Reasoning in the presence of inconsistency through Preferential $\mathcal{ALC}$

Graham Deane, Krysia Broda, and Alessandra Russo

Imperial College London {graham.deane10,k.broda,a.russo}@imperial.ac.uk

Abstract

This paper presents an inconsistency tolerant semantics for the Description Logic $\mathcal{ALC}$ called Preferential $\mathcal{ALC}$ ($p$-$\mathcal{ALC}$). A $p$-$\mathcal{ALC}$ knowledge base is comprised of defeasible and non-defeasible axioms. The defeasible ABox and TBox are labelled with confidence weights that could reflect an axiom’s provenance. Entailment is defined through the notion of preferred interpretations which minimise the total weight of the inconsistent axioms. We introduce a modified $\mathcal{ALC}$ tableau algorithm in which the open branches give rise to the preferred interpretations, and show that it can compute $p$-$\mathcal{ALC}$ entailment by refutation. The modified algorithm is implemented as an incremental answer set program (ASP) that exploits optimisation to capture preferred interpretations of $p$-$\mathcal{ALC}$.

1 Introduction

Web Ontology Languages [25] (OWL) are based on Description Logics, a family of decidable logics that offer low complexity of reasoning. Popular reasoners such as Hermit [13] and Fact++ [24] assume Tarskian model theoretic semantics and thus require a consistent Knowledge Base (KB). However, for many applications, such as multi-agent systems, knowledge is collated from multiple sources and inconsistencies inevitably arise. Inconsistencies within a KB may come about in any of the three basic settings: direct conflicts between ABox axioms (e.g. $\text{Woman}(\text{alex})$ and $\neg\text{Woman}(\text{alex})$); contradictions between TBox and ABox axioms (e.g. $\text{Woman}(\text{alex})$, $\neg\text{Female}(\text{alex})$ and $\text{Woman} \sqsubseteq \text{Female}$); and potentially conflicting TBox axioms (e.g. a KB including the TBox axioms $\text{Woman} \sqsubseteq \text{Human}$, $\text{Human} \sqsubseteq \text{Animal}$, $\text{Woman} \sqsubseteq \neg\text{Animal}$ can become inconsistent once an individual classified as a $\text{Woman}$ is added to it).

The inevitability of inconsistency in Knowledge Bases has inspired the search for alternative solutions that allow reasoning without having to locate and repair the inconsistencies (e.g. [17]). In the literature, two strands of work have been investigated. On one hand, a notion of “typicality” has been proposed for which KBs are designed under the assumption that certain knowledge is typically true but admits “exceptions”. Languages are augmented with defeasible implication and supported by some form of non-classical semantics [3, 4, 5, 8, 9, 10, 12, 15, 22]. On the other hand, inconsistency tolerant semantics have been proposed for KBs assumed to be designed for and used by classical reasoners [16, 19, 20, 21, 23, 26]. An example is [19], where entailment is defined in terms of $\textit{ABox closed repair semantics}$, which consider maximally consistent consequences derivable from subsets of ABox axioms that are consistent with

TBox axioms. This method, however, does not provide for “arbitration” between conflicting consequences, limiting therefore the set of consequences that could be derived from an inconsistent KB. This limitation is even more evident when such a notion of entailment is extended to TBox repairs [20], because each TBox axiom is treated atomically, it is either included in the inference process or omitted.

This paper follows the second strand of research and proposes an alternative semantics to the problem of reasoning in the presence of inconsistent KBs that does not suffer from the above limitations. A KB is defined as comprising of defeasible and non-defeasible axioms. Each defeasible axiom is labelled with an integer weight that expresses the degree of confidence in that axiom. A preferential semantics, called Preferential ALC and denoted as p-ALC, is proposed, which defines entailment through a notion of preferred interpretations. These are interpretations that preferentially retain the defeasible knowledge in which there is the greatest confidence and use the weights to inform the inference of consequences from conflicting axioms. A tableau-based algorithm, which combines techniques used in DL tableau methods, is also proposed for determining what is provable or not from a given inconsistent KB within our new p-ALC semantics. A prototype implementation of our tableau-based algorithm has been developed using a state-of-the-art incremental answer set solver [11], which exploits the optimisation features of Answer Set Programming (ASP) to incrementally compute preferred interpretations. Preliminary evaluations of our prototype have been conducted to demonstrate the applicability of our approach.

The paper is structured as follows. Section 3 introduces our new notion of preferential semantics. Section 4 presents our tableau-based algorithm for computing consequences in the presence of inconsistencies with respect to the preferential semantics, and Section 5 describes the implementation of our algorithm in Answer Set Programming (ASP). Evaluation and related work are discussed in Sections 6 and 7, while Section 8 concludes the paper and outlines future work.

2 Background

In this section we recall the syntax and semantics of the Description Logic ALC. An ALC signature is a tuple \( \langle N_I, N_R, N_C \rangle \) where \( N_I \), \( N_R \), \( N_C \) are finite sets of names that refer, respectively, to individuals, roles and named concepts. Given a signature, concepts are either named concepts or complex concepts defined inductively as follows: \( C, D \equiv \top, \bot \mid \mid \mid A \mid \neg(C \cap D) \mid C \sqcup D \mid \exists R.C \mid \forall R.C \), where \( A \in N_C, R \in N_R \), and \( \top \) and \( \bot \) denote the top and bottom concept. An ABox (resp. TBox) is a finite set of axioms of the form \( C(x) \) or \( R(x, y) \) (resp. \( C \subseteq D \) or \( C \equiv D \)) where \( x, y \in N_I, R \in N_R, C, D \) are concepts and \( C \equiv D \) abbreviates the set of axioms \( C \subseteq D \) and \( D \subseteq C \). A knowledge base \( \mathcal{K} \) is defined as a tuple \( \langle \mathcal{A}, \mathcal{T} \rangle \), where \( \mathcal{A} \) is the ABox of \( \mathcal{K} \) and \( \mathcal{T} \) is the TBox of \( \mathcal{K} \). The set of all concepts that can be formed using the signature of \( \mathcal{K} \) is referred to as the language of \( \mathcal{K} \).

