	$\neg\mathtt{tripled}(x,v,x') \leftarrow \mathtt{subEx}(v,z), \mathtt{const}(x'), \neg\mathtt{instd}(x,z).$
(pdlr-nsupex)	$\neg\mathtt{instd}(x,y) \leftarrow \mathtt{supEx}(y,r,w), \mathtt{const}(x), \mathtt{all\_nrel}(x,r).$
(pdlr-nsubr)	$\neg\mathtt{tripled}(x,v,x') \leftarrow \mathtt{subRole}(v,w), \neg\mathtt{tripled}(x,w,x').$
(pdlr-ninv1)	$\neg\mathtt{tripled}(y,v,x) \leftarrow \mathtt{inv}(u,v), \neg\mathtt{tripled}(x,u,y).$
(pdlr-ninv2)	$\neg\mathtt{tripled}(y,u,x) \leftarrow \mathtt{inv}(u,v), \neg\mathtt{tripled}(x,v,y).$
(pdlr-allnrel1)	$\mathtt{all\_nrel\_step}(x,r,y) \leftarrow \mathtt{first}(y), \neg\mathtt{tripled}(x,r,y).$
(pdlr-allnrel2)	$\mathtt{all\_nrel\_step}(x,r,y) \leftarrow \mathtt{all\_nrel\_step}(x,r,y'), \mathtt{next}(y',y), \neg\mathtt{tripled}(x,r,y).$
(pdlr-allnrel3)	$\mathtt{all\_nrel}(x,r) \leftarrow \mathtt{last}(y), \mathtt{all\_nrel\_step}(x,r,y).$

Output translation $O(\alpha)$

(o-concept)	$A(a) \mapsto \{\mathtt{instd}(a,A).\}$
(o-role)	$R(a,b) \mapsto \{\mathtt{tripled}(a,R,b).\}$

by adopting an encoding of rational numbers in ASP [15] and including (possibly external) computations for score normalisation; compare also the approach of [16].

Normal form for *DL-Lite*$_\mathcal{R}$. For the DL knowledge base part of our PKBs, we assume that *DL-Lite*$_\mathcal{R}$ axioms are in the *normal form* presented in [9], which allows to simplify the formulation of the traslation cases. The kind of axioms included in the normal form are shown in Table 1.

A set of rules to transform any *DL-Lite*$_\mathcal{R}$ PKB into this normal form and a proof of equivalence of the rewritten PKB can be given analogously to the original paper. The only notable difference in the case of PKBs is the fact that the only form of "defeasible" axioms is the one of prototype axioms.

Table 3
Input and deduction rules for prototype axioms (cfr. [9, Table 4])

Input rules for prototype axioms $I_{\mathrm{D}}(S)$

(id-subc) $P \sqsubseteq B \in T_P \mapsto \{\, \texttt{def\_subclass}(P, B).\, \}$

Deduction rules for prototype axioms P_{D}: overriding rules

(ovr-subc) $\texttt{ovr}(x, y, z) \leftarrow \texttt{def\_subclass}(y, z), \texttt{instd}(x, y), \neg\texttt{instd}(x, z).$

Deduction rules for prototype axioms P_{D}: application rules

(app-subc) $\texttt{instd}(x, z) \leftarrow \texttt{def\_subclass}(y, z), \texttt{instd}(x, y), \texttt{not ovr}(x, y, z).$

Table 4
Prototype descriptions input and preference rules

Input rule for prototype descriptions $I_{\mathcal{P}}(S)$

(iproto) $P(C_1 : w_1, \ldots, C_n : w_n) \mapsto \{\, \texttt{isProto}(P)., \texttt{featwt}(P, C_1, w_1)., \ \ldots\ \texttt{featwt}(P, C_n, w_n).\, \}$

Deduction rules for score and preference P_{pref}

(dpref-as)	$\texttt{addScore}(x, p, wi) \leftarrow \texttt{instd}(x, p), \texttt{instd}(x, ci), \texttt{featwt}(p, ci, wi).$
(dpref-sc)	$\texttt{score}(x, p, n) \leftarrow \texttt{instd}(x, p), \texttt{isProto}(p), n = \#sum\{wi : \texttt{addScore}(x, p, wi)\}.$
(dpref-os)	$\texttt{ovrscore}(x, p, b, n) \leftarrow \texttt{ovr}(x, p, b), \texttt{score}(x, p, n).$
(dpref-wc)	$\leftsquigarrow \texttt{ovrscore}(x, p, b, n).\ [n]$

