The paper introduces a logical framework for negotiation among dishonest agents. The framework relies on the use of abductive logic programming as a knowledge representation language for agents to deal with incomplete information and preferences. The paper shows how intentionally false or inaccurate information of agents can be encoded in the agents’ knowledge bases. Such disinformation can be effectively used in the process of negotiation to have desired outcomes by agents. The negotiation processes are formulated under the answer set semantics of abductive logic programming, and they enable the exploration of various strategies that agents can employ in their negotiation. A preliminary implementation has been developed using the ASP-Prolog platform.

Categories and Subject Descriptors: I.2.3 [Deduction and Theorem Proving]: Logic Programming; I.2.4 [Knowledge Representation Formalisms and Methods]: Representations; I.2.11 [Distributed Artificial Intelligence]: Multiagent systems

General Terms: Theory, Design

Additional Key Words and Phrases: Negotiation, Abductive Logic Programming, Dishonesty, Negotiation Agent, Answer Set Programming

ACM Reference Format:

DOI: http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION AND MOTIVATION

Bargaining and negotiation are fundamental forms of social interaction, appearing at all levels of society—e.g., from bargaining prices in a market to resolving labor disputes to negotiating peace agreements [Lewicki et al. 2003; Pruitt and Kim 2004; Thompson 2005]. Research in the field of intelligent software agents and multi-agent systems has placed a strong emphasis in investigating models of interactions, coordination, and cooperation among agents. In particular, the role of bargaining and negotiation has been widely recognized as central, and has attracted the attention of researchers from the field of Artificial Intelligence, as well as from several inter-disciplinary domains (e.g., economics, mathematics, psychology, sociology) [Zlotkin and Rosenschein 1991; Kraus 2001; Bui and Shakun 1996]. Indeed, intelligent agents capable of bargaining on behalf of humans are becoming a concrete possibility (e.g., [Meyer et al. 2004; Sandholm 2002; Neumann et al. 2003; Shim and Hsiao 1999; Kersten and Lai 2007]).

Negotiation, viewed as the process whereby agents communicate and interact to achieve a mutually acceptable agreement, is at the foundation of coordination and cooperation among both human and artificial agents. It appears when agents are being cooperative as well as when they try to achieve their own, self-interested, goals. A large variety of models for negotiation have been proposed in the literature (e.g., [Amgoud et al. 2006; Chen et al. 2006; Fatima et al. 2004; 2006; 2009; Nguyen and Jennings 2004; 2005; Kraus et al. 2004; Sandholm 2002; Neumann et al. 2003; Shim and Hsiao 1999; Kersten and Lai 2007]).
Languages for negotiation have been studied in [Wooldridge and Parsons 2000]. Negotiation models have been classified according to several criteria. For example, Jennings et al. [Jennings et al. 2000] offered a broad classification of research in negotiation based on whether the emphasis is on protocols (e.g., rules of interaction), objects (e.g., single vs. multi-issue bargaining), or decision models. Alternative classifications have been proposed, based on the underlying reasoning mechanisms used to guide the negotiation processes (e.g., negotiation models based on game theory, on argumentation, on logic programming) or on whether the focus is on theoretical (i.e., description, specification, and reasoning about the key features of negotiation agents) or computational models (i.e., development and implementation of key data structures of negotiation agents) [Lopes et al. 2008]. Some of the key challenges for negotiation models are multi-issues, incomplete information, preferences, and goal changes (e.g., [Fatima et al. 2006; 2009; Rahwan et al. 2004]). Some of these issues are illustrated in the next example.

Example 1.1. Let us consider a dialogue between a buyer agent \( b \) and a seller \( s \):

\( b_1 \) : “I like a digital camera manufactured by \( C \). I want one that has good quality at a discounted price.”
\( s_1 \) : “The product \( A \) is made by manufacturer \( C \) and has good quality. We provide a discounted price to students.” (In reality, the seller does not know the quality of \( A \).)
\( b_2 \) : “I am not a student.”
\( s_2 \) : “The product \( B \) by the manufacturer \( D \) also has good quality and it is provided at a low price for every customer paying in cash.” (In reality, the seller knows that \( B \) is not of good quality.)
\( b_3 \) : “I am willing to consider products of \( D \) only at the lowest price.”
\( s_3 \) : “If you join our mailing list and pay cash, we can provide the products made by \( D \) at the lowest price.”
\( b_4 \) : “I’d like to join the list and purchase at the price.” (In reality, the buyer does not want to join the list and will block every email from the shop with a spam filter.)

The above simple scenario shows that a negotiation between real-life agents involves reasoning with incomplete information, preferences, goal changes, multiple issues, and dishonesty. Here, the buyer does not prefer products made by \( D \). The seller, at the beginning of the negotiation, does not know that the buyer is not a student—this fact is learned during the negotiation. Furthermore, the seller tries to sell something else during the negotiation (a different product and a different configuration) and makes untruthful statements about the quality of the products.

Although the development of negotiation models has progressed tremendously in the last ten years, Lopes et al. [Lopes et al. 2008], in a recent survey, points out that “most computational models are being used successfully in a wide variety of real world domains, but often without any rigorous theoretical underpinning.” Providing rigorous theoretical foundations for negotiation is, therefore, an important endeavor in the development of negotiation agents and, in general, multi-agent systems. The previous example also highlights the need to study negotiation agents who might lie or make untruthful statements—since dishonesty is an (unfortunate) trait of human beings. To the best of our knowledge, there have been very few proposals of negotiation systems that consider agents who might be dishonest [Sakama et al. 2011; Zlotkin and Rosenschein 1991].