An interpretation of a knowledge base \( \mathcal{K} \) is a pair \( \mathcal{I} = \langle \Delta^\mathcal{I}, \cdot^\mathcal{I} \rangle \), where \( \Delta^\mathcal{I} \) is a non-empty set, called domain of the interpretation, and \( \cdot^\mathcal{I} \) is an interpretation function. The function \( \cdot^\mathcal{I} \) interprets each individual \( x \), in the signature, as \( x^\mathcal{I} \in \Delta^\mathcal{I} \), each named concept \( C \), in \( N_C \), as \( C^\mathcal{I} \subseteq \Delta^\mathcal{I} \), and each role \( R \), in \( N_R \), as \( R^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I} \). It is extended inductively over all concepts that can be formed from a given signature: \( \top^\mathcal{I} = \Delta^\mathcal{I}, \bot^\mathcal{I} = \emptyset, (\neg C)^\mathcal{I} = \Delta^\mathcal{I} \setminus C^\mathcal{I}, (C \cap D)^\mathcal{I} = C^\mathcal{I} \cap D^\mathcal{I}, (C \cup D)^\mathcal{I} = C^\mathcal{I} \cup D^\mathcal{I}, (\forall R.C)^\mathcal{I} = \{ u \in \Delta^\mathcal{I} \mid \forall v \ [u, v] \in R^\mathcal{I} \rightarrow v \in C^\mathcal{I} \} \), \( (\exists R.C)^\mathcal{I} = \{ u \in \Delta^\mathcal{I} \mid \exists v \ [u, v] \in R^\mathcal{I} \land v \in C^\mathcal{I} \} \). Given a knowledge base \( \mathcal{K} \), an axiom \( Z \) in \( \mathcal{K} \) is said to be satisfied by an interpretation \( \mathcal{I} \) if and only if one of the following cases holds: (i) \( Z = C(x) \) then \( x^\mathcal{I} \in C^\mathcal{I} \); (ii) \( Z = R(x, y) \) then \( (x^\mathcal{I}, y^\mathcal{I}) \in R^\mathcal{I} \); (iii) \( Z = C \subseteq D \) then \( C^\mathcal{I} \subseteq D^\mathcal{I} \).
An interpretation $\mathcal{I}$ is said to be a model of a knowledge base $\mathcal{K}$ if every axiom in $\mathcal{K}$ is satisfied by $\mathcal{I}$. We also say, in this case, that $\mathcal{K}$ is satisfied in $\mathcal{I}$. If there is no model of $\mathcal{K}$ then $\mathcal{K}$ is said to be inconsistent. Given a knowledge base $\mathcal{K}$ and an axiom $Z$, $\mathcal{K}$ is said to entail $Z$, written $\mathcal{K} \models Z$, if and only if $Z$ is satisfied in every model of $\mathcal{K}$.

In this paper we assume uniqueness of names, namely for every interpretation $\mathcal{I}$ each pair of names $x, y \in N_1$ satisfies $x^\mathcal{I} \neq y^\mathcal{I}$. Without loss of generality, we assume that all axioms are given in negation normal form (NNF), i.e. negation appears only in front of named concepts. Following [2], we will use the notation $\neg C$ to indicate that $\neg C$ is written in NNF. In addition, we also assume that in concepts of the form $\exists x, y \; C$, any complex concept $C$ within the scope of a quantifier, can be reduced to our assumed simpler form by substituting it with a new named concept, say $C_n$, and adding the axiom $C_n \equiv C$ to the TBox.

## 3 Preferential ALC

We can now introduce the semantics of our preferential ALC. To accommodate inconsistency we introduce a notion of defeasible axioms. Intuitively, these are axioms in the knowledge base that we are prepared to falsify during the reasoning process. A defeasible axiom is defined as an ALC axiom with an associated positive integer weight $w$ that indicates a measure of confidence in the truth of the axiom. The higher the value the greater the level of confidence. The notation $Z^{[w]}$ denotes a defeasible axiom with weight $w$. Given a set $S$ of defeasible axioms, the notation $S^{-W}$ will be used to refer to $S^{-W} = \{ Z | Z^{[w]} \in S \}$.

**Definition 3.1.** A p-ALC knowledge base $\mathcal{K}$ is a tuple $\mathcal{K} = \langle A, T, A_d, T_d \rangle$, where $A$ and $T$ are, respectively, two sets of ABox and TBox ALC axioms, and $A_d$ and $T_d$ are, respectively, two sets of defeasible ABox and TBox axioms such that $A$ and $A_d$ are disjoint and $T$ and $T_d$ are also disjoint.

In general, weights of the defeasible axioms do not have to be equal, but in the case where they have equal weights a p-ALC knowledge is said to be uniform. Furthermore, p-ALC knowledge bases for which $\langle A, T \rangle$ is satisfiable are said to be credible. From now on we will consider only credible p-ALC knowledge bases unless otherwise stated.

The semantics of p-ALC extends the notion of ALC interpretations to the defeasible axioms and by introducing a notion of distance of the interpretation.

**Definition 3.2.** Let $\mathcal{K} = \langle A, T, A_d, T_d \rangle$ be a p-ALC knowledge base. A (p-ALC) interpretation of $\mathcal{K}$ is a classical ALC interpretation $\mathcal{I} = \langle \Delta^\mathcal{I}, \mathcal{T}^\mathcal{I} \rangle$ of $\langle A \cup A_d^{[w]}, T \cup T_d^{[w]} \rangle$. Let $Z$ be a defeasible axiom in $\mathcal{K}$, $Z \in A_d \cup T_d$. The set of unsatisfied instances of $Z$ with respect to $\mathcal{I}$, denoted as $U(Z, \mathcal{I})$, is defined as follows:

<table>
<thead>
<tr>
<th>$U(Z, \mathcal{I})$</th>
<th>where $Z$ is $\mathcal{C}(x)^{[w]}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>${ (x^\mathcal{I}, x^\mathcal{I}) }$</td>
<td>if $C(x)^\mathcal{I} \notin \mathcal{I}$</td>
</tr>
<tr>
<td>${ (R(x, y)^{[w]}, x^\mathcal{I}) }$</td>
<td>if $R(x, y)^\mathcal{I} \notin \mathcal{I}$</td>
</tr>
<tr>
<td>${ (C \subseteq D^{[w]}, u)</td>
<td>(u \in C^\mathcal{I}) \land (u \notin D^\mathcal{I}) }$</td>
</tr>
</tbody>
</table>

The set of unsatisfied instances of defeasible axioms in $\mathcal{K}$ with respect to $\mathcal{I}$, denoted as $\mathcal{U}(\mathcal{K}, \mathcal{I})$, is defined as

$$\mathcal{U}(\mathcal{K}, \mathcal{I}) = \bigcup_{Z \in A_d \cup T_d} U(Z, \mathcal{I}).$$

\[\text{For the form } R(x, y)^{[w]}, y^\mathcal{I} \text{ is not required to define a unique axiom instance.}\]
The distance of the interpretation $I$, denoted as $d(U(K,I))$, is then given by

$$d(U(K,I)) = \sum_{(I,K) \in U(I,K)} w.$$  

Defeasible (instances of) axioms that are falsified by an interpretation are said be defeated. An interpretation $I$ of $K$ is said to be $n$-distant if $n = d(U(K,I))$. Using the notion of distance of interpretations, a partial ordering relation, $\prec$, can be defined over the set of all interpretations of a p-ALC knowledge base $K$. Given two interpretations $I_1$ and $I_2$ of $K$, $I_1 \prec I_2$ if and only if $d(U(K,I_1)) < d(U(K,I_2))$. Preferred interpretations are those interpretations with minimal distance value.

**Definition 3.3.** Let $I$ be an interpretation of a p-ALC knowledge base $K$. $I$ is said to be a preferred interpretation of $K$ if and only if (i) $I$ satisfies $A \cup T$ and (ii) there is no other interpretation $I'$ of $K$ such that $I' \prec I$. $K$ is said to be n-inconsistent if the preferred interpretations of $K$ are n-distant.

The entailment relation of our p-ALC is based on all preferred interpretations.

