A Dialogue System for Coherent Reasoning with Inconsistent Knowledge Bases

Silvio do Lago Pereira, Luiz Felipe Zarco dos Santos, Lucio Nunes de Lira

Department of Information Technology, FATEC-SP/CEETEPS, São Paulo, Brazil
Email: slago@fatecsp.br, luiz.santos58@fatec.sp.gov.br, lucio.lira@fatec.sp.gov.br

Received 3 July 2015; accepted 9 August 2015; published 12 August 2015

Copyright © 2015 by authors and Scientific Research Publishing Inc.
This work is licensed under the Creative Commons Attribution International License (CC BY).
http://creativecommons.org/licenses/by/4.0/

Abstract

Traditionally, the AI community assumes that a knowledge base must be consistent. Despite that, there are many applications where, due to the existence of rules with exceptions, inconsistent knowledge must be considered. One way of restoring consistency is to withdraw conflicting rules; however, this will destroy part of the knowledge. Indeed, a better alternative would be to give precedence to exceptions. This paper proposes a dialogue system for coherent reasoning with inconsistent knowledge, which resolves conflicts by using precedence relations of three kinds: explicit precedence relation, which is synthesized from precedence rules; implicit precedence relation, which is synthesized from defeasible rules; mixed precedence relation, which is synthesized by combining explicit and implicit precedence relations.

Keywords

Defeasible Reasoning, Inconsistent Knowledge, Precedence Relation, Dialogue System

1. Introduction

A knowledge base is a set of rules representing the knowledge of an expert in a specific domain. Traditionally, the Artificial Intelligence (AI) community assumes that a knowledge base must be free of inconsistency; otherwise, it turns out to be useless for an automated reasoning system. This assumption is motivated by the ex falso quodlibet principle [1], which establishes that “from a falsehood, anything follows”. According to this principle, an inconsistent knowledge base should force an automated reasoning system to collapse.

Despite that, there are many practical applications of automated reasoning where, due to the existence of rules with exceptions, inconsistent knowledge must be used (e.g., law, politics, and medicine) [2]. For example, let \( \Delta \) be a knowledge base with the following pieces of knowledge: “penguins do not fly”, “birds fly”, and “Tweety is a bird”. Then, since there is no counter evidence, it is coherent to infer “Tweety flies” from \( \Delta \). Now, suppose
that the new piece of knowledge “Tweety is a penguin” is inserted into \( \Delta \), resulting in a new knowledge base \( \Delta' \). Then, both “Tweety flies” and “Tweety does not fly” can be inferred from \( \Delta' \), and that is not a coherent reasoning. One way of restoring the consistency of \( \Delta' \) is to withdraw one of its conflicting pieces of knowledge [3], but this will destroy part of the knowledge. A better alternative would be to give precedence to the exception “penguins do not fly”. In this case, only “Tweety does not fly” can be coherently inferred from \( \Delta' \). Indeed, by using precedence relations, coherent reasoning in presence of inconsistency turns out to be possible.

In the last decades, reasoning with inconsistent knowledge has attracted great interest in the AI community. Nowadays, argumentation [4] is a common approach for coherent reasoning in presence of inconsistency, and several different formal models of argumentation have been proposed in the literature (e.g., [5]-[8]).

This paper proposes a system for coherent reasoning, based on dialogical argumentation and defeasible reasoning, which resolves conflicts by using precedence relations of three kinds: explicit precedence relation, which is synthesized from precedence rules; implicit precedence relation, which is synthesized from defeasible rules; mixed precedence relation, which is synthesized by combining explicit and implicit precedence relations.

The paper is organized as follows: Section 2 introduces the fundamentals of defeasible reasoning and explains how the three kinds of precedence relations are synthesized in our system; Section 3 describes the dialectical proof procedure on which our system is based; Section 4 presents some features of the dialogue system prototype implemented in Prolog; finally, Section 5 presents the conclusion of the paper.