The goal of this paper is to develop an abstraction of real-life negotiation where agents may behave dishonestly. The main contributions of the paper are

— The notion of abductive programs with disinformation (ALD-programs), that can represent disinformation and realize dishonest reasoning by agents (Section 2).
— The formalization of negotiation using negotiation knowledge bases, represented by ALD-programs, and exploration of different strategies used in negotiation (Sections 3 and 4).
— A platform for negotiation systems (Section 5).

Addressing the issues of negotiation with incomplete information, preferences and disinformation in a single framework makes our work significantly different from previously developed models of negotiation using logic programming, abduction, and argumentation (e.g., [Amgoud et al. 2006; Chen et al. 2006; Kakas and...
Moraïtis 2006; Sadri et al. 2002; Sakama and Inoue 2007; Son and Sakama 2009; Rahwan et al. 2003), or using utility-based approaches (e.g., [Fatima et al. 2004; 2005]). To this end, we employ abductive logic programming (ALP) in formalizing the proposed negotiation framework. We relate our work to other proposals and discuss future work in Section 6.

2. DISHONEST REASONING BY ABDUCTIVE LOGIC PROGRAMMING

In this section, we review the notion of abductive logic programs with preferences [Sakama and Inoue 2000] and introduce the notion of abductive logic programs with disinformaton.

2.1. Abductive Programs with Preferences

A (logic) program consists of rules of the form:

\[ \ell_1; \ldots; \ell_l \leftarrow \ell_{l+1}, \ldots, \ell_m, \text{not} \ \ell_{m+1}, \ldots, \text{not} \ \ell_n \]  

where each \( \ell_i \) (\( n \geq m \geq l \geq 0 \)) is an atom \( a \) or its negation \( \neg a \) from a finite propositional language. Observe that a rule with variables is viewed simply as a shorthand for the set of all ground instances. A literal is either an atom or its negation; for an atom \( a \), we have that \( \neg \neg a = a \). The syntax \( \text{not} \ a \) denotes a negation as failure literal (naf-literal). The symbol `\!' represents a disjunction. The left-hand side and the right-hand side of a rule \( r \) of the form (1) are called the head and the body of \( r \), respectively. head(\( r \)) denotes the set \( \{\ell_1, \ldots, \ell_l\} \); pos(\( r \)) and neg(\( r \)) denote \( \{\ell_{l+1}, \ldots, \ell_m\} \) and \( \{\ell_{m+1}, \ldots, \ell_n\} \), respectively. A non-disjunctive rule is a rule having a single literal in its head (\( l = 1 \)). A rule with empty head (\( l = 0 \)) is a constraint, while a non-disjunctive rule with empty body (i.e., \( m = n = 1 \)) is a fact. A fact \( \ell \leftarrow \) is simply identified with the literal \( \ell \).

Given a set of literals \( X \), let us denote with \( \text{comp}(X) = X \cup \{\neg \ell | \ell \in X\} \).

The semantics of a program is given by the answer set semantics [Gelfond and Lifschitz 1990]. Let \( X \) be a set of ground literals. \( X \) is consistent if there is no atom \( a \) such that \( \{a, \neg a\} \subseteq X \). The body of a rule \( r \) of the form (1) is satisfied by \( X \) if neg(\( r \)) \( \cap \) \( X \) = \( \emptyset \) and pos(\( r \)) \( \subseteq \) \( X \). A rule of the form (1) with a non-empty head is satisfied by \( X \) if either its body is not satisfied by \( X \) or head(\( r \)) \( \cap \) \( X \) \( \neq \emptyset \). A constraint is satisfied by \( X \) if its body is not satisfied by \( X \). By \( X \models r \) we denote that \( r \) is satisfied by \( X \).

For a consistent set \( S \) of ground literals and a program \( \Pi \), the reduct of \( \Pi \) w.r.t. \( S \), denoted by \( \Pi^S \), is the program obtained from the set of all ground instances of \( \Pi \) by deleting (i) each rule \( r \) such that neg(\( r \)) \( \cap \) \( S \) \( \neq \emptyset \), and (ii) all naf-literals in the bodies of the remaining rules.

\( S \) is an answer set of \( \Pi \) [Gelfond and Lifschitz 1990] if it satisfies the following conditions: (i) \( \Pi \) does not contain any naf-literals (i.e., \( m = n \) in every rule of \( \Pi \)) then \( S \) is a minimal consistent set of literals that satisfies all the rules in \( \Pi \); and (ii) \( \Pi \) contains some naf-literals (i.e., \( m < n \) in some rules of \( \Pi \)), then \( S \) is an answer set of \( \Pi \) if \( S \) is an answer set of \( \Pi^S \). Note that \( \Pi^S \) does not contain naf-literals and its answer sets are defined in (i). A program \( \Pi \) is consistent if it has an answer set, otherwise it is inconsistent.

Several extensions of the logic programming language have been introduced to ease the use of logic programming in knowledge representation and reasoning. In this paper, we will make use of choice atoms of the form \( l\{\ell_1, \ldots, \ell_m\} \) where \( \ell_j \)'s are literals and \( l \leq u \) are integers. The semantics of logic programs with choice atoms can be found in [Simons et al. 2002].