Translation rules overview. The datalog encoding is composed by different sets of translation rules (inspired by the materialization calculus in [17]). The complete set of rules in our translation is provided in Tables 2–4. The encoding has a set of *input rules* which translate DL axioms and signature in datalog, *deduction rules* that provide instance level inference by datalog rules, and *output rules* that encode as a datalog fact the ABox assertion to be proved. In our case, the translation includes the following set of rules:

DL-Lite$_{\mathcal{R}}$ input rules: input *DL-Lite$_{\mathcal{R}}$* rules (in Table 2) translate (strict) KB axioms in normal form into their ASP encoding: for example, an atomic concept inclusion $A \sqsubseteq B$ is encoded by the rule:

$$A \sqsubseteq B \mapsto \{\texttt{subClass}(A, B)\}.$$

DL-Lite$_{\mathcal{R}}$ deduction rules: deduction rules for *DL-Lite$_{\mathcal{R}}$* (in Table 2) allow instance level reasoning on the interpretation of encoded axioms: for example, in the case of atomic concept inclusions:

$$\texttt{instd}(x, z) \leftarrow \texttt{subClass}(y, z), \texttt{instd}(x, y).$$

DL-Lite$_{\mathcal{R}}$ output rules: output rules (in Table 2) define the translation of instance queries to the ASP encoding: for example, for an atomic concept assertion $A(a)$, we have the rule:

$$A(a) \mapsto \{\texttt{instd}(a, A).\}.$$

Previous rules allow to translate and reason over the strict part of the KB. Other set of rules provide datalog traslation and reasoning rules for the defeasible part of the PKB:

Prototype axioms input rules: prototype axioms of the form $P \sqsubseteq C$ can be translated using the input rules for defeasible axioms (in Table 4) from the original translation: we can encode $P \sqsubseteq B$ by the rule

$$P \sqsubseteq B \mapsto \{\texttt{def\_subclass}(P, B).\}$$

Note that, with respect to [9], this is the only form of "defeasible" axiom that is defined in our formalism.

Overriding rules: overriding rules (in Table 4) are used to determine when an overriding to the above axiom occurs:

$$\texttt{ovr}(x, y, z) \leftarrow \texttt{def\_subclass}(y, z), \texttt{instd}(x, y), \neg\texttt{instd}(x, z).$$

Defeasible application rules: the following rule (in Table 4) defines when the prototype axiom can be applied, leaving out the instances for which an overriding can be proved:

$$\texttt{instd}(x, z) \leftarrow \texttt{def\_subclass}(y, z), \texttt{instd}(x, y), \texttt{not}\,\texttt{ovr}(x, y, z).$$

The translation rules presented so far are simply a restriction to the use of the encoding proposed in [9]: in fact, this provides an encoding of the standard DL part of a PKB. We now have to introduce an ASP encoding of prototype descriptions and scores in order to define a preference over such answer sets (compatible with the preference defined in our semantics).

Prototype description rule: prototype descriptions can be easily added to the program in form of facts with the following rule (in Table 4):

$$P(C_1 : w_1, \dots, C_n : w_n) \mapsto \{\ \texttt{isProto}(P), \texttt{featwt}(P, C_1, w_1).\ \dots\ \texttt{featwt}(P, C_n, w_n).\ \}$$

Preference rules: we can then use this information to compute the score associated to the exceptions in our answer sets. The score for a particular instance of a prototype can be computed using the rules (in Table 4):

$$\begin{aligned}\texttt{addScore}(x, p, wi) &\leftarrow \texttt{instd}(x, p), \texttt{instd}(x, ci), \texttt{featwt}(p, ci, wi).\\ \texttt{score}(x, p, n) &\leftarrow \texttt{instd}(x, p), \texttt{isProto}(p), n = \#sum\{wi : \texttt{addScore}(x, p, wi)\}.\end{aligned}$$

Then, we can associate a score to the overriding on an individual, based on its score for the particular prototype:

$$\texttt{ovrscore}(x, p, b, n) \leftarrow \texttt{ovr}(x, p, b), \texttt{score}(x, p, n).$$

Using weak constraints, we can prefer the answer sets where overridings occur on the *less typical* elements of prototypes:

$$\leftsquigarrow \texttt{ovrscore}(x, p, b, n).[n]$$

Translation process. Given a PKB $\mathcal{K}$ in *DL-Lite*$_\mathcal{R}$ normal form, a program $PK(\mathcal{K})$ that encodes query answering for $\mathcal{K}$ can be encoded as:

$$PK(\mathcal{K}) = I_{dlr}(\mathcal{K}) \cup I_{\mathrm{D}}(\mathcal{K}) \cup I_{\mathcal{P}}(\mathcal{K}) \cup P_{dlr} \cup P_{\mathrm{D}} \cup P_{pref}$$