2. Background

In this section, we start by defining the language used to specify knowledge bases in our dialogue system; then, we present the principles of defeasible reasoning with inconsistent knowledge; and, finally, we discuss how to synthesize three different kinds of precedence relations from the information declared in a knowledge base.

2.1. Knowledge Representation

An atom denotes an atomic proposition. A literal \( \lambda \) is an atom \( \alpha \) or a negated atom \( \neg \alpha \). Two literals \( \lambda \) and \( \lambda' \) are complementary literals if \( \lambda = \alpha \) and \( \lambda' = \neg \alpha \), or \( \lambda = \neg \alpha \) and \( \lambda' = \alpha \). The literal \( \top \) denotes a true proposition and it has no complementary literal. A conjunction is an expression \( \lambda_1 \land \cdots \land \lambda_k \), where each \( \lambda_i \) (\( 1 \leq i \leq k \)) is a literal. Technically, a conjunction \( \varphi = \lambda_1 \land \cdots \land \lambda_k \) is only a syntactic sugar notation for the set \( \Lambda(\varphi) = \{ \lambda_1, \cdots, \lambda_k \} \). Particularly, the trivial conjunction \( \varphi = \top \) denotes the set \( \Lambda(\varphi) = \emptyset \).

A defeasible rule is an expression \( \varphi \rightarrow \lambda \), where \( \varphi \) is a conjunction, called antecedent, and \( \lambda \neq \top \) is a literal, called consequent. Intuitively, a defeasible rule states that the literals in \( \Lambda(\varphi) \) are reasons to believe in \( \lambda \), if there is no evidence contrary to \( \lambda \). A defeasible rule is consistent if the set \( \Lambda(\varphi) \cup \{ \lambda \} \) has no complementary literals. A defeasible rule \( \top \rightarrow \lambda \) is a presumption. A labeled defeasible rule is an expression \( \ell : \pi \), where \( \pi \) is a defeasible rule and \( \ell \) is a unique label identifying \( \pi \). Two labeled defeasible rules \( \ell : \pi \) and \( \ell' : \pi' \) are called conflicting defeasible rules, denoted by \( \ell \delta \ell' \), if they have complementary consequents. Evidence against the consequent of a defeasible rule can emerge from its conflicts with other defeasible rules.

A precedence rule is an expression \( \ell \prec \ell' \), where \( \ell \) and \( \ell' \) are labels of conflicting defeasible rules, stating that the rule \( \ell \) precedes the rule \( \ell' \) (i.e., that the priority of rule \( \ell \) is higher than the priority of the rule \( \ell' \)). Since precedence rules do not involve atoms of the logical language, they are considered as meta-knowledge, whose only purpose is to provide information necessary to resolve conflicts between defeasible rules.

A knowledgebase \( \Delta \) is a finite set of consistent labeled defeasible rules and precedence rules. For example,

\[
\Delta' = \{ 1: p \rightarrow b, 2: b \rightarrow f, 3: p \rightarrow f, 4: \top \rightarrow p, 3 \prec 2 \}
\]

is a knowledgebase, where \( p \), \( b \), and \( f \) stand, respectively, for “penguin”, “bird”, and “fly”. In this knowledge base, the defeasible rule \( 2: b \rightarrow f \) states that “birds fly”, the defeasible rule \( 3: p \rightarrow f \) states that “penguins do not fly”, and the precedence rule \( 3 \prec 2 \) states that the defeasible rule \( 3 \) has precedence over the defeasible rule \( 2 \).

2.2. Defeasible Reasoning

As already said, a defeasible rule \( \varphi \rightarrow \lambda \) states that the literals in \( \Lambda(\varphi) \) are reasons to believe in the literal
\( \lambda \), if there is no counter evidence to \( \lambda \). In this context, the symbols \( \rightarrow \), \( \wedge \) and \( \rightarrow \) are not interpreted as in classical logic, since neither *modus ponens* (i.e., \( \{ \varphi, \varphi \rightarrow \lambda \} \models \lambda \)), nor *modus tollens* (i.e., \( \{ \varphi \rightarrow \lambda, \neg \lambda \} \models \neg \varphi \)) holds for defeasible rules. In fact, even when the antecedent of a defeasible rule is true, its consequent may be false.

