Journal of Computer and Communications, 2015, 3, 11-19
Published Online August 2015 in SciRes. http://www.scirp.org/journal/jcc
http://dx.doi.org/10.4236/jcc.2015.38002

Scientific Research Publishing

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

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 Δ 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 Δ. Now, suppose

How to cite this paper: do Lago Pereira, S., dos Santos, L.F.Z. and de Lira, L.N. (2015) A Dialogue System for Coherent Reasoning with Inconsistent Knowledge Bases. *Journal of Computer and Communications*, **3**, 11-19.
http://dx.doi.org/10.4236/jcc.2015.38002

that the new piece of knowledge "*Tweety is a penguin*" is inserted into Δ, resulting in a new knowledge base Δ'. Then, both "*Tweety flies*" and "*Tweety does not fly*" can be inferred from Δ', and that is not a coherent reasoning. One way of restoring the consistency of Δ' 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 Δ'. 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* λ is an atom α or a negated atom $\neg\alpha$. Two literals λ and λ' 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 $\lambda_1 \wedge \cdots \wedge \lambda_k$, where each λ_i $(1 \le i \le k)$ is a literal. Technically, a conjunction $\varphi = \lambda_1 \wedge \cdots \wedge \lambda_k$ 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 $\Lambda(\varphi) = \varnothing$.

A *defeasible rule* is an expression $\varphi \to \lambda$, where φ 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 λ, if there is no evidence contrary to λ. A defeasible rule is *consistent* if the set $\Lambda(\varphi) \cup \{\lambda\}$ has no complementary literals. A defeasible rule $\top \to \lambda$ is a *presumption*. A *labeled defeasible rule* is an expression $\ell : \pi$, where π is a defeasible rule and ℓ is a unique *label* identifying π. Two labeled defeasible rules $\ell : \pi$ and $\ell' : \pi'$ are called *conflicting* defeasible rules, denoted by $\ell \Diamond \ell'$, 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 ℓ and ℓ' are labels of conflicting defeasible rules, stating that the rule ℓ *precedes* the rule ℓ' (*i.e.*, that the priority of rule ℓ is *higher* than the priority of the rule ℓ'). 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* Δ is a finite set of consistent labeled defeasible rules and precedence rules. For example,

$$\Delta^1 = \{1 : p \to b, 2 : b \to f, 3 : p \to \neg f, 4 : \top \to p, 3 \prec 2\}$$

is a knowledgebase, where p, b, and f stand, respectively, for "*penguin*", "*bird*", and "*fly*". In this knowledge base, the defeasible rule $2 : b \to f$ states that "*birds fly*", the defeasible rule $3 : p \to \neg f$ 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 $\varphi \to \lambda$ states that the literals in $\Lambda(\varphi)$ are reasons to believe in the literal

λ, if there is no counter evidence to λ. In this context, the symbols $\neg$, $\wedge$ and $\rightarrow$ are not interpreted as in classical logic, since neither *modus ponens* (*i.e.*, $\{\varphi, \varphi \rightarrow \lambda\} \vdash \lambda$), nor *modus tollens* (*i.e.*, $\{\varphi \rightarrow \lambda, \neg\lambda\} \vdash \neg\varphi$) 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 λ. Anyway, a necessary (although not sufficient) condition to believe in a literal λ is that it can be, at least, *defeasibly derived* from the knowledge base.

A *defeasible derivation tree* of a literal λ from a knowledge base Δ, denoted by $\Upsilon_\Delta(\lambda)$, is a tree such that:

- The root of $\Upsilon_\Delta(\lambda)$ is labeled with the literal λ.
- For each node of $\Upsilon_\Delta(\lambda)$ labeled with a literal λ', there exists a defeasible rule $\varphi \rightarrow \lambda' \in \Delta$.
- If $\varphi = \top$, then the node labeled with λ' is a leaf in $\Upsilon_\Delta(\lambda)$; otherwise, if $\varphi = \lambda_1 \wedge \cdots \wedge \lambda_k$, then that node has exactly k children nodes, which are labeled with $\lambda_1, \cdots, \lambda_k$, 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 Δ^2 is depicted in **Figure 1**.

$$\Delta^2 = \{1: \top \rightarrow p, 2: \top \rightarrow q, 3: \top \rightarrow r, 4: p \wedge q \rightarrow s, 5: r \rightarrow t, 6: s \wedge t \rightarrow u\}$$

A literal λ is *defeasibly derivable* from Δ if, and only if, there exists a defeasible derivation tree $\Upsilon_\Delta(\lambda)$. For example, as shown in **Figure 2**, both literals f ("*Tweety flies*") and $\neg f$ ("*Tweety does not fly*") are defeasibly derivable from the knowledge base Δ^1.