**Definition 3.4.** Let $K$ be a p-ALC knowledge base and let $Z$ be an axiom written in the language of $K$. $Z$ is a preferred consequence of $K$, written $K \models Z$, if and only if $Z$ is satisfied in every preferred interpretation $I$ of $K$.

Using the ALC finite model property (i.e., every satisfiable axiom admits a finite model) it is possible to show that for any given p-ALC knowledge base there exists an n-distant preferred interpretation, with $n \geq 0$.

**Proposition 1.** Let $K$ be a credible p-ALC knowledge base. There exists a non-negative integer $n$ and a preferred interpretation $I$ of $K$ such that $I$ is n-distant.

From Proposition 1 and Definition 3.3 it follows that any given credible knowledge base $K$ is n-inconsistent, for some non-negative integer $n$.

**Example 1.** Let $K_1 = \{\emptyset, \emptyset, \{R(b,a)\}, \{\forall R.C(b)\}\}$. Every interpretation of $K_1$ fails to satisfy at least one defeasible ABox axiom. If we order all its interpretations according to their distance value, the preferred interpretations are all 1-distant, i.e., those interpretations for which the sum of the weighted of unsatisfied defeasible axioms is equal to 1. $K_1$ is therefore 1-inconsistent.

When computing the distance of an interpretation, ABox axioms are treated atomically since each axiom is either satisfied or unsatisfied in an interpretation. Different is the case of defeasible TBox axioms. If TBox axioms were treated atomically, a single individual falsifying the axiom in a given interpretation would lead to all other instances of the TBox to be defeated even though the interpretation would not enforce it. In our notion of distance, and therefore preferred interpretations, we consider violations of instances of defeasible TBox axioms.

**Example 2.** Let $K_2 = \{\emptyset, \emptyset, \{C(a), \neg D(a), C(b)\}, \{C \subseteq D\}\}$. $K_2$ includes an inconsistency associated with $a$. The distance of a p-ALC interpretation takes into account each domain element for which a TBox axiom is defeated. For instance, the interpretation $I_1$ where $C^{a_1} = \{a^{a_1}, b^{a_2}\}$ and $D^{a_1} = \{a^{a_1}, b^{a_1}\}$ would be a 1-distant interpretation of $K_2$, whereas the interpretation $I_2$, where $C^{a_2} = \{a^{a_2}, b^{a_2}\}$ and $D^{a_2} = \{a^{a_2}\}$, would be a 2-distant interpretation. $I_1$ would therefore be a preferred interpretation for which $D(b)$ would be true. $K_2$ is 2-inconsistent. $K_2 \models C(b)$ and $K_2 \models D(b)$.
Example 3. Let $a$, $b$, and $c$ be individuals. Let’s consider the following ABox axioms: “$a$ belongs to a group of patients” ($P(a)$); “everyone that $b$ referred is healthy” ($\forall R.H(b)$), “$c$ is sick” ($S(c)$), “everybody that $c$ referred is sick” ($\forall R.S(c)$), and “$c$ referred at least one patient in the group” ($\exists R.P(c)$). The domain is represented by the credible knowledge base $K = \langle A_3, T_3, A_{d3}, T_{d3} \rangle$, where:

$A_3 : P(a)$ \hspace{1cm} (1) \hspace{1cm} $\forall R.H(b)$ \hspace{1cm} (2) \hspace{1cm} $S(c)$ \hspace{1cm} (3) \hspace{1cm} $\forall R.S(c)$ \hspace{1cm} (4) \hspace{1cm} $\exists R.P(c)$ \hspace{1cm} (5)

$T_3 : H \sqsubseteq \neg S$ \hspace{1cm} (6)

$A_{d3} : R(b,c)$ \hspace{1cm} (7)

$T_{d3} : P \sqsubseteq H$ \hspace{1cm} (8)

$K$ includes two sets of axioms leading to inconsistencies: \{(2), (3), (6), (7)\} and \{(4), (5), (6), (8)\}. Each preferred interpretation must satisfy the non defeasible axioms and hence must satisfy (1) – (6). But in the first set, any preferred interpretation that satisfies (2), (3) and (6) must also satisfy $\neg H(c)$ and therefore defeats $R(b,c)$ (i.e. (7)). But any such preferred interpretation must also satisfy (5). So there is some (named or unnamed) individual $x$ in the domain for which $R(c,x)$ and $P(x)$ are satisfied. By (4), $S(x)$ is also satisfied and since $(\neg H \cup \neg S)(x)$ is satisfied, so is $\neg H(x)$. Thus, $(\neg P \cup H)(x)$ is not satisfied (i.e. (8) is defeated). Hence, each preferred interpretation must be at least 2-distance. Hence, $K_3$ is 2-inconsistent, and since in every preferred interpretation (8) is satisfied for $a$, we have that $K_3 \models H(a)$. From the above argument we have $K_3 \models S(c)$ and $K_3 \models (\exists R.(P \cap S))(c)$. The latter follows because each preferred interpretation includes some individual reified as $x$ for which $P(x)$ and $S(x)$ are satisfied. We can infer that $a$ is healthy, because he/she belongs to the group, but there is at least one unhealthy individual within this group.

Considering (3) to be defeasible, i.e. $S(c)^{[1]}$, would allow for the possibility that $c$ might not be sick. Now there are 2-distance preferred interpretations in which $c$ is sick and others in which $c$ is healthy. $K_3 \not\models H(c)$ and $K_3 \not\models S(c)$. The conflict can be arbitrated by choosing a higher weight for axiom (3) or (7). For example, considering (7) to be $R(b,c)^{[2]}$ leads to $K_3 \models \neg S(c)$. In an interpretation satisfying $S(c)$, $\neg H(c)$ is also satisfied from (6), and (7) is defeated. But defeating (7) adds a weight of 2 and any such interpretation would be at least 3-distance and therefore not preferred. Having defined the semantics for our preferential $ALC$, we are now interested in computing the preferred consequences of a given $p-ALC$ knowledge base. Tableau algorithms underpin many modern description logic reasoners including [13] and [24]. Such algorithms incorporate blocking strategies that limit the size of the domains considered during reasoning, in the presence of cycles in the TBox in order to guarantee termination. Inspired by these existing techniques we have developed a modified tableau algorithm that accommodates our notion of preferential consequences (see Section 4). We have then encoded this algorithm (see Section 5) in Answer Set Programming (ASP) and used its optimisation features to target minimal (or maximal) satisfaction of constraints when searching for models (or solutions).

4 Tableau for Preferential $ALC$

In this section we present an $ALC$ satisfiability tableau algorithm [2] for the $p-ALC$ semantics. A satisfiability tableau algorithm is a proof procedure that when applied to a knowledge base generates some partial model of the knowledge base if this is consistent, or a closed tableau otherwise. Many common inference tasks for description logics are implemented using satisfiability tableau algorithms. These are proof by refutation algorithms used to check whether
Reasoning in the presence of inconsistency through Preferential $\mathcal{ALC}$

Deane, Broda and Russo

$\mathcal{K} \models C(x)$ by checking whether $\mathcal{K} \cup \neg C(x)$ is satisfiable (i.e. leads to a closed tableau). Before embarking on the presentation of our tableau method it is important to show that the concept of proof by refutation is also true in the context of our $p\mathcal{ALC}$ semantics.