For completeness of the *DL-Lite*$_\mathcal{R}$ translation (and in particular to reason on roles for negation of existential axioms), $PK(\mathcal{K})$ is completed with a set of supporting facts about constants: for every literal $\texttt{nom}(c)$ or $\texttt{supEx}(a, r, c)$ in $PK(\mathcal{K})$, $\texttt{const}(c)$ is added to $PK(\mathcal{K})$. Then, given an arbitrary enumeration $c_0, \dots, c_n$ s.t. each $\texttt{const}(c_i) \in PK(\mathcal{K})$, the facts $\texttt{first}(c_0)$, $\texttt{last}(c_n)$ and $\texttt{next}(c_i, c_{i+1})$ with $0 \le i < n$ are added to $PK(\mathcal{K})$.

Query answering $\mathcal{K} \models \alpha$ is then obtained by testing whether the (instance) query, translated to datalog by $O(\alpha)$, is a consequence of $PK(\mathcal{K})$, i.e., whether $PK(\mathcal{K}) \models O(\alpha)$ holds.

Example 3. *We can encode in ASP the example PKB introduced in Example 1 using the presented translation. In particular, by applying the DL-Lite*$_\mathcal{R}$ *input rules* I_{dlr} *on the knowledge part of the example PKB* $\mathcal{K}$ *(in normal form) we obtain the following facts:*

$\mathtt{supNot}(nottrusted, trusted).$
$\mathtt{supNot}(trusted, nottrusted).$
$\mathtt{subClass}(dog, hasLegs).$
$\mathtt{subClass}(wolf, hasLegs).$

$\mathtt{insta}(balto, dog).$
$\mathtt{insta}(balto, wolf).$
$\mathtt{insta}(pluto, dog).$
$\mathtt{insta}(alberto, wolf).$
$\mathtt{insta}(cerberus, dog).$

$\mathtt{insta}(balto, livesInWoods).$
$\mathtt{insta}(balto, hasLegs).$
$\mathtt{insta}(balto, isTamed).$

$\mathtt{insta}(pluto, hasCollar).$
$\mathtt{insta}(pluto, hasLegs).$
$\mathtt{insta}(pluto, isTamed).$

$\mathtt{insta}(alberto, hasLegs).$
$\mathtt{insta}(alberto, hunts).$
$\mathtt{insta}(cerberus, nottrusted).$

Input rules for prototype axioms I_{D} applied to $\mathcal{K}$ produce the following facts, encoding the two prototype axioms of our example:

$$\mathtt{def\_subclass}(dog, trusted). \qquad \mathtt{def\_subclass}(wolf, nottrusted).$$

Then, input rules $I_{\mathcal{P}}$ for prototype axioms add facts about the prototypes and prototype description scores in $\mathcal{P}$ to the program (considering here a normalized integer version of the scores):

$\mathtt{isProto}(dog).$
$\mathtt{isProto}(wolf).$

$\mathtt{featwt}(dog, hasCollar, 3).$
$\mathtt{featwt}(dog, livesInHouse, 2).$
$\mathtt{featwt}(dog, hasLegs, 1).$
$\mathtt{featwt}(dog, isTamed, 4).$

$\mathtt{featwt}(wolf, livesInWoods, 3).$
$\mathtt{featwt}(wolf, hasLegs, 1).$
$\mathtt{featwt}(wolf, livesInPack, 2).$
$\mathtt{featwt}(wolf, hunts, 4).$

The program $PK(\mathcal{K})$ is then completed by the deduction rules $P_{dlr} \cup P_{\mathrm{D}} \cup P_{pref}$ and the supporting facts as by its definition above.

If we solve the resulting program, consistently with what we have shown in the semantics, we obtain two "candidate" answer sets: S_1 that contains $\mathtt{ovr}(balto, wolf, nottrusted)$, $\mathtt{ovr}(cerberus, dog, trusted)$ and S_2 containing $\mathtt{ovr}(balto, dog, trusted)$, $\mathtt{ovr}(cerberus, dog, trusted)$. By applying the rules for the computation of scores, in S_1 we have that $\mathtt{ovrscore}(balto, wolf, nottrusted, 4)$ and in S_2 we have $\mathtt{ovrscore}(balto, dog, trusted, 5)$, while both contain $\mathtt{ovrscore}(cerberus, dog, trusted, 1)$. Then, as expected by our preference, we have that the weak constraint prefers the answer set with the "less expensive" set of overridings: thus, the preferred answer set is S_1. $\diamond$

3.2. Correctness

In the following, we show that the presented ASP encoding provides a sound and complete materialization calculus for *DL-Lite*$_{\mathcal{R}}$ PKBs in normal form (with the assumption of integer and normalized input weights). As in [9], in the translation we assume UNA on elements of $\mathcal{K}$ and consider *named models*, models restricted to $sk(N_{\mathcal{K}})$, i.e. all constants that occur in $\mathcal{K}$ and their Skolem constants. We show the correctness result on the *least model* of $\mathcal{K}$ with respect to an exception assumptions set χ, denoted with $\hat{\mathcal{I}}(\chi)$.