An abductive (logic) program \( \mathcal{P} \) is a pair \( \langle \Pi, \mathcal{A} \rangle \) where \( \Pi \) and \( \mathcal{A} \) are (propositional) programs. \( \Pi \) is called the background knowledge, while \( \mathcal{A} \) is called the set of abducibles. An abducible \( a \in \mathcal{A} \) is also called an abducible rule (resp. abducible fact) if \( a \) is a rule (resp. a fact). An abducible containing variables is a shorthand for the set of its ground instances. So any ground instance \( \rho(r) \) of an element \( r \) from \( \mathcal{A} \) is also an abducible and is written as \( \rho(r) \in \mathcal{A} \). Without loss of generality, we assume that literals in the heads of rules of \( \mathcal{A} \) do not occur in the heads of rules of \( \Pi \) [Kakas et al. 1998]. Abducibles are hypothetical rules which are used to account for observations together with the background knowledge \( \Pi \).

\textbf{Definition 2.1 (Belief Set).} A set \( S \) of literals is a belief set of \( \langle \Pi, \mathcal{A} \rangle \) if \( S \) is an answer set of \( \Pi \cup E \) for some \( E \subseteq \mathcal{A} \).
Let $X$ be a set of literals. We define

$$X_{\mathcal{A}} = \{ r \mid r \in \mathcal{A}, \text{head}(r) \cap X \neq \emptyset, X \models \text{body}(r) \}$$

(2)

representing the set of abducibles that are satisfied by $X$ and whose body is true in $X$. Each belief set $S$ of $P = (\Pi, \mathcal{A})$ reduces to an answer set of $\Pi$ when $S_{\mathcal{A}} = \emptyset$. It is easy to see that $S_{\mathcal{A}}$ is the set of rules in $\mathcal{A}$ that are needed in the creation of $S$. An abductive program $\langle \Pi, \mathcal{A} \rangle$ is consistent if it has a belief set; otherwise, it is inconsistent. In what follows, we associate a name $n_r$ to each rule $r$ and freely use the name to represent the rule.\(^1\)

**Example 2.2.** Consider the program $P = (\Pi, \mathcal{A})$ where

$$\Pi = \{ n_1 : s \leftarrow, \quad n_2 : \not p, \not q \}
\mathcal{A} = \{ n_3 : p \leftarrow r, \quad n_4 : q \leftarrow \not r \}$$

We can easily check that $\Pi$ is inconsistent. $P$ has the belief sets $\{p, s\}$, $\{q, s\}$, $\{p, q, s\}$, obtained by adding $\{n_3\}$, $\{n_4\}$, and $\{n_3, n_4\}$ to $\Pi$.

When multiple sets of abducible rules can be used to generate belief sets of a program, an abductive program has different belief sets. Theoretically, this is not an issue. However, in practical applications, this might become a computational issue and/or is sometimes not intuitive. For instance, if abducible rules represent possible faulty components of a system and the program encodes the behavior of the system, then a belief set could encode a possible explanation for a faulty behavior; in this case, it is often sufficient and plausible to consider only a subset of all possible belief sets [Balduccini and Gelfond 2003], which is often referred to as the set of preferred belief sets. Similar consideration has also been made for ALP in [Sakama and Inoue 2000], which introduces a preference relation among abducibles. Given two abducibles $n_1, n_2 \in \mathcal{A}$, $\Pi$ can include atoms of the form $n_1 < n_2$ meaning that $n_2$ is preferred to $n_1$. The relation $<$ is transitive and anti-symmetric. Observe that we do not require $<$ to be a total order, because (i) it is not natural; and (ii) any partial order can be extended to be a total order. This relation can be extended to a preference relation over sets in $\mathcal{A}$. For later use, we present a general definition of a preference relation over sets of abducibles that extends a transitive and anti-symmetric relation $<$.

**Definition 2.3 (Set Preferences).** Let $S$ be a set of elements with a partial order $<$ over $S$. For $Q_1, Q_2 \subseteq S$, $Q_1$ is preferred to $Q_2$, denoted by $Q_2 \sqsubset^* Q_1$, if

- There is $n_1 \in Q_1 \setminus Q_2$ such that $n_2 < n_1$ for some $n_2 \in Q_2 \setminus Q_1$ and $n_1 \not< n_3$ for any $n_3 \in Q_2 \setminus Q_1$; and
- There exists no $p_1 \in Q_2 \setminus Q_1$ and $p_2 \in Q_1 \setminus Q_2$ such that $p_2 < p_1$.

The preferred relation $\sqsubset^*$ over $S$ focuses on the elements in the set differences between two subsets $Q_1$ and $Q_2$ of $S$. In practice, the subset relationship between $Q_1$ and $Q_2$ is often seen as a natural preferred relation between subsets of a set (e.g., prefer ‘more’ or prefer ‘less’). To take this natural preferred relation into consideration, we say that $Q_1$ is downward preferred to $Q_2$, denoted by $Q_2 \ll Q_1$, if $Q_2 \sqsubset^* Q_1$ or $Q_1 \subseteq Q_2$; and $Q_1$ is upward preferred to $Q_2$, denoted by $Q_2 \ll^* Q_1$ if $Q_2 \sqsubset^* Q_1$ or $Q_2 \subseteq Q_1$.

We can show that the relations $\ll$ and $\ll^*$, applied to the set $\mathcal{A}$ with an ordering $<$ among abducibles, are irreflexive and anti-symmetric. Irreflexivity is obvious from the definition of $\ll$ and $\ll^*$.