**Theorem 1.** Let $\mathcal{K} = \langle A, T, A_d, T_d \rangle$ be an $n$-inconsistent $p\mathcal{ALC}$ knowledge base and let $C(x)$ be an ABox axiom in the language of $\mathcal{K}$. Then $\mathcal{K} \models C(x)$ if and only if either (i) $\langle A \cup \{\neg C(x)\}, T, A_d, T_d \rangle$ is inconsistent or (ii) $\langle A \cup \{\neg C(x)\}, T, A_d, T_d \rangle$ is $m$-inconsistent, for some $m > n$.

We now introduce a modified $\mathcal{ALC}$ tableau algorithm for the $p\mathcal{ALC}$ semantics. A satisfiability $\mathcal{ALC}$ tableau algorithm begins with the given ABox and applies a set of rules to develop all possible expansions (branches). The TBox are expanded by including suitable ABox axioms. Expansions continue until each possible expansion either a) leads to a contradiction, called a clash, or b) no further expansion is possible. In the latter case the branch constructed so far represents a (partial) model of $\mathcal{K}$. Expansions for existential quantifiers introduce names called parameters that do not appear in the knowledge base signature. These individuals serve as witnesses to the expansions. Each name introduced is required to be fresh to the ABox being expanded, meaning it does not appear within this ABox. To ensure termination in the presence of a cyclic TBox, a blocking strategy is used. Informally, the idea is to detect when expanding an axiom would provide no new information, i.e. there is some other individual within the ABox that is already required to satisfy the same set of concepts.

The technique is adapted for $p\mathcal{ALC}$ by considering all possible valid expansions using the non-defeasible axioms, omitting subsets of the defeasible ABox axioms and TBox axioms from defeasible TBox rule applications. The intuition is that an interpretation based on an open branch which minimises the omissions is maximally consistent and coincides with a preferred interpretation. Algorithm 1 constructs a $p\mathcal{ALC}$ tableau branch using the tableau rules defined in Table 1 and the expansions formalised in Definitions 4.1 - 4.2. Note that a branch will only be composed of (non-defeasible) ABox axioms and we will sometimes refer to a branch as a sequence of ABox axioms.

Algorithm 1 starts with the non-defeasible ABox axioms and a chosen subset of the defeasible ABox axioms. The remaining (those not chosen) defeasible ABox axioms are added to an omitted set $O_b$. As the ABox axioms are "expanded", all possible applications of $\rightarrow T$ rule to the non-defeasible TBox axioms in $T$ are performed. For the defeasible TBox axioms, the application of the $\rightarrow T_d$ rule may or may not lead to the addition of an ABox axiom to the branch. The set $S$ in line 9 of the algorithm keeps track of the defeasible TBox axiom instances to which $\rightarrow T_d$ rule is applied. The defeasible TBox axiom instances for which an ABox axiom was not added to the branch are at the end added to the omitted set $O_b$ (see line 12). However, new individuals may be introduced during the tableau expansion (i.e. see rule $\rightarrow \exists \forall$). To guarantee termination, a blocking strategy is used (see Definition 4.1). This assumes an ordering in which individuals are introduced, during the construction of a branch, for which some individual are defined to be older than others, so preventing mutual blocking ($x$ blocks $y$ and $y$ blocks $x$). Once an individual is blocked in a branch, it will never become unblocked. When no further expansions can be applied to a branch, the omitted set corresponds to the defeasible axioms instances that will be assumed to be false (i.e. defeated) in the interpretation constructed from that branch, and the sum of their weights defines the distance of the interpretation. Note that by changing the initial choice of defeasible ABox axioms and/or choosing different additions to a branch of defeasible TBox related axiom instance, Algorithm 1 can generate many possible

---

2Our notion of blocking is an example of static subset blocking in [18].
branches. As shown later, those open branches with minimal sum of weights of the defeated axiom instances in the omitted set correspond to preferred interpretations.

**Definition 4.1.** Let \( A_0 \subseteq A_1 \ldots \subseteq A_n \) denote the sequence of (not closed) ABoxes of a branch in a sequence of applications of tableau rules. An individual \( x \) is older than an individual \( y \) if \( x \) is introduced in \( A_i \) and \( y \) is introduced in \( A_j \) where \( 0 \leq i < j \leq n \). An individual \( y \) is blocked by individual \( x \) at step \( A_j \) if (i) \( x \) is older than \( y \) and (ii) \( \{ C \mid C(y) \in A_j \} \subseteq \{ C \mid C(x) \in A_j \} \). If \( y \) is blocked by \( x \) we say \( y \) is blocked.

**Algorithm 1:** A branch generating tableau algorithm

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
<th>1. Choose ( A_0 ) a subset of ( A_d );</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A_0 ) is open and not complete do</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( r ) an instance of a rule that applies to ( A_0 );</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7. ( A_b := ) the expansion of ( A_0 ) by ( r );</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8. if ( r ) is ( \rightarrow_r ) for unblocked ( x ) and ( C \subseteq D^{[w]} ) then</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( S := S \cup { (C \subseteq D^{[w]}, x) } );</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9. end</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10. ( O_b := O_{\text{init}} \cup { (C \subseteq D^{[w]}, x) \mid (C \subseteq D^{[w]}, x) \in S, (\neg C \cup D)(x) \notin A_0 } );</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11. if ( A_b ) is open then return ( (A_b, O_b) );</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12. else return ( \bot );</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rule</th>
<th>Valid expansion of ( A_b )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \neg \tau )</td>
<td>If ( (C \cap D)(x) \in A_b ), ( x ) is not blocked and ( (C(x) \notin A_b ) or ( D(x) \notin A_b ) )</td>
</tr>
<tr>
<td>then ( A_e = A_b \cup { C(x), D(x) } );</td>
<td></td>
</tr>
<tr>
<td>( \rightarrow )</td>
<td>If ( (C \cup D)(x) \in A_b ), ( x ) is not blocked and ( (C(x) \notin A_b ) and ( D(x) \notin A_b )</td>
</tr>
<tr>
<td>then ( A_e = A_b \cup { C(x) } ) or ( A_e = A_b \cup { D(x) } );</td>
<td></td>
</tr>
<tr>
<td>( \rightarrow_\forall )</td>
<td>If ( (\forall R.C)(x) \in A_b ), ( x ) is not blocked and ( R(x, y) \in A_b ) and ( C(y) \notin A_b )</td>
</tr>
<tr>
<td>then ( A_e = A_b \cup { C(y) } );</td>
<td></td>
</tr>
<tr>
<td>( \rightarrow_\exists )</td>
<td>If ( (C \subseteq D) \in T ) and ( \exists ) an unblocked individual ( x ) in ( A_b ) and ( (\neg C \cup D)(x) \notin A_b )</td>
</tr>
<tr>
<td>then ( A_e = A_b \cup { (\neg C \cup D)(x) } );</td>
<td></td>
</tr>
<tr>
<td>( \rightarrow_\tau_d )</td>
<td>If ( C \subseteq D^{[w]} \in T_d ) and there exists an unblocked individual ( x ) in ( A_b )</td>
</tr>
<tr>
<td>and ( (C \subseteq D^{[w]}, x) \notin S ) then ( A_e = A_b \cup { (\neg C \cup D)(x) } ) or ( A_e = A_b )</td>
<td></td>
</tr>
<tr>
<td>( \rightarrow_\exists \forall )</td>
<td>If ( (\exists R.C)(x) \in A_b ), ( x ) is not blocked and ( \neg \exists y [R(x, y) \in A_b ) and ( C(y) \in A_b )</td>
</tr>
<tr>
<td>and no other rule applies to ( A_b )</td>
<td></td>
</tr>
<tr>
<td>then ( A_e = A_b \cup { R(x, z), C(z) } \cup { D(z) \mid (\forall R.D)(x) \in A_b } ) where ( z ) is fresh</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: \( p\)-\( \text{ALC} \) blocking tableau rules