Let $\mathcal{I}_{\boldsymbol{\chi}} = \langle \mathcal{I}, \chi \rangle$ be a justified named $\boldsymbol{\chi}$-model: we define the set of overriding assumptions $OVR(\mathcal{I}_{\boldsymbol{\chi}}) = \{\, \mathtt{ovr}(e, P, D) \mid \langle P \sqsubseteq D, e \rangle \in \chi \,\}$. Given a $\boldsymbol{\chi}$-interpretation $\mathcal{I}_{\boldsymbol{\chi}}$ and the input PKB $\mathcal{K}$, we can define a corresponding interpretation $S = I(\mathcal{I}_{\boldsymbol{\chi}})$ for $PK(\mathcal{K})$: the construction of S extends the one used in [9] to the interpretation of the preference rules.

(1). $l \in S$, if $l \in PK(\mathcal{K})$;

(2). $\mathtt{instd}(a, A) \in S$, if $\mathcal{I} \models A(a)$ and $\neg\mathtt{instd}(a, A) \in S$, if $\mathcal{I} \models \neg A(a)$;

(3). $\mathtt{tripled}(a, R, b) \in S$, if $\mathcal{I} \models R(a, b)$ and $\neg\mathtt{tripled}(a, R, b) \in S$, if $\mathcal{I} \models \neg R(a, b)$;

(4). $\mathtt{tripled}(a, R, aux^{\alpha}) \in S$, if $\mathcal{I} \models \exists R(a)$ for $\alpha = A \sqsubseteq \exists R$;

(5). $\mathtt{all\_nrel}(a, R) \in S$ if $\mathcal{I} \models \neg\exists R(a)$;

(6). $\mathtt{ovr}(e, P, D) \in S$, if $\mathtt{ovr}(e, P, D) \in OVR(\mathcal{I}_{\boldsymbol{\chi}})$;

(7). $\{\texttt{isProto}(P), \texttt{featwt}(P, C_1, w_1), \dots, \texttt{featwt}(P, C_n, w_n)\} \subseteq S$, if $P(C_1 : w_1, \dots, C_n : w_n) \in \mathcal{P}$;

(8). $\texttt{addScore}(e, P, w) \in S$, if $\mathcal{I} \models P(e), C(e)$ and $\texttt{featwt}(P, C, w) \in S$;

(9). $\texttt{score}(e, P, n) \in S$, if $n = \mathit{score}_P(e)$;

(10). $\texttt{ovrscore}(e, P, D, n) \in S$, if $\texttt{ovr}(e, P, D) \in S$ and $\texttt{score}(e, P, n) \in S$.

The next proposition shows that the least models of $\mathcal{K}$ can be represented by the answer sets of the program $PK(\mathcal{K})$. Considering that the translation presented for *DL-Lite*$_\mathcal{R}$ is simply a restriction to the use of the encoding proposed in [9], and given the analogous interpretation of defeasible axioms and their justification, we clearly inherit their similar result (Proposition 7) for our translation.

Lemma 1. *Let $\mathcal{K}$ be a PKB in DL-Lite$_\mathcal{R}$ normal form with normalized integer scores and assuming that scores are stable. Then:*

(i). for every (named) justified exception assumption set χ, the interpretation $S = I(\hat{\mathcal{I}}(\chi))$ is an answer set of $PK(\mathcal{K})$;

(ii). every answer set S of $PK(\mathcal{K})$ is of the form $S = I(\hat{\mathcal{I}}(\chi))$ where χ is a (named) justified exception assumption set for $\mathcal{K}$.