Defeasible reasoning is based on an inference rule called *modus non excipiens* [9]. This inference rule differs from *modus ponens* because it has an implicit premise stating that the consequent of a defeasible rule follows from its antecedent, provided that there is no exception to the rule. Therefore, defeasible reasoning is a kind of reasoning that produces only a *contingent* demonstration of a literal \( \lambda \). Anyway, a necessary (although not sufficient) condition to believe in a literal \( \lambda \) is that it can be, at least, *defeasibly derived* from the knowledge base.

A defeasible derivation tree of a literal \( \lambda \) from a knowledge base \( \Delta \), denoted by \( Y_\lambda(\lambda) \), is a tree such that:

- The root of \( Y_\lambda(\lambda) \) is labeled with the literal \( \lambda \).
- For each node of \( Y_\lambda(\lambda) \) labeled with a literal \( \lambda' \), there exists a defeasible rule \( \varphi \rightarrow \lambda' \in \Delta \).
- If \( \varphi = \top \), then the node labeled with \( \lambda' \) is a leaf in \( Y_\lambda(\lambda) \); otherwise, if \( \varphi = \lambda_1 \wedge \cdots \wedge \lambda_k \), then that node has exactly \( k \) children nodes, which are labeled with \( \lambda_1, \ldots, \lambda_k \), respectively.

A defeasible derivation tree is generated by a backward search procedure, similar to SLD-refutation [10]. For example, a defeasible derivation tree of the literal \( u \) from \( \Delta^2 \) is depicted in Figure 1.

A literal \( \lambda \) is defeasibly derivable from \( \Delta \) if, and only if, there exists a defeasible derivation tree \( Y_\lambda(\lambda) \).

Notice that defeasible derivation is a *non-monotonic* process, since the extension of \( \Delta \) with new knowledge cannot avoid the derivation of previously derived literals. Nevertheless, defeasible reasoning is a *monotonic* process, since the extension of \( \Delta \) with new knowledge can make a previously coherent conclusion becomes incoherent, and vice-versa. For example, consider the following knowledge base:

\[
\Delta^3 = \{ 1 \colon c \rightarrow b, 2 \colon b \rightarrow f, 3 \colon c \rightarrow \neg f, 4 \colon c \wedge s \rightarrow f, 5 \colon \top \rightarrow c, 3 < 2, 4 < 3 \}
\]

where \( c, b, f \), and \( s \) stand for “chicken”, “bird”, “fly”, and “scared”, respectively. Clearly, both \( f \) and \( \neg f \) are defeasibly derivable from \( \Delta^3 \), since \( A_1 = \{ \top \rightarrow c, c \rightarrow b, b \rightarrow f \} \models f \) and \( A_2 = \{ \top \rightarrow c, c \rightarrow \neg f \} \models \neg f \). However, because \( 3 < 2 \), \( A_3 \) is considered stronger than \( A_1 \) and, hence, only \( \neg f \) is a coherent conclusion from \( \Delta^1 \). In other words, arguments \( A_1 \) and \( A_2 \) *attack* each other, but \( A_2 \) *defeats* \( A_1 \). Now, suppose that \( \Delta^1 \) is extended, becoming \( \Delta^3 := \Delta^3 \cup \{ 6 \colon \top \rightarrow s \} \). Then, a third argument \( A_3 = \{ \top \rightarrow c, \top \rightarrow s, c \wedge s \rightarrow f \} \models f \) can be constructed based on the extended \( \Delta^3 \) and, since \( 4 < 3 \), the new argument \( A_3 \) *defeats* \( A_2 \) and *reinstates* \( A_1 \). As a result, the previously coherent conclusion \( \neg f \) becomes an incoherent conclusion, and the previously incoherent conclusion \( f \) becomes a coherent conclusion. This idea is illustrated in Figure 3.