**Definition 4.2.** Let \( A_b \) be a current (open) branch generated starting from a given knowledge base \( (A, T, A_d, T_d) \). \( A_e \) is a valid expansion of \( A_b \) with respect to \( T \) and \( T_d \) if and only if \( A_e \) is
generated from \( \mathcal{A}_b \) applying an instance of a \( p\text{-}\mathbf{ALC} \) tableau rule in \( \mathcal{R} \) defined in Table 1, with blocking conditions in Def. 4.1. \( \mathcal{A}_b \) is said to be complete if there are no valid expansions of \( \mathcal{A}_b \) under the set of rules \( \mathcal{R} \); \( \mathcal{A}_b \) it is said to be closed if \( \mathcal{A}_b \) includes \( C(x) \) and \( \neg C(x) \) for some concept \( C \) and open otherwise.

Given a \( p\text{-}\mathbf{ALC} \) knowledge base \( \mathcal{K} = \langle \mathcal{A}, \mathcal{T}, \mathcal{A}_d, \mathcal{T}_d \rangle \), the application of Algorithm 1 to \( \mathcal{K} \) generates possible closed or open branches. Each branch has an associated omitted set \( \mathcal{O}_b \) and we call the distance of a branch the sum of the weights of the defeasible axiom instances in the omitted set. An open branch is called an \( m\text{-minimal} \) branch, if there is no other open branch whose distance is strictly smaller than \( m \).

**Definition 4.3.** Let \( \mathcal{K} = \langle \mathcal{A}, \mathcal{T}, \mathcal{A}_d, \mathcal{T}_d \rangle \) be a credible knowledge base and \( \mathcal{B} \) be the set of \( m\text{-minimal} \) branches generated for \( \mathcal{K} \) by Algorithm 1. Let \( b = \langle \mathcal{A}_b, \mathcal{O}_b \rangle \in \mathcal{B} \), \( P(b) \) denote the set of unblocked parameters introduced in the branch and \( \mathcal{A}_b' \) be the set of axioms in \( \mathcal{A}_b \) not involving blocked individuals. Let \( \mathcal{D}_b \) be the Herbrand domain based on the signature of \( \mathcal{K} \) in which \( \mathcal{N}_I \) is augmented by the unblocked parameters \( P(b) \). The Herbrand interpretation \( \mathcal{I}_b \) of \( \mathcal{K} \) based on \( \mathcal{D}_b \) is defined as follows. For every \( x, y \in \mathcal{D}_b \), \( A \in \mathcal{N}_C \) and \( R \in \mathcal{N}_R \): \( A(x) \) is true iff \( A(x) \in \mathcal{A}_b; R(x, y) \) is true if either \( R(x, y) \in \mathcal{A}_b' \) or \( R(x, z) \in \mathcal{A}_b \) and \( z \) is blocked by \( y \) and is false otherwise.

**Proposition 2.** Let \( \mathcal{K} = \langle \mathcal{A}, \mathcal{T}, \mathcal{A}_d, \mathcal{T}_d \rangle \) be a credible \( n\text{-inconsistent} \) knowledge base. Let \( \mathcal{B} \) be the set of \( m\text{-minimal} \) branches generated by Algorithm 1 applied to \( \mathcal{K} \). Then \( m \geq n \) and for each branch \( b \in \mathcal{B} \) the interpretation \( \mathcal{I}_b \) constructed from \( b \) is an \( m\text{-distant} \) interpretation of \( \mathcal{K} \).

**Proposition 3.** Let \( \mathcal{K} = \langle \mathcal{A}, \mathcal{T}, \mathcal{A}_d, \mathcal{T}_d \rangle \) be a credible \( n\text{-inconsistent} \) knowledge base. Algorithm 1, applied to \( \mathcal{K} \), generates at least one \( n\text{-minimal} \) open branch.

Given the above results, to check whether \( \mathcal{K} \models C(x) \) the idea would be first enumerate all the branches generated by applying Algorithm 1 to \( \mathcal{K} \) to identify the distance \( m \) of the minimal branches, this can be done by using a branch-and-bound search (or similar). Then Algorithm 1 can be applied to \( \mathcal{K}' = \langle \mathcal{A} \cup \neg C(x), \mathcal{T}, \mathcal{A}_d, \mathcal{T}_d \rangle \) to search for exactly an \( m\text{-distant} \) branch. If such a branch is found then we now conclude \( \mathcal{K} \not\models C(x) \). Otherwise \( \mathcal{K}' \) was inconsistent or \( \mathcal{K}' \) is \( n\text{-inconsistent} \) for some \( n > m \). In either case, we can conclude that \( \mathcal{K} \not\models C(x) \). In the next section we show how the computation of such branches can be done in Answer Set Programming (ASP), exploiting the optimisation features of (ASP).

## 5 Implementation in ASP

In this section we present the ASP implementation of our tableau method for computing the entailment relation of our \( p\text{-}\mathbf{ALC} \). It is implemented using Clingo (version 4.5.0)[11]. To make the paper self-contained we briefly summarise the features of the ASP fragment that we use in our implementation. For a full specification of the **ASP-Core-2** syntax and semantics, the reader is referred to [7].

Terms are constants (integers or strings starting with a lower case letter), variables, represented as strings starting with an upper case letter, or functional terms of the form \( f(t_1, ..., t_n) \) where \( f \) is a functor, \( t_i \), for \( 0 \leq i \leq n \), are terms. Atoms are of the form \( p(t_1, ..., t_n) \) where \( p \) is a predicate name, \( n > 0 \), and \( t_1, ..., t_n \) are terms. Built-in comparison atoms \( =, \neq, <, >, \leq, \geq \) follow infix notation. A literal is an atom \( b \) or a negated atom not \( b \) where not denotes negation by failure. A rule \( r \) takes the form \( h_1 | ... | h_m \leftarrow b_1, ..., b_n \) where \( m \geq 0, n \geq 0, h_i \) are atoms, the symbol | denotes disjunction, \( b_1, ..., b_n \), are literals where ,
denotes conjunction. \( \{ h_1, ..., h_m \} \ (\{ b_1, ..., b_n \}) \) is the head (resp. body) of \( r \). A fact is a rule with an empty body, an integrity constraint is a rule with an empty head. A weak constraint takes the form \( \sim b_1, ..., b_n[w, t_1, ..., t_m] \) where \( n \geq 1 \), \( b_1, ..., b_n \) are literals, \( w \) is an integer, \( m \geq 1 \) and \( t_1, ..., t_m \) are terms. A program \( P \) is a finite set of rules and weak constraints.