Proof (Sketch). Intuitively, since the newly added preference rules play a minor role in the computation of such answer sets (they only add information from $\mathcal{P}$ and additional facts about scores), the result can be proved analogously to [9, Proposition 7] in original *DL-Lite*$_\mathcal{R}$ translation. This is obtained by showing that the answer sets of (the rules part of) $PK(\mathcal{K})$ correspond to the sets $S = I(\hat{\mathcal{I}}(\chi))$ with χ a justified exception assumption set for $\mathcal{K}$, using the construction of S detailed above. More in detail: *(i)* is proved by showing that $S = I(\hat{\mathcal{I}}(\chi))$ built from a justified χ satisfies $S \models PK(\mathcal{K})$ and S is minimal with respect to the reduct on `ovr` NAF-literals; on the other hand, *(ii)* can be shown by considering any answer set S of $PK(\mathcal{K})$ and building a justified $\boldsymbol{\chi}$-model $\mathcal{I}_S$ for $\mathcal{K}$ such that $S = I(\mathcal{I}_S) = I(\hat{\mathcal{I}}(\chi))$ holds. □

The correspondence with PKB models is then obtained by considering the interpretation of preference on answer sets defined by weak constraints (which minimize the score of overridings) and the definition of preference SimpleMDP on $\boldsymbol{\chi}$-models in the semantics.

Lemma 2. *Let $\mathcal{K}$ be a PKB in DL-Lite$_\mathcal{R}$ normal form with normalized integer scores and assuming that scores are stable.*

Then, $\hat{\mathcal{I}}$ is a PKB model of $\mathcal{K}$ iff there exists a (named) justified exception assumption set $\overline{\chi}$ s.t. $I(\hat{\mathcal{I}}(\overline{\chi}))$ is an optimal answer set of $PK(\mathcal{K})$.

Proof. The claim is proved by showing that $\hat{\mathcal{I}}$ is a PKB model iff:

(i). there exists a (named) justified clashing assumption $\overline{\chi}$ s.t. $I(\hat{\mathcal{I}}(\overline{\chi}))$ is an answer set of $PK(\mathcal{K})$;

(ii). $I(\hat{\mathcal{I}}(\overline{\chi}))$ is an optimal answer set of $PK(\mathcal{K})$.

Condition (i) is directly verified from Lemma 1 and the definition of PKB model.

To prove (ii), we have to show the correspondence of the SimpleMDP preference on exception assumption sets with the order induced by the objective function $H^{PK(\mathcal{K})}(S)$ on answer sets. In other words, $I(\hat{\mathcal{I}}(\overline{\chi}))$ is optimal iff there does not exist a justified χ' s.t. $\chi' > \overline{\chi}$ (with respect to SimpleMDP). ($\Leftarrow$) In one direction, suppose that $\overline{\chi}$ is preferred, that is there does not exist a justified χ' s.t. $\chi' > \overline{\chi}$. This means that for every such χ', χ' is either preferred (and thus, intuitively, with the same "cost" of $\overline{\chi}$) or we have $\chi' < \overline{\chi}$. By the definition of SimpleMDP, in this case we have then that for every $\langle P \sqsubseteq D, e\rangle \in \overline{\chi} \setminus \chi'$, there exists a $\langle Q \sqsubseteq E, f\rangle \in \chi' \setminus \overline{\chi}$ with $\mathit{score}_P(e) < \mathit{score}_Q(f)$. This means that in χ' there exists at least an "additional" $\langle Q \sqsubseteq E, f\rangle$ such that $\mathit{score}_Q(f)$ is larger than any $\mathit{score}_P(e)$ for the $\langle P \sqsubseteq D, e\rangle$ in $\overline{\chi}$. Considering now the interpretation $S' = I(\hat{\mathcal{I}}(\chi'))$, we can show that it has necessarily an higher cost with respect to $S = I(\hat{\mathcal{I}}(\overline{\chi}))$. First, we note that weak constraints are only associated to instances of `ovrscore` atoms: these atoms only depend on the instantiations of the `score` atoms, that are computed by the (dpref-as) and (dpref-sc) rules and the same for all answer sets (as for the stable scores assumption), and the overriding atoms `ovr`. The optimization of the answer sets is thus only dependent on minimization of aspects related on the score of such atoms (which, on the other hand, are related to the scores of exception assumptions). Since $\langle Q \sqsubseteq E, f\rangle \in \chi'$, by construction of S' we have that the corresponding $\mathtt{ovr}(f, Q, E) \in S'$ and $\mathtt{ovrscore}(f, Q, E, m) \in S'$ with $m = score_Q(f)$. Thus, we have an instantiation of the weak constraint rule (dpref-wc) relative to $\mathtt{ovrscore}(f, Q, E, m)$, which adds a weak constraint violation to S' with cost m. Similarly, considering the $\langle P \sqsubseteq D, e\rangle \in \overline{\chi}$ above, the construction of S adds $\mathtt{ovr}(e, P, D) \in S'$ and $\mathtt{ovrscore}(e, P, D, n) \in S'$ with $n = score_P(e)$: we know, by the preference, that it holds $m > n$. Now, considering the definition of the optimization function $H^{PK(\mathcal{K})}$ from [18], since the violation in S' is at the same level of the violation in S, then the cost m of the violation (knowing that it is larger than the costs of corresponding violations in S) assures that the cost of violations in S' is larger than those in S. Thus, we have that $H^{PK(\mathcal{K})}(S') > H^{PK(\mathcal{K})}(S)$. This shows the optimality of $I(\hat{\mathcal{I}}(\overline{\chi}))$.