**Proposition 2.4.** For $Q_1, Q_2 \subseteq \mathcal{A}$, the following properties hold:

- If $Q_2 \ll Q_1$ then $Q_1 \not< Q_2$.
- If $Q_2 \ll^* Q_1$ then $Q_1 \not<^* Q_2$.

**Proof.** To prove that $\ll$ is anti-symmetric, observe that $Q_2 \ll Q_1$ implies one of two cases:

- If $Q_1 \subseteq Q_2$ then this implies that $Q_1 \not< Q_2$ via the first condition in Definition 2.3.

\(^1\)We omit the rule names when they are not needed in the discussion.
— $Q_2 \supset Q_1$, then from the second point in Def. 2.3 we have that $Q_1 \not\supset Q_2$. Furthermore, $Q_2 \not\supset Q_1$: if that was not the case, then $Q_2 \setminus Q_1 = \emptyset$, which would make it impossible to have $Q_2 \supset Q_1$.

The two cases prove the conclusion of the proposition. The proof is analogous for $<^*$. 

This preference relation allows us to compare belief sets of abductive programs.

**Definition 2.5 (Preferred Belief Set).** Let $S_1$ and $S_2$ be belief sets of $P = (\Pi, A)$. $S_1$ is preferred to $S_2$, denoted by $S_2 \ll S_1$, if $S_2|_A \ll S_1|_A$. A belief set $S$ of $(P, A)$ is most preferred if there is no belief set $S'$ of $(P, A)$ such that $S \ll S'$.

**Example 2.6.** Consider the abductive program $P = (\Pi, A)$ where

$\Pi = \{ \leftarrow \neg p, \neg q, \quad r \leftarrow, \quad n_2 < n_1 \}$

$A = \{ n_1 : p \leftarrow, \quad n_2 : q \leftarrow \neg p \}$

We can check that $\Pi$ is inconsistent but $P$ has two belief sets $S_1 = \{p, r, n_2 < n_1\}$ and $S_2 = \{q, r, n_2 < n_1\}$, obtained by adding $\{n_1\}$ and $\{n_2\}$ to $\Pi$, respectively. $S_1$ is preferred to $S_2$ and $S_1$ is the most preferred belief set of $(\Pi, A)$.

**Proposition 2.7.** Every consistent program $(\Pi, A)$ has at least one most preferred belief set.

**Proof.** Follows immediately from Proposition 2.4 and the fact that the set of belief sets of $(\Pi, A)$ is not empty and finite.

We note that the notion of abductive logic program with preferences used in this paper is similar to the notion of a CR-Prolog program with preferences introduced in [Balduccini and Gelfond 2003]. The key difference between these two notions lies in the restriction on minimal (with respect to set inclusion) sets of abducibles for computing belief sets of a CR-Prolog program.

### 2.2. Abductive Logic Programs with Disinformation

Dishonest agents are those who intentionally use false or inaccurate information. Let $\ell$ be a literal in a propositional language. In this paper, we consider the following two cases:

— An agent $a$, who believes that $\ell$ is true, informs another agent $b$ that $\neg \ell$ is true.

— An agent $a$, who believes neither $\ell$ nor $\neg \ell$, informs another agent $b$ that $\ell$ (or $\neg \ell$) is true.

The first case is commonly referred to as a lie [Mahon 2008], while the second case is called a bullshit (shortly, BS) [Frankfurt 2005]. In both cases, the information $\ell$ provided to the agent $b$ is false or inaccurate (in contrast to the reality as believed by the agent $a$). We call such piece of information $\ell$ a disinformation. For instance, consider a salesperson who believes that a product is of poor quality. If he/she communicates to a customer that the product has a good quality, then he/she will produce a lie. If a salesperson ignorant about the quality of a product communicates to a customer that the product has a good quality, then this will create a BS. In abductive programs, disinformation is defined as follows.

**Definition 2.8 (Disinformation).** Let $(\Pi, A)$ be an abductive program, and $L$ and $B$ be two sets of literals. $D = (L, B)$ is called disinformation w.r.t. $(\Pi, A)$ if

— if $L \neq \emptyset$ then $(\Pi, A)$ is consistent;

— $\forall \ell \in L$, $\neg \ell$ belongs to every belief set of $(\Pi, A)$;

— $\forall \ell \in B$, neither $\ell$ nor $\neg \ell$ belongs to any belief set of $(\Pi, A)$.

Literals in $L$ represent lies, as their opposite facts are included in every belief set of $(\Pi, A)$ (and, by requiring $(\Pi, A)$ being consistent, we have evidence of such). Literals in $B$ represent BS, as none of them (or their negations) are present in any belief set. This definition entails that $L \cap B = \emptyset$. We observe that the proposed definition represents a skeptical reasoner. An alternative view on lies and BS could be defined for credulous reasoners. Although this is an interesting topic, we will leave it for future investigations since it is not the focus of this paper. It is easy to see that the following holds.
Observation 1. For an inconsistent abductive program \((\Pi, A)\), any disinformation w.r.t. \((\Pi, A)\) is of the form \((\emptyset, B)\).

We next introduce a framework for realizing dishonest reasoning using abductive programs. An agent may use disinformation if he/she cannot achieve a goal without it. The situation is realized using an abductive program with disinformation as follows.

Definition 2.9 (ALD-program). An abductive logic program with disinformation (ALD-program) is a tuple \((\Pi, A, L, B)\) where \((\Pi, A)\) is an abductive program, called the base (of the ALD-program), and \(D = (L, B)\) is a disinformation w.r.t. \((\Pi, A)\).