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

$$\Delta^3 = \{1: c \rightarrow b, 2: b \rightarrow f, 3: c \rightarrow \neg f, 4: c \wedge s \rightarrow f, 5: \top \rightarrow c, 3 \prec 2, 4 \prec 3\}$$

where c, b, f, and s stand for "*chicken*", "*bird*", "*fly*", and "*scared*", respectively. Clearly, both f and $\neg f$ are defeasibly derivable from Δ^3, since $A_1 = \{\top \rightarrow c, c \rightarrow b, b \rightarrow f\} \vdash f$ and $A_2 = \{\top \rightarrow c, c \rightarrow \neg f\} \vdash \neg f$. However, because $3 \prec 2$, A_2 is considered stronger than A_1 and, hence, only $\neg f$ is a coherent conclusion from Δ^3. In other words, arguments A_1 and A_2 *attack* each other, but A_2 *defeats* A_1. Now, suppose that Δ^3 is extended, becoming $\Delta^3 := \Delta^3 \cup \{6: \top \rightarrow s\}$. Then, a third argument $A_3 = \{\top \rightarrow c, \top \rightarrow s, c \wedge s \rightarrow f\} \vdash f$ can be constructed based on the extended Δ^3 and, since $4 \prec 3$, 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 $3 \prec 2$ and $4 \prec 3$, the conflicts between the arguments could not be resolved and, consequently, neither f, nor $\neg f$ could be accepted as a coherent conclusion from Δ^3. When two conflicting defeasible rules have the same strength, we say that they *block* each other.

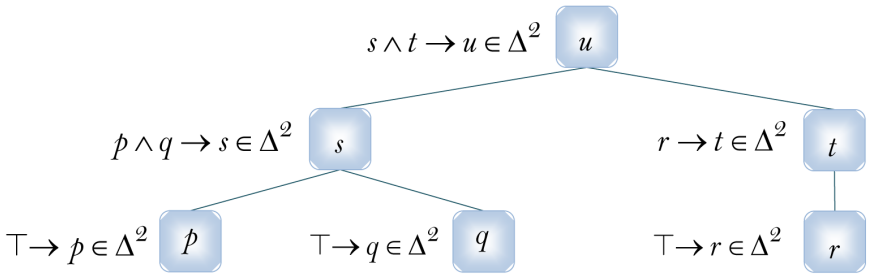

Figure 1. Defeasible derivation tree of u from Δ^2.

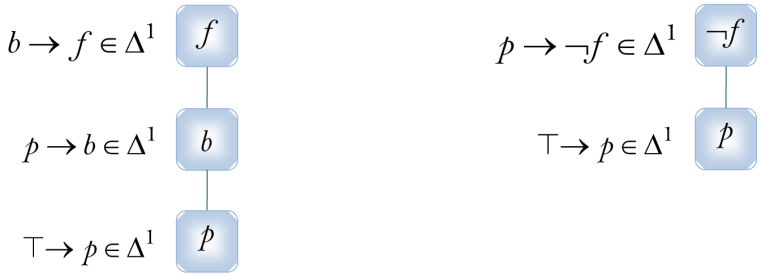

Figure 2. Defeasible derivation trees of f and $\neg f$ from Δ^1.

Figure 3. Attack, defeat and reinstatement.

2.3. Precedence Relations

Let L_Δ be the set of labels used in a knowledge base Δ. A *strict partial order* over L_Δ is a binary relation $\prec$ such that $\ell \not\prec \ell$ (*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 $\ell' \not\prec \ell$).