It is worthy noticing that, without the precedence rules \( 3 < 2 \) and \( 4 < 3 \), the conflicts between the arguments could not be resolved and, consequently, neither \( f \), nor \( \neg f \) could be accepted as a coherent conclusion from \( \Delta^1 \). When two conflicting defeasible rules have the same strength, we say that they *block* each other.
2.3. Precedence Relations

Let $L_{\Delta}$ be the set of labels used in a knowledge base $\Delta$. A strict partial order over $L_{\Delta}$ is a binary relation $\prec$ such that $\ell \neq \ell'$ (irreflexivity), and if $\ell \prec \ell'$ and $\ell' \prec \ell''$, then $\ell \prec \ell''$ (transitivity), for all $\ell, \ell', \ell'' \in L_{\Delta}$. Clearly, if $\prec$ is an irreflexive and transitive relation, it is also an asymmetric relation (i.e., if $\ell \prec \ell'$, then $\ell' \notin \prec \ell$).

Let $\Pi_{\Delta} = \{\ell \prec \ell' \in \Delta\}$ be the set of precedence rules explicitly declared in $\Delta$. We assume that the transitive closure of $\Pi_{\Delta}$, denoted by $E_{\Delta}$, is a strict partial order over $L_{\Delta}$. Moreover, since precedence rules between non-conflicting defeasible rules are useless, we define $e_{\Delta} = \{\ell \prec \ell' : \ell \prec \ell'\}$. Indeed, the set $E_{\Delta}$ is an explicit precedence relation over defeasible rules declared in $\Delta$. For example, consider the following knowledge base:

$$\Delta^4 = \{1: a \rightarrow \neg f, 2: a \wedge w \rightarrow f, 3: c \rightarrow \neg f, 4: c \wedge s \rightarrow f, 5: c \wedge w \rightarrow \neg f, 6: b \rightarrow f, 7: b \rightarrow w, 8: c \rightarrow b, 9: \top \rightarrow \neg w, 10: \top \rightarrow c, 11: \top \rightarrow s\}$$

where $a$, $f$, $w$, $c$, $s$, and $b$ stand for “animal”, “fly”, “winged”, “chicken”, “scared”, and “bird”, respectively. Let $\Delta^3$ be $\Delta^4 \cup \{2 < 1, 3 < 2, 4 < 3\}$. Then, we have:

- $\Pi_{\Delta}^3 = \{2 < 1, 3 < 2, 4 < 3\}$
- $E_{\Delta}^3 = \{2 < 1, 3 < 1, 3 < 2, 4 < 1, 4 < 2, 4 < 3\}$
- $E_{\Delta}^3 = \{2 < 1, 3 < 2, 4 < 1, 4 < 3\}$

An implicit precedence relation over defeasible rules declared in $\Delta$, based on the criterion of specificity [11], can also be defined. In this work, we adopt a criterion of specificity that favors two aspects of a defeasible rule: precision (amount of information in the rule’s antecedent) and conciseness (number of steps to reach the rule’s antecedent). Let $\ell_1: \phi_1 \rightarrow \lambda_1$ and $\ell_2: \phi_2 \rightarrow \lambda_2$ be conflicting defeasible rules in $\Delta$; let $\Delta_0 = \{\ell \prec \lambda \in \Delta : \phi \neq \top\}$ be the set of defeasible rules of $\Delta$ that are not presumptions; let $\Delta_1 = \Delta_0 \cup \{\top \rightarrow \lambda : \lambda \in \Lambda(\phi_1)\}$ be a knowledge base where all presumptions are reasons to believe in $\lambda_1$; and let $\Delta_2 = \Delta_0 \cup \{\top \rightarrow \lambda : \lambda \in \Lambda(\phi_2)\}$ be a knowledge base where all presumptions are reasons to believe in $\lambda_2$. Then $\ell_1$ is more specific than $\ell_2$, denoted by $\ell_1 \preceq \ell_2$, if and only if each literal $\lambda \in \Lambda(\phi_2)$ is defeasibly derivable from $\Delta_1$, and there is at least one literal $\lambda \in \Lambda(\phi_1)$ that is not defeasibly derivable from $\Delta_2$. Intuitively, $\ell_1 \preceq \ell_2$ means that the antecedent of $\ell_2$ can be derived from the antecedent of $\ell_1$, but not the other way around (i.e., $\ell_1$ is an exception of $\ell_2$). For example, considering $\Delta^4$, $4: c \wedge s \rightarrow f$ is more specific than $3: c \rightarrow \neg f$ (since $c$ is derivable from $c \wedge s$, but $c \wedge s$ is not derivable from $c$). Intuitively, rule 4 is
more precise than rule 3. Analogously, $3: c \rightarrow \neg f$ is more specific than $2: a \land w \rightarrow f$ (since $a \land w$ is derivable from $c$, but $c$ is not derivable from $a \land w$). Intuitively, rule 3 is more concise than rule 2.