The semantics of an ALD-program \((\Pi, A, L, B)\) will be defined by belief sets of an abductive logic program obtained from \((\Pi, A)\), taking into consideration that the reasoner, who employs \((\Pi, A, L, B)\) instead of \((\Pi, A)\), would use the disinformation \((L, B)\) to achieve his/her goal. For an abductive logic program \(P = (\Pi, A)\) and a disinformation \(D = (L, B)\), let

\[
I(P, D) = \{ r \mid r \in \Pi \text{ and } head(r) \cap L^\neq = \emptyset \},
\]

\[
\Phi(P, D) = \{ n_j < n_i \mid n_i \in I(P, D) \text{ and } n_j \in A \} 
\cup \{ n_h < n_k \mid n_k \in A \cup I(P, D) \text{ and } n_h \in (L \cup comp(B)) \} 
\cup \{ n_s < n_t \mid n_s \in L \text{ and } n_t \in comp(B) \}
\]

where \(L^\neq = \{ -\ell \mid \ell \in L \}\). We call \(P_D = (\Pi \setminus I(P, D) \cup \Phi(P, D), A \cup I(P, D) \cup L \cup comp(B))\) the inclusion of \(D\) into \(P\). We observe that \(P_D\) is obtained from \(P = (\Pi, A)\) and \(D = (L, B)\) by

(i) Removing from \(\Pi\) every rule whose head contains a literal that conflicts with lies in \(L\);

(ii) Adding to the set of abducibles all rules that have been removed from the background knowledge \(\Pi\) and the disinformation; and

(iii) Adding to its background knowledge the set \(\Phi\) that specifies preferences over abducibles:

(a) every rule, that has been removed form \(\Pi\), is preferred to abducibles in \(A\),

(b) every rule in \(A \cup I(P, D)\) is preferred to disinformation \(L \cup comp(B)\), and

(c) BS is preferred to lies.

(i) and (ii) guarantee that literals in \(L^\neq\) cannot be derived using the new background knowledge \((\Pi \setminus I(P, D) \cup \Phi(P, D))\) thus allowing literals in \(L\) to be used as abducibles if necessary. Additionally, (ii) also allows for literals in \(comp(B)\) to be used as abducibles, making them grounded whenever it is necessary. (iii) represents the fact that we assume that agents try to be honest as much as possible. Indeed, (a) states that every rule from the background knowledge \(\Pi\) is preferred to hypotheses in \(A\); (b) reflects our assumption that agents try to be honest as much as possible, and they prefer rules from the original program than lies or BS; (c) states that lies are considered more harmful than BS, since lies are wrong beliefs while BS are ungrounded beliefs. Observe that the inclusion of \(D = (\emptyset, \emptyset)\) into an abductive program \((\Pi, A)\) leads to the original abductive program \((\Pi, A)\).

Definition 2.10 (Belief Sets of ALD-program). Let \((\Pi, A, L, B)\) be an abductive program with disinformation and \(P_D\) be the inclusion of \(D = (L, B)\) into \(P = (\Pi, A)\). A set of literals \(S\) is a belief set of \((\Pi, A, L, B)\) if \(S\) is a belief set of \(P_D\).

The preference relation between belief sets of an ALD-program \((\Pi, A, L, B)\) is defined next.

Definition 2.11. Let \(S_1\) and \(S_2\) be two belief sets of an ALD-program \(P = (\Pi, A, L, B)\). We say that \(S_1\) is preferred to \(S_2\), denoted by \(S_1 \ll S_2\), if

- \(S_1 \cap L \subseteq S_2 \cap L\); or
- \(S_1 \cap L = S_2 \cap L\) and \(S_1 \cap \text{comp}(B) \subseteq S_2 \cap \text{comp}(B)\); or

ACM Transactions on Computational Logic, Vol. V, No. N, Article A, Publication date: YYYY.
A belief set $S$ of $P$ is most preferred if there is no belief set $S'$ of $P$ such that $S \ll S'$.

It is easy to see that the above notion of preferred belief sets favors honesty:

**Proposition 2.12.** Let $P = \langle \Pi, A, L, B \rangle$ be an ALD-program. Assume that $S_1$ and $S_2$ are two belief sets of $P$ such that $S_1 \cap (L \cup \text{comp}(B)) = \emptyset$ and $S_2 \cap (L \cup \text{comp}(B)) \neq \emptyset$. Then, $S_2 \ll S_1$.

**Proof.** $S_1 \cap (L \cup \text{comp}(B)) = \emptyset$ implies that $S_1 \cap L = \emptyset$ and $S_2 \cap \text{comp}(B) = \emptyset$. $S_2 \cap (L \cup \text{comp}(B)) \neq \emptyset$ implies that $S_2 \cap L \neq \emptyset$ or $S_2 \cap \text{comp}(B) \neq \emptyset$. The conclusion of the proposition follows from either the first or the second item of Definition 2.11. $\square$

In the absence of disinformation, Definition 2.11 reduces to the notion of preferred belief sets discussed in Section 2.1.

**Proposition 2.13.** Let $Q = \langle \Pi, A, L, B \rangle$ be an ALD-program and $S$ be a most preferred belief set of $P = \langle \Pi, A \rangle$. Then, $S \cup \Phi(P, \mathcal{D})$, where $\mathcal{D} = (L, B)$, is a most preferred belief set of $Q$.