($\Rightarrow$) The other direction can be shown with a similar reasoning: supposing that $S = I(\hat{\mathcal{I}}(\overline{\chi}))$ is optimal, then for all other $S' = I(\hat{\mathcal{I}}(\chi'))$ we have $H^{PK(\mathcal{K})}(S') > H^{PK(\mathcal{K})}(S)$. Thus, by the definition of the optimization function, we have that there exists at least a violation on a $\mathtt{ovrscore}(f, Q, E, m)$ causing an higher cost with respect to S. Considering the corresponding exception assumption sets, we can map to the definition of their preference: there need to exist $\langle Q \sqsubseteq E, f\rangle \in \chi' \setminus \overline{\chi}$ with $score_Q(f)$ that is larger than any $score_P(e)$ for $\langle P \sqsubseteq D, e\rangle \in \overline{\chi} \setminus \chi'$. This corresponds to the definition of the SimpleMDP preference: thus, $\overline{\chi}$ is preferred and we proved the result. □

The correctness of the translation with respect to instance checking is then a direct consequence of the previous results.

Theorem 1. *Let $\mathcal{K}$ be a PKB in DL-Lite$_\mathcal{R}$ normal form with normalized integer scores and assuming that scores are stable. Let $\alpha \in \mathcal{L}_\Sigma$ s.t. $O(\alpha)$ is defined: then, $\mathcal{K} \models \alpha$ iff $PK(\mathcal{K}) \models O(\alpha)$.*

4. Related Work

Regarding the formalization of DLs with Prototype Descriptions, our work can be compared to approaches for non-monotonic DLs like [3, 4]: these approaches are inspired by the historical work on defeasible reasoning in propositional logic presented in [19, 20], where formal properties, known as *KLM properties*, have been introduced as properties that any non-monotonic logic should satisfy. Moreover, implicitly or explicitly they rely on a notion of *typicality* for explaining the defeasibility of their systems. Of particular interest for our work are formalisms developed starting from [3], which use weights and have a multi-preferential relation over the individuals with respect to the concepts they are instances of, as, for instance, [21, 22]. Despite these commonalities, differently from such systems our approach does not introduce new operators or logical connectives to identify defeasible axioms: this allows to apply the non-monotonic mechanism by simply extending the classical DL knowledge base with the needed prototype descriptions. Moreover, in our approach the preference relations hold among models and are determined by reasoning on knowledge, while in the aforementioned works the ordering regards the elements of the domain.

Regarding the ASP translation, we note that reasoning by rewriting DLs in ASP programs has been adopted also in other approaches for non-monotonic DLs like [22], in particular for taking advantage of the non-monotonic features and declarative definition of answer set programs. The ASP encoding presented in this paper derives from the ASP rewriting proposed in [8] for defeasible reasoning in *Contextualized Knowledge Repositories (CKR)* with Justifiable Exceptions. We note that in [23] this translation was already extended with weak constraints to model a similar notion of preference: however, in [23] preference was not defined on "scores" derivable from the KB, but with respect to a fixed contextual structure. In [24], such answer set preference was further refined, using an extension of ASP, to allow for reasoning with more general contextual structures.

5. Conclusions and Future Work

In this paper, we presented an ASP encoding enabling reasoning on DLs with Prototype Descriptions, in particular in the *DL-Lite*$_\mathcal{R}$ language. The encoding extends the ASP translation for *DL-Lite*$_\mathcal{R}$ with Justifiable Exception from [9] in order to consider the model preference defined by prototype scores. By considering reasonable assumptions on the input PKB and a basic form of model preference, we have shown that our ASP translation is complete with respect to instance checking in *DL-Lite*$_\mathcal{R}$ prototype knowledge bases. In particular, our contribution shows that an ASP encoding of PKBs based only on standard ASP constructs (and thus, readily usable over standard ASP solvers) is already enough to reason in our formalism with a simple case of preference induced by prototype scores. Moreover, since rules encoding answer set preference are distinct from the rules encoding the contents of the PKB, the proposed translation provides a foundation for including more complex preferences (or, on the other hand, consider more expressive DL languages like, e.g., $\mathcal{SROIQ}$-RL [8]).