Let $\Pi^e_\Delta = \{\ell \prec \ell' \in \Delta\}$ be the set of precedence rules *explicitly* declared in Δ. We assume that the transitive closure of Π^e_Δ, denoted by $\mathcal{E}^c_\Delta$, is a strict partial order over L_Δ. Moreover, since precedence rules between non-conflicting defeasible rules are useless, we define $\mathcal{E}_\Delta = \{\ell \prec \ell' \in \mathcal{E}^c_\Delta : \ell \lozenge \ell'\}$. Indeed, the set $\mathcal{E}_\Delta$ is an *explicit precedence relation* over defeasible rules declared in Δ. For example, consider the following knowledge base:

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

where a, f, w, c, s, and b stand for "*animal*", "*fly*", "*winged*", "*chicken*", "*scared*", and "*bird*", respectively. Let Δ^5 be $\Delta^4 \cup \{2 \prec 1, 3 \prec 2, 4 \prec 3\}$. Then, we have:

- $\Pi^e_{\Delta^5} = \{2 \prec 1, 3 \prec 2, 4 \prec 3\}$
- $\mathcal{E}^c_{\Delta^5} = \{2 \prec 1, 3 \prec 1, 3 \prec 2, 4 \prec 1, 4 \prec 2, 4 \prec 3\}$
- $\mathcal{E}_{\Delta^5} = \{2 \prec 1, 3 \prec 2, 4 \prec 1, 4 \prec 3\}$

An implicit precedence relation over defeasible rules declared in Δ, 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 $\ell_1 : \varphi_1 \to \lambda_1$ and $\ell_2 : \varphi_2 \to \lambda_2$ be conflicting defeasible rules in Δ; let $\Delta_0 = \{(\varphi \to \lambda) \in \Delta : \varphi \neq \top\}$ be the set of defeasible rules of Δ that are not presumptions; let $\Delta_1 = \Delta_0 \cup \{\top \to \lambda : \lambda \in \Lambda(\varphi_1)\}$ be a knowledge base where all presumptions are reasons to believe in λ_1; and let $\Delta_2 = \Delta_0 \cup \{\top \to \lambda : \lambda \in \Lambda(\varphi_2)\}$ be a knowledge base where all presumptions are reasons to believe in λ_2. Then ℓ_1 is *more specific* than ℓ_2, denoted by $\ell_1 \lhd \ell_2$, if and only if each literal $\lambda \in \Lambda(\varphi_2)$ is defeasibly derivable from Δ_1, and there is at least one literal $\lambda \in \Lambda(\varphi_1)$ that is not defeasibly derivable from Δ_2. Intuitively, $\ell_1 \lhd \ell_2$ means that the antecedent of ℓ_2 can be derived from the antecedent of ℓ_1, but not the other way around (*i.e.*, ℓ_1 is an *exception* of ℓ_2). For example, considering Δ^4, $4: c \wedge s \to f$ is more specific than $3: c \to \neg f$ (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 $2: a \wedge w \rightarrow f$ (since $a \wedge w$ is derivable from c, but c is not derivable from $a \wedge w$). Intuitively, rule 3 is more *concise* than rule 2.