**Proof.** Let $P_\mathcal{D} = \langle \Pi_1, A_1 \rangle$ be the inclusion of $\mathcal{D}$ into $P$, where $\Pi_1 = \Pi \setminus I(P, \mathcal{D}) \cup \Phi(P, \mathcal{D})$ and $A_1 = A \cup \Phi(P, \mathcal{D}) \cup L \cup \text{comp}(B)$. Since $S$ is a belief set of $P$, we have that $S$ is an answer set of $\Pi_1 \cup S|_A$ where $S|_A$ is the set of applicable rules in $A$ whose head contains elements in $S$ (see Eq. (2)). Because no element of $\Phi(P, \mathcal{D})$ appears in any rule in $\Pi_1 \cup A$, we have that $S \cup \Phi(P, \mathcal{D})$ is an answer set of $\Pi_1 \cup \Phi(P, \mathcal{D}) \cup S|_A$. This implies that $S \cup \Phi(P, \mathcal{D})$ is a belief set of $\langle \Pi_1, A_1 \rangle$, i.e., $S \cup \Phi(P, \mathcal{D})$ is a belief set of $Q$.

Assume that $S \cup \Phi(P, \mathcal{D})$ is a most preferred belief set of $Q$. By Definition 2.11, there exists some belief set $S'$ of $Q$ such that $S \cup \Phi(P, \mathcal{D}) \ll S'$. We will first show that $S' \setminus \Phi(P, \mathcal{D})$ is also a belief set of $P$. Since $S$ is a belief set of $P$, we have that $S \cap (L \cup \text{comp}(B)) = \emptyset$. Because $S \ll S'$, Definition 2.11 implies that $S' \cap (L \cup \text{comp}(B)) = \emptyset$. $S'$ is a belief set of $Q$ implies that $S'$ is an answer set of $\Pi_1 \cup S'|_A$. Because $S' \cap (L \cup \text{comp}(B)) = \emptyset$, we have that $S'|_A \subseteq A \cup I(P, \mathcal{D})$. This allows us to conclude that $S' \setminus \Phi(P, \mathcal{D})$ is a belief set of $P$. From the definition of preferred belief sets, we can conclude that $S \ll S' \setminus \Phi(P, \mathcal{D})$ which contradicts the fact that $S$ is a most preferred belief set of $P$, i.e., we have proved that $S \cup \Phi(P, \mathcal{D})$ is a most preferred belief set of $Q$. $\square$

It follows from the proposition above that if $\langle \Pi, A \rangle$ is consistent then the program $P$ possesses a most preferred belief set without disinformation. In other words, when the base of an ALD-program is consistent, disinformation could become useless. On the other hand, it is important to observe that an ALD-program might be consistent even when its base is inconsistent, i.e., disinformation could be used to restore the consistency of an abductive program.

**Example 2.14.** Consider the abductive program $\langle \Pi, A \rangle$ where

\[ \Pi = \{ \leftarrow \neg q, \quad q \leftarrow p, r, \quad \neg p \leftarrow \} \]

$\langle \Pi, A \rangle$ is inconsistent and $\mathcal{D} = (\emptyset, \{ q \})$ is a disinformation w.r.t. $\langle \Pi, A \rangle$. The inclusion of $\mathcal{D}$ into $\langle \Pi, A \rangle$ is $\langle \Pi_\mathcal{D}, A_{\mathcal{D}} \rangle$ where

\[ \Pi_\mathcal{D} = \{ \leftarrow \neg q, \quad q \leftarrow p, r, \quad \neg p \leftarrow, \quad n_2 < n_1 \quad n_3 < n_1 \} \]

and $A_{\mathcal{D}} = \{ n_1 : r, \quad n_2 : q, \quad n_3 : \neg q \}$. The program $\langle \Pi_\mathcal{D}, A_{\mathcal{D}} \rangle$ has two belief sets $\{ \neg p, q, n_2 < n_1, n_3 < n_1 \}$ and $\{ \neg p, r, n_2 < n_1, n_3 < n_1 \}$, i.e., the ALD-program $\langle \Pi, A, \emptyset, \{ q \} \rangle$ is consistent.

### 3. KNOWLEDGE BASES AND PROPOSALS

This section develops the notion of a negotiation knowledge base for negotiation among dishonest agents, building on the fundamentals of ALD programs.

---

2 We identify preference relations among belief sets of ALP programs and ALD-programs with the same notation $\ll$, with a slight abuse of notation—since we view the relation for ALD-programs to be an extension of the corresponding relation for ALP programs.
3.1. Negotiation Knowledge Bases

A knowledge base for negotiation serves as a means for an agent to create his/her proposals/responses during a negotiation exchange, and to decide whether he/she should accept/reject a proposal. To this end, it must encode an agent’s beliefs, rules for negotiation with their preferences, possible assumptions about the other agent, and the willingness to behave dishonestly with regards to certain items of knowledge. We model these features using ALD programs.

Definition 3.1 (Negotiation KB). A negotiation knowledge base (NKB) is defined as a tuple $K = (P, H, N^K)$ such that:

- $P = \langle \Pi, A, L, B \rangle$ is an ALD-program.
- $H$ is a set of literals (called assumptions) such that $\text{comp}(H) \subseteq \text{head}(A)$ where $\text{head}(A)$ denotes the set $\{\ell \mid \ell \in \text{head}(r) \text{ for } r \in A\}$.
- $N^K$ is a set of literals (called negotiated conditions) associated with a strict partial order $\prec$ on its elements.