The next steps in further developing a reasoning method for DLs with Prototype Descriptions will thus include the extension of our translation for encoding more general criteria for preference, as the preferences based on *model dependent* and *model independent* scores presented in [10]. In this regard, more advanced tools for answer set preferences can be adopted. For example, a similar model preference has been encoded using Asprin in [24], but more general comparisons across answer sets could also be obtained with ASP with *algebraic measures* [25]. Another natural direction for our work is the implementation of the ASP translation: one easy way to achieve this is to extend the CKR*ew* datalog rewriter[3] used in [9] for the *DL-Lite*$_\mathcal{R}$ translation by implementing the rewriting of rules for the adopted preference. On the other hand, we are currently continuing the study of the formalization of DLs with Prototype Descriptions: for example, while in this paper we assumed independence of scores across features, we are currently studying the options for representing dependencies between features and the impact of this on the evaluation of weights. Future proposals for reasoning methods will then need to be adapted to consider such further developments of the formalism.

Acknowledgments

We acknowledge the financial support through the 'Abstractron' project funded by the Autonome Provinz Bozen - Südtirol (Autonomous Province of Bolzano-Bozen) through the Research Südtirol/Alto Adige 2022 Call.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] C. Strasser, G. A. Antonelli, Non-monotonic Logic, in: E. N. Zalta (Ed.), The Stanford Encyclopedia of Philosophy, Summer 2019 ed., Metaphysics Research Lab, Stanford University, 2019.

[2] J. McCarthy, P. J. Hayes, Some Philosophical Problems from the Standpoint of Artificial Intelligence, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1987, p. 26–45.

[3] L. Giordano, V. Gliozzi, N. Olivetti, G. Pozzato, Semantic characterization of rational closure: From propositional logic to description logics, Artificial Intelligence 226 (2015) 1–33. URL: https://www.

[3] https://ckrew.fbk.eu/

sciencedirect.com/science/article/pii/S0004370215000673. doi:https://doi.org/10.1016/j.artint.2015.05.001.

[4] K. Britz, G. Casini, T. Meyer, K. Moodley, U. Sattler, I. Varzinczak, Principles of klm-style defeasible description logics, ACM Trans. Comput. Logic 22 (2020). URL: https://doi.org/10.1145/3420258. doi:10.1145/3420258.

[5] G. Sacco, L. Bozzato, O. Kutz, Defeasible reasoning with prototype descriptions: First steps, in: Proceedings of the 36th International Workshop on Description Logics (DL 2023), volume 3515 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2023.

[6] G. Sacco, L. Bozzato, O. Kutz, Generics in defeasible reasoning. exceptionality, gradability, and content sensitivity, in: 7th CAOS Workshop 'Cognition and Ontologies', 9th Joint Ontology Workshops (JOWO 2023), co-located with FOIS 2023, 19-20 July, 2023, Sherbrooke, Québec, Canada, volume 3637 of *CEUR-WS*, 2023.

[7] G. Righetti, P. Galliani, O. Kutz, D. Porello, C. Masolo, N. Troquard, Weighted Description Logic for Classification Problems, in: D. Calvanese, L. Iocchi (Eds.), GCAI 2019. Proceedings of the 5th Global Conference on Artificial Intelligence, volume 65 of *EPiC Series in Computing*, EasyChair, 2019, pp. 108–112. URL: https://easychair.org/publications/paper/S5bV. doi:10.29007/vd1q.

[8] L. Bozzato, T. Eiter, L. Serafini, Enhancing context knowledge repositories with justifiable exceptions, Artif. Intell. 257 (2018) 72–126.

[9] L. Bozzato, T. Eiter, L. Serafini, Reasoning on *DL-Lite*$_{\mathcal{R}}$ with defeasibility in ASP, Theory Pract. Log. Program. 22 (2022) 254–304.

[10] G. Sacco, L. Bozzato, O. Kutz, Defeasible reasoning with prototype descriptions: A new preference order, in: L. Giordano, J. C. Jung, A. Ozaki (Eds.), Proceedings of the 37th International Workshop on Description Logics (DL 2024), Bergen, Norway, June 18-21, 2024, volume 3739 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2024. URL: https://ceur-ws.org/Vol-3739/paper-9.pdf.

[11] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, R. Rosati, Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family, J. Automated Reasoning 39 (2007) 385–429.