Let $\mathcal{I}_\Delta = \{\ell \prec \ell' : \ell, \ell' \in L_\Delta \text{ and } \ell \triangleleft \ell'\}$ be the set of implicit precedence rules *automatically synthesized* from the defeasible rules declared in Δ.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 $\ell \triangleleft \ell'$, the antecedent of ℓ' is defeasibly derived from the antecedent of ℓ, but not vice-versa), and $\mathcal{I}_\Delta$ is a transitive relation, with respect to conflicting rules (since, if $\ell \triangleleft \ell'$, $\ell' \triangleleft \ell''$ and $\ell'' \triangleleft \ell'''$, then $\ell \Diamond \ell'''$ and the antecedent of ℓ''' is defeasibly derivable from the antecedent of ℓ, but not vice-versa).Therefore, $\mathcal{I}_\Delta$ is an *implicit precedence relation* over defeasible rules declared in Δ. For example, considering Δ^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 $\mathcal{E}_\Delta \cup \mathcal{I}_\Delta$ is not necessarily a strict partial order over L_Δ (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_{Δ^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 $\Pi^m_\Delta := \mathcal{E}_\Delta \cup \mathcal{I}_\Delta$. Then, while Π^m_Δ is a cyclic relation, it finds the set W of the *weakest edges* in a shortest cycle in Π^m_Δ, and defines $\Pi^m_\Delta := \Pi^m_\Delta - W$. Given a cycle $C = \langle \ell_1 \prec \ell_2, \cdots, \ell_k \prec \ell_1 \rangle$, the set W of *weakest edges* in C is $\{\ell \prec \ell' \in C : \ell \prec \ell' \notin \mathcal{E}_\Delta, \ell' \prec \ell'' \in C \text{ and } \ell' \prec \ell'' \in \mathcal{E}_\Delta\}$. When the algorithm stops, Π^m_Δ is an acyclic relation and $\mathcal{E}_\Delta \subseteq \Pi^m_\Delta$. Therefore, the transitive closure of Π^m_Δ, denoted by $\mathcal{M}^c_\Delta$, is a strict partial order over L_Δ and $\mathcal{M}_\Delta = \{\ell \prec \ell' \in \mathcal{M}^c_\Delta : \ell \Diamond \ell'\}$ is a *mixed precedence relation* over defeasible rules declared in Δ.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 Δ^5, we have $\mathcal{M}_{\Delta^5} = \{2 \prec 1, 3 \prec 2, 4 \prec 1, 4 \prec 3, 5 \prec 2, 5 \prec 4, 9 \prec 7\}$.

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 Δ, this proof procedure can

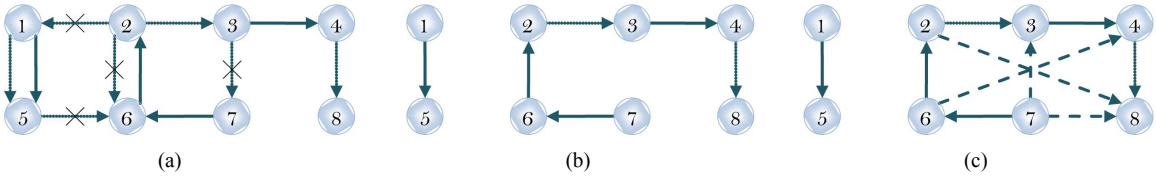

Figure 4. Mixed precedence relation synthesis. (a) *Weakest edges in cycles in* $\mathcal{E}_\Delta \cup \mathcal{I}_\Delta$; (b) *Corresponding acyclic relation* Π^m_Δ; (c) *Mixed precedence relation* $\mathcal{M}_\Delta$.

decide whether a conclusion can be coherently inferred from Δ, by analyzing its pros and cons. The dialectical proof procedure is a kind *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 Δ is a knowledge base and λ is a literal:

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

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 λ is a coherent conclusion from Δ, 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 λ being a coherent conclusion from Δ, 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*
$A:claim(\lambda)$	$B:why(\lambda)$	$B:agree(\lambda)$
$A:why(\lambda)$	$B:since(\ell:\varphi\to\lambda)$, for $\ell:\varphi\to\lambda\in\Delta$	$B:retract(\lambda)$
$A:since(\ell:\varphi\to\lambda)$	$B:why(\lambda')$, for $\lambda'\in\Lambda(\varphi)$ $B:since(\ell':\varphi'\to\lambda')$, for $\ell':\varphi'\to\lambda'\in\Delta$, if λ and λ' are complementary literals	$B:agree(\lambda)$
$A:agree(\lambda)$	*nil*	*nil*
$A:retract(\lambda)$	*nil*	*nil*

$$\Delta^6 = \{1: b \to f, 2: c \to \neg f, 3: c \wedge s \to f, 4: c \to b, 5: \top \to c, 2 \prec 1, 3 \prec 2\}$$

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 Δ^6. 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 λ is a coherent conclusion from Δ, *con* tries to raise doubts about that claim. Notice, however, that *con* does not defend the opposite claim (*i.e.*, that the complement of λ is a coherent conclusion from Δ). 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, λ is *accepted* (and, consequently, the complement of λ is *rejected*); on the other hand, when *con* wins a dispute, λ is *rejected* (and there is no warranty that the complement of λ 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 $\Delta^7 = \{1: \top \to p, 2: \top \to \neg p\}$, 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 $\neg$, $\wedge$, $\to$, 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.