A belief set of $P$ is said to be a belief set of the NKB $K$. A NKB is consistent if $P$ is consistent.

$P$ represents an agent’s domain-specific knowledge, goals and preferences with disinformation. $H$ represents unknown information about the other agent—and the agent will be able to make assumptions about this in the course of a negotiation. $N^K$ specifies desired outcomes in a negotiation. $p \prec q$ means that $q$ is preferred to $p$. For simplicity, we often write $N^K = \{p \prec q \prec r\}$ if $p \prec q$ and $q \prec r$ hold over the set $N^K = \{p, q, r\}$. Since an ALD-program serves as a means for an agent to generate arguments in negotiation, it is assumed to be consistent.

We now present two NKBs, one for a seller and one for a buyer, describing the agents addressed in the introduction. Intuitively, the seller agent will have a NKB for him/her to negotiate a sale, while the buyer agent will have a NKB for him/her to negotiate a purchase. The buyer wants to get a product for a low price, while the seller wants to sell a product for a high price.

Example 3.2. The seller agent $s$ in Example 1.1 might use the following NKB $K_s = (P_s, H_s, N^K_s)$ where:

- $P_s = \langle \Pi_s, A_s, L_s, B_s \rangle$ with $(L_s, B_s) = (\{\text{qual}_B\}, \{\text{qual}_A\})$ and

$\Pi_s = \{\begin{align*}
\text{sale}_{A} & \leftarrow \text{prod}_A, \text{price}_1 \\
\text{sale}_{B} & \leftarrow \text{prod}_B, \text{price}_2 \\
\text{not} \text{sale}_{A}, \text{not} \text{sale}_{B} & \\
\text{mak}_{C} & \leftarrow \text{prod}_A \\
\text{mak}_{D} & \leftarrow \text{prod}_B \\
\text{qual}_{B} & \leftarrow \text{prod}_B \\
\text{bargain}_{B} & \\
\text{prod}_A & \\
\text{prod}_B & \\
n_i < n_1 & \\
\text{high, low} & \\
\text{high, lowest} & \\
\text{low, lowest} & \\
\text{price}_1 & \in \{\text{high, low}\} \\
\text{price}_2 & \in \{\text{low, lowest, high}\} \\
\end{align*}\}$

3. We distinguish $\prec$ from $<$ that is defined over abducibles.
The next example details the NKB of the buyer.

Example 3.3. The buyer agent $b$ in Example 1.1 can use the NKB $K_b = (P_b, H_b, N_b)$ where:

- $P_b = \langle \Pi_b, \mathcal{A}_b, L_b, B_b \rangle$ with $(L_b, B_b) = (\emptyset, \{\text{mailing}\})$ and

$$\Pi_b = \{\begin{array}{l}
\text{purchase}_x \leftarrow \text{prod}_x, \text{qual}_x, \text{price}_3 \\
\text{not purchase}_A, \text{not purchase}_B \\
\text{not student} \\
\text{cash} \leftarrow \text{low, lowest} \\
n_2 < n_1, n_3 < n_2, n_3 < n_1
\end{array}\}$$

where $x \in \{A, B\}$ and $\text{price}_3 \in \{\text{low, lowest}\}$ and

$$\mathcal{A}_b = \{\begin{array}{l}
n_1 : \text{lowest} \leftarrow \text{mak}_C \\
n_2 : \text{lowest} \leftarrow \text{mak}_D \\
n_3 : \text{low} \leftarrow \text{mak}_C \\
n_4 : \text{qual}_A \\
n_4^{-} : \neg \text{qual}_A \\
n_5 : \text{qual}_B \\
n_5^{-} : \neg \text{qual}_B \\
n_6 : \text{mak}_C \\
n_6^{-} : \neg \text{mak}_C \\
n_7 : \text{mak}_D \\
n_7^{-} : \neg \text{mak}_D \\
n_8 : \text{prod}_A \\
n_8^{-} : \neg \text{prod}_A \\
n_9 : \text{prod}_B \\
n_9^{-} : \neg \text{prod}_B
\end{array}\}$$

$\Pi_b$ encodes the fact that the buyer has a goal of making a purchase which can be satisfied by a product $A$ or $B$ of good quality. He/she is not a student and will pay by cash. Also, he/she is willing to pay...
either low or lowest price for the product. \( A_b \) encodes that the buyer is willing to pay a higher price for products made by the maker \( C \)—though the preference \( n_3 < n_1 \) indicates that he/she prefers to pay the lowest price. \( \Pi_b \) also contains preferences over the abducibles. Before the negotiation, the buyer does not know whether the products that he/she would like to buy are available. The buyer does not care about the mailing list of the seller but could pretend to join it if it works to his/her advantage. This is the reason the buyer uses the disinformation \( D_b = (\emptyset, \{ \text{mailing} \}) \).

- \( H_b = \{ \text{qual}_A, \text{qual}_B, \text{mak}_C, \text{mak}_D, \text{prod}_A, \text{prod}_B \} \) which represents properties of products that the buyer needs to check.
- \( N_b^\subset = \{ \text{low} < \text{lowest} \} \) indicates that the buyer prefers to pay the lowest price.

It is easy to check that \( K_b \) is also consistent.