\( HUP \) (and \( HBP \)) denotes the Herbrand Universe (and Herbrand Base) of \( P \). A ground instance of \( P \), \( gnd(P) \) is obtained by substituting each variable appearing in a rule or weak constraint with an element from \( HUP \) and evaluating built-in predicates. Given a program \( P, I \subseteq HBP \) is a (Herbrand) interpretation of \( P \); a rule \( r \in gnd(P) \) is satisfied by \( I \) if some \( h \in \{ b_1, ..., b_n \} \) is true w.r.t \( I \) when \( b_1, ..., b_n \) are true w.r.t \( I \) if \( P \) is a model of \( P \) if every rule in \( gnd(P) \) is satisfied by \( I \). The reduct of \( P \) w.r.t \( I \), denoted \( P^I \), is the set of rules from \( gnd(P) \) for which \( b_1, ..., b_n \) are true w.r.t \( I \); \( I \) is an answer set of \( P \) if \( I \) is a \( \subseteq \)-minimal model of the reduct \( P^I \). Weak constraints identify optimal answer sets. The optimality of an answer set \( S \) of a program \( P \) is the weighted sum of \( w \) for each unsatisfied ground instance of weak constraints having a unique set of terms \( t_1, ..., t_m \). \( S \) is optimal for \( P \) if no other answer set of \( P \) has a smaller optimality.

A given knowledge base \( K \) is represented in ASP as a set of facts denoted \( K^\tau \). Each name \( N \) in the signature is translated into an ASP constant \( N^\tau \) by mapping the first letter to its lower case. Concepts are translated to ground ASP terms using the unary or binary function symbols \( \text{neg}, \land, \lor, \circ \text{Some} \) and \( \circ \text{All} \) to denote the constructors \( \neg, \land, \lor, \exists, \forall \) (resp.). Table 2 defines the inductive mappings used to translate concepts and axioms to ASP terms.

<table>
<thead>
<tr>
<th>C</th>
<th>Term C^\tau</th>
<th>C</th>
<th>Term C^\tau</th>
<th>Z</th>
<th>Term Z^\tau</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>thing</td>
<td>C_1 \sqcap C_2 \sqcap ... \sqcap C_n \land \circ \text{all}(C_1^\tau,(C_2 \sqcup \ldots \sqcup C_n)^\tau)</td>
<td>C(x)</td>
<td>ca(C^\tau, x^\tau)</td>
<td></td>
</tr>
<tr>
<td>⊥</td>
<td>neg(thing)</td>
<td>C_1 \sqcup C_2 \sqcup ... \sqcup C_n \lor \circ \text{some}(R^\tau, C^\tau)</td>
<td>R(x,y)</td>
<td>ra(R^\tau^\tau, y^\tau)</td>
<td></td>
</tr>
<tr>
<td>¬C</td>
<td>neg(C^\tau)</td>
<td>\exists R.C</td>
<td>C \sqsubseteq D</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(C)</td>
<td>(C^\tau)</td>
<td>\forall R.C</td>
<td>sc(C^\tau, D^\tau)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Mapping concepts \( C \) and axioms \( Z \) to ASP terms

**Definition 5.1.** Let \( K = \langle A, T, A_d, T_d \rangle \) be a knowledge base with signature \( \langle N_C, N_R, N_C \rangle \). The encoding in ASP of \( K \) and \( \text{sig}(K) \) is defined as follows:

\[
\text{enc}(K)^\tau = \{ ax(Z^\tau, w), |Z| \in A \cup T \} \cup \{ ax(Z, w), |Z|^w \in A_d \cup T_d \}
\]

\[
\text{sig}(K)^\tau = \{ i(x^\tau), |x| \in N_I \} \cup \{ r(R^\tau), |R| \in N_R \} \cup \{ c(C^\tau), |C| \in N_C \}
\]

where the translations by \( \tau \) are given in Table 2.

Considering Example 3, \( K_3 \) is encoded by the ASP facts:

\[
\begin{align*}
ax(ca(p,a), 0). & \quad ax(ca(o\text{all}(r, h), b), 0). & \quad ax(ca(o\text{all}(s, c), 0). \\
ax(ca(o\text{all}(r, s), c), 0). & \quad ax(ca(o\text{some}(r, p), c), 0). & \quad ax(sc(h, \text{neg}(s)), 0). \\
ax(ra(r, b, c), 1). & \quad ax(sc(p, h), 1). & \quad ax(sc(h, \text{neg}(s)), 0). \\
i(a). & \quad i(b). & \quad r(r). c(h). c(p). c(s).
\end{align*}
\]

The full ASP encoding of a defeasible knowledge base is given by \( \text{ASP}(K) = K^\tau \cup \text{sig}(K)^\tau \cup \text{P}_{\text{base}} \cup \text{P}_{\text{cum}} \) where \( \text{P}_{\text{base}} \) and \( \text{P}_{\text{cum}} \) are defined below.

The program \( \text{P}_{\text{base}} \) uses predicates \( \text{isa}/2 \) and \( \text{hasa}/3 \) to represent ABox axioms within a tableau expansion. The supported inference of ground atoms representing non-defeasible ABox axioms is unconditionally, whereas that of atom representing defeasible ABox axiom instances is subject to choice (see rule (a3)):

\[
\text{isa}(X, C) \leftarrow ax(ca(C, X), 0)
\]
Reasoning in the presence of inconsistency through Preferential ALC

For example, given a non-defeasible ABox \((C \cap D)\) in \(\mathcal{K}\), answer sets of \(\text{ASP}(\mathcal{K})\) will include 
\(ax(ca(\text{and}(c,d),a),0)\). The atom \(isa(\text{and}(c,d),a)\) will be included in every answer set by (a1).

If the non-defeasible ABox \(R(a,b)\) is in \(\mathcal{K}\), the ASP program \(\mathcal{K}^+\) will include \(ax(ra(a,b),2)\).