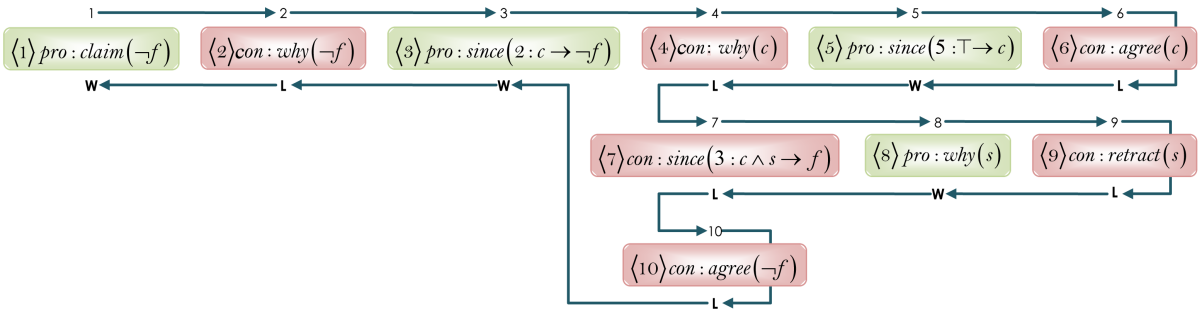

Figure 5. Dialectical tree warranting that $\neg f$ is a coherent conclusion from Δ^6.

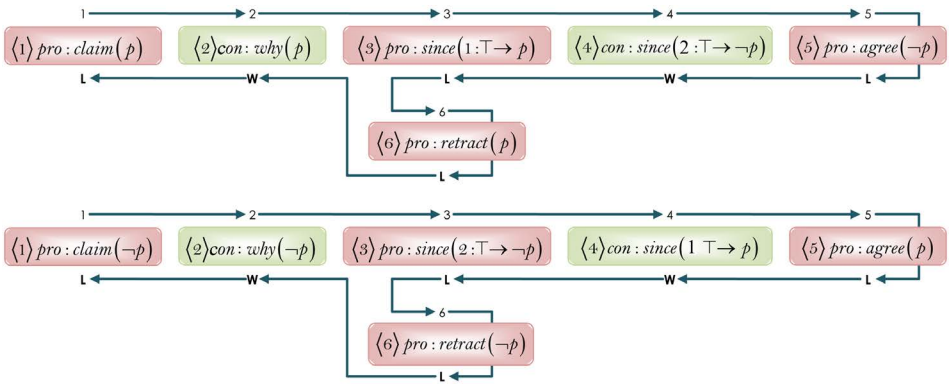

Figure 6. The open-world assumption.

[1]Available at www.ime.usp.br/~slago/dsp.zip.

```
% kb.pl – a simple knowledge base
1: bird(X) then fly(X).
2: chicken(X) then not fly(X).
3: chicken(X) and scared(X) then fly(X).
4: dead(X) then not fly(X).
5: chicken(X) then bird(X).
6: true then chicken(tina).
7: true then chicken(bina).
8: true then scared(tina).
   4 precedes 3.
```

```
?- precedence_relations(kb).
Explicit: [4<3]
Implicit: [2<1, 3<2]
Mixed...: [2<1, 3<2, 4<1, 4<3]
```

```
?- kb # fly(tina).
<1> pro : claim(fly(tina))
<2> con : why(fly(tina))
<3> pro : since(1:bird(tina) then fly(tina))
<4> con : why(bird(tina))
<5> pro : since(5:chicken(tina) then bird(tina))
<6> con : why(chicken(tina))
<7> pro : since(6:true then chicken(tina))
<8> con : agree(chicken(tina))
<9> con : agree(bird(tina))
<10> con : since(2:chicken(tina) then not fly(tina))
<11> pro : since(3:chicken(tina) and scared(tina) then fly(tina))
<12> con : why(scared(tina))
<13> pro : since(8:true then scared(tina))
<14> con : agree(scared(tina))
<15> con : since(4:dead(tina) then not fly(tina))
<16> pro : why(dead(tina))
<17> con : retract(dead(tina))
<18> con : agree(fly(tina))
Accepted: fly(tina)
```

Figure 7. Dialogue System Prototype: knowledge base representation, precedence relations and query's output.