Let $\mathcal{I}_\Delta = \{ \ell \prec \ell' : \ell, \ell' \in L_\Delta \text{ and } \ell \not\prec \ell' \}$ be the set of implicit precedence rules automatically synthesized from the defeasible rules declared in $\Delta$. Clearly, $\mathcal{I}_\Delta$ is an irreflexive relation (since the specificity criterion is defined only for conflicting rules), $\mathcal{I}_\Delta$ is an asymmetric relation (since, if $\ell \prec \ell'$, the antecedent of $\ell'$ is defeasibly derivable from the antecedent of $\ell$, but not vice-versa), and $\mathcal{I}_\Delta$ is a transitive relation, with respect to conflicting rules (since, if $\ell \prec \ell'$, $\ell' \prec \ell''$ and $\ell'' \prec \ell'''$, then $\ell \prec \ell'''$ and the antecedent of $\ell'''$ is defeasibly derivable from the antecedent of $\ell$, but not vice-versa). Therefore, $\mathcal{I}_\Delta$ is an implicit precedence relation over defeasible rules declared in $\Delta$. For example, considering $\Delta^4$, we have:

- $\mathcal{I}_{\Delta^4} = \{2 \prec 1, 3 \prec 2, 4 \prec 1, 4 \prec 3, 5 \prec 2, 7 \prec 9\}$

The synthesis of an implicit preference relation is based only on the syntax of the defeasible rules declared in a knowledge base and, therefore, it has the advantage of being a criterion independent of the application domain. However, not all precedence rules can be defined in terms of specificity and, frequently, a knowledge base also contains explicit precedence rules defined by a domain expert. In this case, a mixed preference relation (synthesized by combining explicit and implicit preference relations) may be used. Notice, however, that $\mathcal{E}_\Delta \cup \mathcal{I}_{\Delta_5}$ is not necessarily a strict partial order over $L_{\Delta^5}$ (since explicit and implicit precedence relations can disagree about the relative precedence of two defeasible rules). For example, for the knowledge base $\Delta^5 = \Delta^4 \cup \{5 \prec 4, 9 \prec 7\}$, $\mathcal{E}_{\Delta^5} \cup \mathcal{I}_{\Delta_5}$ is not a strict partial order over $L_{\Delta^5}$, as can be easily verified:

- $\mathcal{E}_{\Delta^5} = \{5 \prec 4, 9 \prec 7\}$
- $\mathcal{I}_{\Delta^5} = \{2 \prec 1, 3 \prec 2, 4 \prec 1, 4 \prec 3, 5 \prec 2, 7 \prec 9\}$