[12] G. Brewka, J. P. Delgrande, J. Romero, T. Schaub, asprin: Customizing answer set preferences without a headache, in: B. Bonet, S. Koenig (Eds.), Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA, AAAI Press, 2015, pp. 1467–1474. URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9535.

[13] P. Galliani, G. Righetti, O. Kutz, D. Porello, N. Troquard, Perceptron connectives in knowledge representation, in: C. M. Keet, M. Dumontier (Eds.), Knowledge Engineering and Knowledge Management, Springer International Publishing, Cham, 2020, pp. 183–193.

[14] V. W. Marek, M. Truszczynski, Nonmonotonic logic - context-dependent reasoning, Artificial intelligence, Springer, 1993.

[15] F. Pacenza, J. Zangari, Extending answer set programming with rational numbers, CoRR abs/2312.04249 (2023).

[16] D. Porello, G. Righetti, N. Troquard, R. Confalonieri, O. Kutz, An Ontological Modelling of Prototype Theories, in: C. Beierle, K. Sauerwald, F. Schwarzentruber, F. Stolzenburg (Eds.), Proceedings of the 9th Workshop on Formal and Cognitive Reasoning, Berlin, Germany, September 26, 2023, volume 3500 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2023, pp. 20–31. URL: https://ceur-ws.org/Vol-3500/paper1.pdf.

[17] M. Krötzsch, Efficient inferencing for OWL EL, in: T. Janhunen, I. Niemelä (Eds.), Logics in Artificial Intelligence - 12th European Conference, (JELIA 2010), volume 6341 of *LNCS*, Springer, 2010, pp. 234–246.

[18] N. Leone, G. Pfeifer, W. Faber, T. Eiter, G. Gottlob, S. Perri, F. Scarcello, The DLV system for knowledge representation and reasoning, CoRR cs.AI/0211004 (2002). URL: http://arxiv.org/abs/cs.AI/0211004.

[19] S. Kraus, D. Lehmann, M. Magidor, Nonmonotonic reasoning, preferential models and cumulative logics, Artificial Intelligence 44 (1990) 167–207. URL: https://www.sciencedirect.com/science/article/pii/0004370290901015. doi:https://doi.org/10.1016/0004-3702(90)90101-5.

[20] D. Lehmann, M. Magidor, What does a conditional knowledge base entail?, Artificial Intelligence 55 (1992) 1–60. URL: https://www.sciencedirect.com/science/article/pii/000437029290041U. doi:https://doi.org/10.1016/0004-3702(92)90041-U.

[21] L. Giordano, D. Theseider Dupré, Weighted defeasible knowledge bases and a multipreference semantics for a deep neural network model, in: Logics in Artificial Intelligence: 17th European Conference, JELIA 2021, Virtual Event, May 17–20, 2021, Proceedings 17, Springer, 2021, pp. 225–242.

[22] L. Giordano, D. Theseider Dupré, An ASP approach for reasoning on neural networks under a finitely many-valued semantics for weighted conditional knowledge bases, Theory and Practice of Logic Programming 22 (2022) 589–605. doi:10.1017/S1471068422000163.

[23] L. Bozzato, L. Serafini, T. Eiter, Reasoning with justifiable exceptions in contextual hierarchies, in: M. Thielscher, F. Toni, F. Wolter (Eds.), Principles of Knowledge Representation and Reasoning: Proceedings of the Sixteenth International Conference, KR 2018, Tempe, Arizona, 30 October - 2 November 2018, AAAI Press, 2018, pp. 329–338. URL: https://aaai.org/ocs/index.php/KR/KR18/paper/view/18032.

[24] L. Bozzato, T. Eiter, R. Kiesel, Reasoning on multirelational contextual hierarchies via answer set programming with algebraic measures, Theory Pract. Log. Program. 21 (2021) 593–609. URL: https://doi.org/10.1017/S1471068421000284. doi:10.1017/S1471068421000284.

[25] T. Eiter, R. Kiesel, Weighted LARS for quantitative stream reasoning, in: G. D. Giacomo, A. Catalá, B. Dilkina, M. Milano, S. Barro, A. Bugarín, J. Lang (Eds.), ECAI 2020 - 24th European Conference on Artificial Intelligence, 29 August-8 September 2020, Santiago de Compostela, Spain, August 29 - September 8, 2020 - Including 10th Conference on Prestigious Applications of Artificial Intelligence (PAIS 2020), volume 325 of *Frontiers in Artificial Intelligence and Applications*, IOS Press, 2020, pp. 729–736. URL: https://doi.org/10.3233/FAIA200160. doi:10.3233/FAIA200160.