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

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

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

[1] Carnielli, W.A. and Marcos, J. (2001) Ex Contradictione Non Sequitur Quodlibet. *Proceedings of the II Annual Conference on Reasoning and Logic*, Bucharest, July 2001, 89-109. http://wslc.math.ist.utl.pt/ftp/pub/MarcosJ/01-CM-ECNSQL.pdf

[2] Walton, D. (2006) Fundamentals of Critical Argumentation. Cambridge University Press, Cambridge.

[3] Potyka, N. and Thimm, M. (2014) Consolidation of Probabilistic Knowledge Bases by Inconsistency Minimization. *Proceedings of the* 21*st European Conference on Artificial Intelligence*, Prague, 27 May 2014, 729-734. http://www.mthimm.de/pub/2014/Potyka_2014.pdf

[4] Efstathiou, V. (2010) Algorithms for Computational Argumentation in Artificial Intelligence. Ph.D. Thesis, University College London, London. http://discovery.ucl.ac.uk/1301992/1/1301992.pdf

[5] Dung, P.M. (1995) On the Acceptability of Arguments and Its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and *N*-Person Games. *Artificial Intelligence*, **77**, 321-357. http://www.sciencedirect.com/science/article/pii/000437029400041X

[6] Modgil, S.J. and Prakken, H. (2014) The *ASPIC*+ Framework for Structured Argumentation: A Tutorial. *Argument and Computation*, **5**, 31-62. http://www.cs.uu.nl/groups/IS/archive/henry/ASPICtutorial.pdf

[7] Gorogiannis, N. and Hunter, A. (2011) Instantiating Abstract Argumentation with Classical Logic Arguments: Postulates and Properties. *Artificial Intelligence*, **175**, 1479-1497. http://www0.cs.ucl.ac.uk/staff/a.hunter/papers/arglog.pdf http://dx.doi.org/10.1016/j.artint.2010.12.003

[8] García, A.J. and Simari, G.R. (2004) Defeasible Logic Programming: An Argumentative Approach. *Theory and Practice of Logic Programming*, **4**, 95-138. http://cs.uns.edu.ar/~ajg/papers/2004TPLPGarciaSimari.pdf http://dx.doi.org/10.1017/S1471068403001674

[9] Verheij, B. (1999) Logic, Context and Valid Inference or: Can There Be a Logic of Law? In: Jaap van den Herik, H., *et al*., Eds., *Legal Knowledge Based Systems*, GNI, Nijmegen, 109-121. http://www.ai.rug.nl/~verheij/publications/pdf/jurix99.pdf

[10] Kowalski, R. (1974) Predicate Logic as a Programming Language. Information Processing, North Holland Publishing Co., Amsterdam, 569-574. http://www.doc.ic.ac.uk/~rak/papers/IFIP%2074.pdf

[11] Stolzenburg, F., *et al*. (2002) Computing Generalized Specificity. *Journal of Applied Non-Classical Logics*, **12**, 1-27. http://link.springer.com/chapter/10.1007/978-94-017-1737-3_4

[12] Besnard, P. and Hunter, A. (2008) Elements of Argumentation. MIT Press, Cambridge. https://mitpress.mit.edu/sites/default/files/titles/content/9780262026437_sch_0001.pdf

[13] Prakken, H. (2006) Formal Systems for Persuasion Dialogue. *Knowledge Engineering Review*, **21**, 163-188. http://www.cs.uu.nl/groups/IS/archive/henry/dgreview.pdf http://dx.doi.org/10.1017/S0269888906000865

[14] Russell, S. and Norvig, P. (2010) Artificial Intelligence: A Modern Approach. 3rd Edition, Prentice Hall, Upper Saddle River.

[15] Bratko, I. (2011) Prolog Programming for Artificial Intelligence. 4th Edition, Pearson, Canada.