To solve this problem, we propose an algorithm that combines explicit and implicit preference relations, by giving preference to explicit precedence rules. This algorithm starts with $\Pi_\Delta^\omega \coloneqq \mathcal{E}_\Delta \cup \mathcal{I}_\Delta$, then, while $\Pi_\Delta^\omega$ is a cyclic relation, it finds the set $W$ of the weakest edges in a shortest cycle in $\Pi_\Delta^\omega$, and defines $\Pi_{\Delta^2}^\omega \coloneqq \Pi_\Delta^\omega - W$. Given a cycle $C = \langle \ell_1, \ell_2, \ldots, \ell_k, \ell_1 \rangle$, the set $W$ of weakest edges in $C$ is $\{ \ell \prec \ell' \in C : \ell \prec \ell' \not\in \mathcal{E}_\Delta, \ell' \prec \ell'' \in C \text{ and } \ell'' \prec \ell'' \in \mathcal{E}_\Delta \}$. When the algorithm stops, $\Pi_\Delta^\omega$ is an acyclic relation and $\mathcal{E}_\Delta \subseteq \Pi_\Delta^\omega$. Therefore, the transitive closure of $\Pi_\Delta^\omega$, denoted by $\mathcal{M}_\Delta$, is a strict partial order over $L_\Delta$, and $\mathcal{M}_\Delta = \{ \ell \prec \ell' : \ell \prec \ell' \in \mathcal{M}_\Delta \}$ is a mixed precedence relation over defeasible rules declared in $\Delta$. The general idea of this process is depicted in Figure 4, considering an arbitrary situation involving eight labels. In this figure, explicit and implicit preference rules are represented by plain and dotted lines, respectively, and the precedence rules resulting from the transitive closure of the acyclic relation are represented by dashed lines. Particularly, for $\Delta^4$, we have $\mathcal{M}_{\Delta^4} = \{2 \prec 1, 3 \prec 2, 4 \prec 1, 4 \prec 3, 5 \prec 2, 7 \prec 9\}$.

3. The Dialectical Proof Procedure

As discussed in Section 2.2, arguments for and against a conclusion can be extracted from defeasible derivation trees. Arguments are similar to proofs but, since they can be defeated by stronger counterarguments, their conclusions cannot be warranted under all circumstances. In this section, we present the fundamentals of the dialectical proof procedure on which our system is based. Given a knowledge base $\Delta$, this proof procedure can

![Figure 4. Mixed precedence relation synthesis. (a) Weakest edges in cycles in $\mathcal{E}_\Delta \cup \mathcal{I}_{\Delta_5}$; (b) Corresponding acyclic relation $\Pi_{\Delta^5}^\omega$; (c) Mixed precedence relation $\mathcal{M}_{\Delta^5}$](Image 15)
decide whether a conclusion can be coherently inferred from $\Delta$, by analyzing its pros and cons. The dialectical proof procedure is a kind of persuasion dialogue [12], based on two main components: a communication language and a protocol. These components are explained in the next subsections.

3.1. The Communication Language

To communicate its viewpoint about an issue, an agent must use a locution. The communication language specifies the locutions that the agents can utter in a conversation [13]. In this work, we adopt the following locutions, where $\Delta$ is a knowledge base and $\lambda$ is a literal:

- **claim($\lambda$)**, to claim that $\lambda$ is a coherent conclusion from $\Delta$.
- **why($\lambda$)**, to ask for reasons to believe that $\lambda$ is a coherent conclusion from $\Delta$.
- **since($\ell : \varphi \rightarrow \lambda$)**, to argue that $\varphi$ is a reason to believe that $\lambda$ is a coherent conclusion from $\Delta$.
- **agree($\lambda$)**, to agree that $\lambda$ is a coherent conclusion from $\Delta$.
- **retract($\lambda$)**, to retract the claim about $\lambda$ being a coherent conclusion from $\Delta$.

The dialectical proof procedure is modeled as a dialogue between agents pro and con. A speech act is a pair formed by an agent and a locution. A dialogue starts with a speech act pro:claim($\lambda$). The role of pro is to utter locutions defending the claim that $\lambda$ is a coherent conclusion from $\Delta$, and the role of con is to utter locutions raising doubt about the truth of that claim. The attitude of pro is credulous, while the attitude of con is skeptical.