By rule (a5), either \(hasa(a,r;b)\) or \(u(ra(r,a,b),a,2)\) will be in each answer set. The weak constraint (a6) increases the total weight of the answer set by 2 (meaning the answer set is less optimal) when \(u(ra(r,a,b),a,2)\) is added to it.

\[
\begin{align*}
\text{hasa}(X,R,Y) & \leftarrow ax(ra(R,X,Y),0) \quad (\text{a2}) \\
\text{isa}(X,C) \mid u(ca(C,X),X,W) & \leftarrow ax(ca(C,X),W), W > 0 \quad (\text{a3}) \\
\neg \text{isa}(ca(C,X),X,W) & \leftarrow [W,ca,C,X] \quad (\text{a4}) \\
\text{hasa}(X,R,Y) \mid u(ra(R,X,Y),X,W) & \leftarrow ax(ra(R,X,Y),W), W > 0 \quad (\text{a5}) \\
\neg \text{isa}(ra(R,X,Y),X,W) & \leftarrow [W,ra,R,X,Y] \quad (\text{a6})
\end{align*}
\]

Rules (e1-e7) capture the expansion rules. The symbol \(\neg\text{neg}\) is a custom function implemented in the Lua language\(^3\) and ensures concepts are expressed in NNF; if \(X\) is an ASP term representing concept \(C\), \(\neg\text{neg}(X) = (\neg C)^+\).

\[
\begin{align*}
\text{isa}(X,C) & \leftarrow \text{isa}(X,\text{and}(C,D)) \quad (\text{e1}) \\
\text{isa}(X,D) & \leftarrow \text{isa}(X,\text{and}(C,D)) \quad (\text{e2}) \\
\text{isa}(X,C) \mid \text{isa}(X,D) & \leftarrow \text{isa}(X,\text{or}(C,D)) \quad (\text{e3}) \\
\text{isa}(X,C) & \leftarrow \text{isa}(X,\text{or}(\neg\text{neg}(C),D)), \text{hasa}(X,R,Y) \quad (\text{e4}) \\
\text{isa}(X,\text{or}(\neg\text{neg}(C),D)) \mid u(\text{sc}(C,D),X,W) & \leftarrow ax(\text{sc}(C,D),W), i(X), W > 0 \quad (\text{e5}) \\
\neg \text{isa}(\text{sc}(C,D),X,W) & \leftarrow [W,\text{sc},C,D,X] \quad (\text{e6})
\end{align*}
\]

\(\text{isa}(X,\text{thing}) \leftarrow i(X)\) \quad (\text{e8})
\[\neg \text{isa}(X,\text{neg}(\text{thing}))\] \quad (\text{e9})
\[\text{isa}(X,C), \text{isa}(X,\text{neg}(C)), c(C)\] \quad (\text{e10})
\[\text{hw}(X,\text{oSome}(R,C)) \leftarrow \text{isa}(X,\text{oSome}(R,C)), \text{hasa}(X,R,Y), \text{isa}(Y,C)\] \quad (\text{e11})
\[\text{need}(X,\text{oSome}(R,C)) \leftarrow \text{isa}(X,\text{oSome}(R,C)), \neg \text{hw}(X,\text{oSome}(R,C))\] \quad (\text{e12})
\[\text{used}(X) \leftarrow i(X)\] \quad (\text{e13})

Rule (e8) captures the property that every named individual has to belong to the “top” concept, rules (e9, e10) guarantee that answer sets include only consistent expansions. Rules (e11-e13) capture the \(\rightarrow_{\exists y}\) tableau rule, where the atom \(\text{hw}(X,\text{oSome}(R,C))\) means that “\(X\) has a witness to the concept \(\exists R.C\)”. Where no such witness exists, \(\text{need}/2\) labels that a parameter must be introduced. However, since we do not know a priori how many parameters are required, new parameters and their associated rules are introduced on an “as-needed basis”. Grounding and solving is implemented iteratively under script control.

Initially, the program \(\mathcal{K}^+ \cup \text{sign}(\mathcal{K})^+ \cup \text{base}\) is grounded, denoted \(P^0_g\), and solved returning some answer set \(S^0\) or \(\text{unsatisfiable}\). In the former case a sequence of one or more iterations is carried out, each generating an extension to the initial grounding \(P^0_g\) and then re-solving the extended ground program. Each \(\text{need}(X,\text{oSome}(R,C))\) atom instance in \(S^0\) indicates that a parameter is needed to serve as a witness to the individual \(X\) for the concept represented as \(\text{oSome}(R,C)\). The program \(P^0_g\) is cumulatively extended with the program \(\text{gnd}(P_{cum})\) giving

---

\(^3\)Lua functions are distinguished by a leading @ symbol.
As (see below) and solved again, either returning an answer set $S$ or unsatisfiable. Subsequent iterations are similarly carried out and terminate either when no further parameters are needed or the solver returns unsatisfiable. We call the final generated answer sets the optimal solutions of program $ASP(K)$.

The program $P_{cum}$ implements the rules required to introduce and expand concepts for a parameter. It begins with the #program directive which instructs the grounder to postpone grounding the subsequent rules until requested. Each need($X, oSome(R,C)$) atom instance is associated with a unique fresh parameter identifier(PID), and used to assign the three arguments in $P_{cum}$: $\neg p$ (a PID), $\neg i$ (an individual X), and $\neg c$ (a concept $oSome(R,C)$), each triple of arguments leading to a set of ground instances of $P_{cum}$.

\[
\begin{align*}
\text{#program } & \text{cum}(i,p,j,c) \\
\text{hasa}(i,R,\neg p) & \leftarrow \text{isa}(i,\neg c), \text{oSome}(R,C) = \neg c \\
\text{isa}(i,p,C) & \leftarrow \text{isa}(i,\neg c), \text{oSome}(R,C) = \neg c \\
\text{isa}(i,p,C) & \leftarrow \text{hasa}(i,R,\neg p), \text{isa}(i,oAll(R,C)) \\
\text{ceax}(i,p,C,\neg i) & \leftarrow \text{oSome}(R,C) = \neg c, \text{isa}(i,\neg c) \\
\text{ceax}(i,p,C,\neg i) & \leftarrow \text{isa}(i,oAll(R,C)), \text{hasa}(i,R,\neg p) \\
\text{used}(i) & \leftarrow \text{ceax}(i,p,C,\neg i) \\
\text{isa}(i,p,\text{thing}) & \leftarrow \text{used}(i) \\
\text{dnb}(Y,\neg p) & \leftarrow \text{used}(Y), Y = \neg p, \text{ceax}(i,p,C,\neg i), \neg \text{isa}(Y,C) \\
\text{b}(\neg p) & \leftarrow \text{used}(\neg p), \text{used}(Y), Y = \neg p, \neg \text{dnb}(Y,\neg p)
\end{align*}
\]

Rules (c2-c6) capture the $\rightarrow_{3\forall}$ rule with respect to fresh parameters. Concepts from the $\rightarrow_{3\forall}$ rule are recorded (c5,c6). Parameters serving as a witness are labelled as used (c7) and added to the top concept (c8). Rules (c9-c10) keep track of the blocking mechanism. Atom $\text{dnb}(Y,\neg p)$ states that “$Y$ does not block $\neg p$”, and atom $\text{b}(\neg p)$ that “$\neg p$ is blocked”. Since the grounding calls to $P_{cum}$ are sequential, each used $Y$ was introduced within an earlier grounding step and represents an older individual within an expansion. Mutual blocking is prevented by enforcing $Y = \neg p$. Rules (c11-c20) expand used, unblocked parameters:

\[
\begin{align*}
\text{isa}(\neg p,C) & \leftarrow \text{isa}(\neg p,\text{and}(C,D)), \text{used}(\neg p), \neg \text{b}(\neg p) \\
\text{isa}(\neg p,D) & \leftarrow \text{isa}(\neg p,\text{and}(C,D)), \text{used}(\neg p), \neg \text{b}(\neg p) \\
\text{isa}(\neg p,C) | \text{isa}(\neg p,D) & \leftarrow \text{isa}(\neg p,\text{or}(C,D)), \text{used}(\neg p), \neg \text{b}(\neg p) \\
\text{isa}(\neg p,\text{or}((\neg \text{neg}(C),D))) & \leftarrow \text{ax}(\text{sc}(C,D),0), \text{used}(\neg p), \neg \text{b}(\neg p) \\
\text{isa}(\neg p,\text{or}((\neg \text{neg}(C),D))) | u(\text{sc}(C,D),\neg p,W) & \leftarrow \\
\text{ax}(\text{sc}(C,D),W), W > 0, \text{used}(\neg p), \neg \text{b}(\neg p) \\
\text{:= u(\text{sc}(C,D),\neg p,W), used(\neg p), \neg b(\neg p) } & [W, sc, C, D, \neg p] \\
\text{need}(\neg p, oSome(R,C)) & \leftarrow \text{isa}(\neg p, oSome(R,C)), \text{used}(\neg p), \neg \text{b}(\neg p) \\
\text{=: isa}(\neg p, \text{neg(thing)}) \\
\text{=: isa}(\neg p,C), \text{isa}(\neg p, \text{neg}(C)), c(C)
\end{align*}
\]

Recall from Section 4 that the query $K \models C(x)$? can be answered by refutation in two steps. Our implementation of Algorithm 1 in ASP also follows the same approach. We first find an optimal answer set with optimality o of $ASP(K)$, called $S_o$, which will indicate that $K$ is o-inconsistent. Then we consider $ASP(K')$, where $K'$ is given by $K$ augmented with $\neg C(x)$. In
this second step we look for an $o$-optimal answer set. If such an answer set is not found then $K'$ is either unsatisfiable or $m$-optimal for $m > o$, and in both cases we can conclude $K \models C(x)$. This is guaranteed by Theorem 1 and the following properties of $ASP(K)$: $ASP(K)$ belongs to the class of Finitely Ground Programs [6], its rules (excluding constraints) are locally stratified, and interpretations constructed from $n$-minimal branches obtained from Algorithm 1 correspond to $n$-optimal solutions of $ASP(K)$.

### 6 Evaluation

We have applied our approach to the amino-acid knowledge base\(^4\), used as a benchmark in [21]. This knowledge base includes 46 named concepts, 5 roles and 1 individual, as signature; and, as axioms, 1 concept assertion, 0 role assertions, 238 concept inclusions, 199 concept equivalences and 12 disjoint concepts. We have generated from it amino-acid0, a (uniform) 0-inconsistent defeasible $p$-$\text{ALC}$ knowledge base with 20 individuals\(^5\) in which every axiom is defeasibly asserted with a weight of 1. In addition, two inconsistent uniform knowledge bases were generated by adding further defeasible concept assertions: amino-acid1 is 1-inconsistent and amino-acid5 is 5-inconsistent. Three tasks were evaluated for each of these three knowledge bases $K$: T1) Establish the $n$-inconsistency of $K$ by finding one optimal answer set of $ASP(K)$; T2) Prove or disprove entailment for a randomly selected concept assertion $C(x)$ where $C \in N_C$ and $x \in N_I$ by searching for an $n$-optimal answer set of $ASP(K \cup \{\neg C(x)\})$, finding such a model proves entailment (PE) and failing to find one disproves entailment (DE); T3) Concept retrieval, i.e. find the concepts to which each individual belongs by computing intersection of all $n$-optimal answer sets of $ASP(K)$.

Table 3 summarises, for each of these experiments, the computational time ($t$ as minutes:seconds), the total number of models found ($m$) and the number of grounding iterations of $P_{cum}(i)$. Values are averages of 10 runs using a platform based on an Intel(R) Core(TM) i7-2600 CPU @3.40GHz with 16G of RAM. $t$-out denotes the test was aborted after 60 minutes. The (non) entailments for amino-acid0 were verified using HermiT\(^{13}\) and for the others was checked manually. The results show that for each knowledge base proving entailment(PE) is faster than disproving entailment(DE), as is typical for tableau refutation. Increasing the numbers of inconsistencies leads to increased computation time.

### 7 Related Work

$p$-$\text{ALC}$ is most closely related to the ICAR repair semantics [19] defined for $DL$-$\text{Lite}_A$. However, the $p$-$\text{ALC}$ semantics are less cautious than the ICAR semantics. To illustrate this, consider a knowledge base $K = (A, T)$ in $DL$-$\text{Lite}_A$ that has an equivalent formulation in $\text{ALC}$. $K$ can

\[^4\]Available in the TONES repository http://www.tonesproject.org/.

\[^5\]1 asserted individual for each amino acid.
Reasoning in the presence of inconsistency through Preferential ALC

Deane, Broda and Russo

be expressed as a uniform p-ALC knowledge \( K' = \langle \emptyset, T, A, \emptyset \rangle \), where each defeasible axiom in \( A \) is assigned a weight of 1. For example, consider \( K_4 = \langle \{ C(a), R(a, b), C(b) \} \cup \{ C \sqsubseteq \neg \exists R \} \rangle \). \( K_4 \models ICAR C(b) \) and \( K'_4 \models C(b) \). For \( K_5 = \langle \{ R(a, b), R(a, c), D(a) \} \cup \{ \exists R \sqsubseteq \neg D \} \rangle \) there are two possible CAR-repairs, the sets \( \{ R(a, b), R(a, c), \neg D(b) \} \) and \( \{ D(a) \} \) which have an empty intersection. In contrast, \( K'_5 \models R(a, b), R(a, c), \neg D(a) \) because an interpretation that corresponds to the first CAR-repair is 1-distant and for the second is 2-distant. Error tolerant TBox reasoning in \( \mathcal{EL}^* \) using repair semantics was investigated in [20], but focused on a different task. They targeted the removal of unwanted consequences by removing subsets of TBox axioms (treated atomically). The work in [16] and [23] is based on applying Possibilistic logic to ALC in order to deal with uncertainty. TBoxes are considered atomically and the weights are compared over an absolute scale rather than cumulatively. The semantics can provide a measure of certainty when computing entailment, a feature that is not provided in p-ALC, but doesn’t take into account derived consequences. Another approach to inconsistent tolerant reasoning is using 4-valued paraconsistent semantics ([21] for a review). An advantage of the approach is that implementation is possible using existing reasoners. However, inferences that rely on disjunctive syllogism are not possible leading to more cautious semantics than p-ALC.

8 Conclusions and Future work

In this paper we have introduced p-ALC, a preferential ALC, proposed a modified tableau algorithm, together with its ASP implementation, and demonstrated the feasibility of approach using (modified) existing knowledge bases. Future directions include, conduct a more substantial evaluation aimed at assessing the impact on computational time of different ratios between defeasible and non-defeasible axioms, and of considering different weights. We will also assess the impact of incorporating tableau optimisations such as lazy unfolding and early clash detection mechanisms [1]. Confidence weights were used to arbitrate inconsistency and determine a measure of inconsistency for a given knowledge base. We will investigate the relationship between this and other inconsistency measures (e.g. Grant et al. [14]) and identify if such measures can be used to inform the assignment of weights to axioms.

References