3.2. The Protocol

A dialogue is a finite sequence of speech acts. The record of all speech acts uttered by the agents, since the beginning of a dialogue until a specific moment, is a narrative. A protocol specifies, for each narrative, the next legal speech act. A legal dialogue is a dialogue consisting only of legal speech acts, according to the protocol.

The protocol used in this work is succinctly described in Table 1. In this table, speech act is the last utterance in the current narrative, and A and B are agents with adversary roles. For each speech act, this protocol specifies a legal reply, which can be an attacking or a surrendering reply. The protocol enforces that each reply must be coherent with the all previous locutions uttered by the agents, according to the current narrative.

The turn taking policy is implicitly defined by the reply structure imposed by the protocol (also specified in Table 1). An agent can give more than one reply to a speech act, repeated locutions are not allowed, and tentative replies must obey the order in which they are defined in Table 1.

During a dialogue, a dialectical tree with all relevant pros and cons for the initial claim is recursively built. The dialogue terminates when no legal reply in the current narrative is possible. A speech act is a winner if all its replies are losers; otherwise, if it has at least a winner reply, it is a loser. By definition, speech acts with the locutions agree($\lambda$) and retract($\lambda$) are losers. When a reply is a loser, the agent can backtrack and try another reply. At the end, the initial claim, about $\lambda$ being a coherent conclusion from $\Delta$, is true if pro:claim($\lambda$) is a winner.

For example, consider the following knowledge base:

| Table 1. Protocol: speech acts and reply structure. |
|-----------------------------------|---------------|---------------|
| **Speech act**                  | **Attack**    | **Surrender** |
| A:claim($\lambda$)              | B:why($\lambda$) | B:agree($\lambda$) |
| A:why($\lambda$)                | B:since($\ell : \varphi \rightarrow \lambda$), for $\ell : \varphi \rightarrow \lambda \in \Delta$ | B:retract($\lambda$) |
| A:since($\ell : \varphi \rightarrow \lambda$) | B:why($\lambda'$), for $\lambda' \in \Lambda(\varphi)$ | B:agree($\lambda'$) |
| A:agree($\lambda$)              | nil           | nil           |
| A:retract($\lambda$)            | nil           | nil           |
\[\Delta^b = \{1: b \rightarrow f, 2: c \rightarrow \neg f, 3: c \land s \rightarrow f, 4: c \rightarrow b, 5: \top \rightarrow c, 2 \prec 1, 3 \prec 2\}\]

where \(b\), \(f\), \(c\), and \(s\) stand for “bird”, “fly”, “chicken”, and “scared”, respectively. **Figure 5** shows a dialectical tree warranting that \(\neg f\) is a coherent conclusion from \(\Delta^b\). Winners and losers are marked with \(W\) and \(L\), respectively.

As said before, the agents play different roles in a dialogue: while **pro** defends the claim that \(\lambda\) is a coherent conclusion from \(\Delta\), **con** tries to raise doubts about that claim. Notice, however, that **con** does not defend the opposite claim (i.e., that the complement of \(\lambda\) is a coherent conclusion from \(\Delta\)). Therefore, to win a dispute, **pro** must defeat the rules used by **con**; whereas, to win a dispute, **con** can defeat or block the rules used by **pro**. Moreover, when **pro** wins a dispute, \(\lambda\) is accepted (and, consequently, the complement of \(\lambda\) is rejected); on the other hand, when **con** wins a dispute, \(\lambda\) is rejected (and there is no warranty that the complement of \(\lambda\) is accepted). Indeed, this proof procedure adheres to the open-world assumption [14], according to which the value of a literal can be unknown. For example, both \(p\) and \(\neg p\) are rejected as coherent conclusions from \(\Delta^f = \{1: \top \rightarrow p, 2: \top \rightarrow \neg p\}\), since the rules 1 and 2 block each other (notice that **pro** can agree with \(p\) and \(\neg p\) because it is a credulous agent) (**Figure 6**).

## 4. The Dialogue System Prototype

A prototype\(^1\) of the proposed dialogue system was implemented in Prolog [15]. It runs in interpreted mode, and its commands are executed as standard Prolog queries. The main commands offered by this prototype are described in **Table 2**. By default, the system uses a mixed precedence relation and runs in verbose mode.

In the knowledge representation language used in the prototype, the symbols \(\neg\), \(\land\), \(\rightarrow\), and \(\prec\) are replaced by the operators not, and, then, and precedes, respectively, the literal \(\top\) is replaced by the keyword true, and defeasible rules can contain free variables. For instance, **Figure 7** (*left, top*) shows a knowledge base coded in this new representation language and saved in a file named kb.pl.

---

\(^1\)Available at [www.ime.usp.br/~slago/dsp.zip](http://www.ime.usp.br/~slago/dsp.zip).
Figure 7. Dialogue System Prototype: knowledge base representation, precedence relations and query’s output.

Table 2. Main commands offered by the dialogue system prototype.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>kb # literal</td>
<td>Asks the system whether literal is a coherent conclusion from kb.</td>
</tr>
<tr>
<td>precedence_relations(kb)</td>
<td>Shows all the three precedence relations synthesized from kb.</td>
</tr>
<tr>
<td>explicit</td>
<td>Choose explicit precedence relation to resolve conflicts.</td>
</tr>
<tr>
<td>implicit</td>
<td>Choose implicit precedence relation to resolve conflicts.</td>
</tr>
<tr>
<td>mixed</td>
<td>Choose mixed precedence relation to resolve conflicts.</td>
</tr>
<tr>
<td>verbose</td>
<td>Alternate between <em>verbose</em> and <em>non-verbose</em> mode. If the <em>verbose</em> mode is active, the user can see each step of the reasoning process; otherwise, he can see only the final result of that process.</td>
</tr>
</tbody>
</table>

The command implemented by the predicate precedence_relations/1 shows the three kinds of precedence relations synthesized from a specific knowledge base. For instance, the precedence relations for the knowledge base kb.pl are shown in Figure 7 (left, bottom).

The command implemented by the predicate #/2 allows the user asking whether a literal is a coherent conclusion from a knowledge base. Only *ground* literals are allowed in queries and, at each query, each defeasible rule with variables is automatically replaced by one of its *ground instances*, according to the literal used in the query. For instance, the result of the query kb # fly(tina) is shown in Figure 7 (right).

The implemented prototype was tested with a series of benchmarking examples found in the literature and intuitively coherent results were obtained for all of them.

As future steps, we plan to study the formal properties of the dialogue system prototype, with respect to well known semantics for argumentation systems [5], as well as to develop a graphical interface to show the dialectical tree structure and the relations between its arguments and counterarguments.

5. Conclusions

The ability of dealing with inconsistent knowledge bases is relevant for many practical applications. As it is well known, in such applications, inconsistency arises mainly due to the existence of rules with exceptions. Thus, one way of coping with inconsistency is to give precedence to exceptions. Based on this idea, this paper proposes a dialogue system for coherent reasoning with inconsistent knowledge bases, which resolves conflicts among defeasible rules by using precedence relations of three different kinds.

More specifically, this paper 1) shows how explicit and implicit precedence relations can be automatically synthesized from an inconsistent knowledge base and also how they can be combined to synthesize a mixed precedence relation (where explicit precedence rules can override conflicting implicit precedence rules); 2) presents a dialectical proof procedure that can be used to decide whether a specific conclusion can, or cannot, be
coherently inferred from an inconsistent knowledge base; 3) implements a prototype system for coherent reasoning with inconsistent knowledge bases.

Future extensions of this work are the study of the formal properties of the proposed system and the development of a graphical interface for it.

Acknowledgements

This research (project number 800476/2014-0) is supported by CNPq (Brazilian National Counsel of Technological and Scientific Development), under grant numbers 305484/2012-5 and 102869/2015-4.

References