### 3.2. Proposals and Acceptability

We will now define the notion of a \((\text{negotiation})\) proposal. In building a proposal, an agent has a goal and can make assumptions about the receiver of the proposal or about unknown propositions related to the agents’ goal. The agent may also decide to reveal information about his/her state-of-belief, rendering some conditions on the feasibility of the proposal. Given a set \( S \) of literals, let \( \text{Goal}(S) = \{ \leftarrow \not\ell \mid \ell \in S \} \). Observe \( \not\ell \) encodes the constraint that, when added to a program, will generate belief sets in which \( \ell \) must be true, i.e., it encodes the goal of the agent to negotiate for \( \ell \). For an ALD-program \( P = (\Pi, A, L, B) \), \( P \cup X \) denotes the program \( P = (\Pi \cup X, A, L, B) \).

**Definition 3.4 (Proposal).** Let \( K = (P, H, N^\subset) \) be the NKB of an agent with \( P = (\Pi, A, L, B) \). For a set of literals \( G \subseteq N^\subset \), a tuple \( \gamma = (G, S, R) \) is a proposal w.r.t. \( K \) if \( P \cup \text{Goal}(G) \) has a belief set \( M \) such that \( S = M \cap \text{comp}(H) \) and \( R \subseteq M \setminus \text{comp}(H) \). \( G, S, R, \) and \( M \) are referred to as the \( \text{goal}, \text{assumptions}, \text{conditions}, \) and \( \text{support of} \) \( \gamma \), respectively. In particular:

- \( \gamma \) is \textit{honest} if \( M \cap (L \cup \text{comp}(B)) = \emptyset \);
- \( \gamma \) is \textit{deceptive} if \( M \cap L \neq \emptyset \); and
- \( \gamma \) is \textit{unreliable} if \( M \cap \text{comp}(B) \neq \emptyset \).

\( \alpha(K, G) \) denotes the set of all possible proposals for \( G \) w.r.t. \( K \).

Intuitively, a proposal \( \gamma = (G, S, R) \) denotes the agent’s attempt to negotiate to achieve the objective \( G \). The reason to put forward \( \gamma \) is that the agent has a support \( M \) for it. The agent indicates the assumptions \( S \) that he/she has made about the receiver of \( \gamma \). In addition, the agent also reveals additional information \( R \) supporting the goal \( G \), which informs the receiver that the information in \( R \) should not be violated.

**Example 3.5.** Consider the NKB \( K_s \) in Example 3.2; then

\[
(\{\text{high}\}, \emptyset, \{\text{prod}_A\}) \quad \text{and} \quad (\{\text{low}\}, \{\text{student}\}, \{\text{prod}_B\})
\]

are two honest proposals by the seller. The former states that the seller can sell \( \text{prod}_A \) for the price \( \text{high} \). The latter states that the seller can sell \( \text{prod}_B \) for the price \( \text{low} \) if the customer is a student.

A deceptive proposal by the seller is

\[
(\{\text{low}\}, \{\text{student}\}, \{\text{prod}_B, \text{qual}_B\})
\]

indicating that the seller can sell \( \text{prod}_B \), that has good quality, for \( \text{low} \) price, provided that the buyer is a student. The proposal is deceptive because \( \text{qual}_B \) is a lie. Similarly,

\[
(\{\text{high}\}, \emptyset, \{\text{prod}_A, \text{qual}_A\})
\]

is an unreliable proposal w.r.t. \( K_s \).

For the NKB \( K_b \) in Example 3.3, we have that:

\[
(\{\text{low}\}, \{\text{prod}_A, \text{mak}_C, \text{qual}_A\}, \emptyset) \quad \text{and} \quad (\{\text{lowest}\}, \{\text{prod}_B, \text{mak}_D, \text{qual}_B\}, \{\neg\text{student}\})
\]
are two possible proposals by the buyer. The first states that the buyer is willing to buy \( \text{prod}_A \) with the low price, assuming that it is made by the maker \( C \) and it is of good quality; the second is similar, for buying \( \text{prod}_B \) with the lowest price, assuming that it is made by the maker \( D \) and it is of good quality. In addition, the buyer indicates that he/she is not a student in the second proposal. These proposals by the buyer are honest.

Given an agent \( a \) and a proposal \( \gamma_b = (G, S, R) \) from another agent \( b \), we can have one of the following three cases:

(i) \( a \) accepts \( \gamma_b \): this means that the goal of \( \gamma_b \) matches the goal of \( a \), or it is at least as good as what \( a \) wants.

The assumptions stated in \( S \) are acceptable to \( a \) and are consistent with his/her knowledge. Furthermore, \( a \) can accept the conditions stated in \( R \).

(ii) \( a \) rejects \( \gamma_b \): this means that \( a \) cannot accept the proposal \( \gamma_b \).

(iii) \( a \) sees some alternative proposals for the goal of \( \gamma_b \), yet \( \gamma_b \) is not suitable for \( a \), i.e., \( a \) considers \( \gamma_b \) a negotiable proposal.

Next we consider the acceptability of proposals.

**Definition 3.6 (Acceptability).** Let \( K_a = (P, H, N^a) \) be the NKB of agent \( a \), with \( P = (\Pi, A, L, B) \), and \( \gamma_b = (G, S, R) \) a proposal from another agent \( b \). Then,

\[ \gamma_b \text{ is acceptable w.r.t. } K_a \text{ if } Q = P \cup \text{Goal}(G) \text{ has a belief set } M \text{ such that } S \subseteq M \text{ and } M \cap \text{comp}(H) \subseteq R \cap \text{comp}(H). \text{ We say that } \gamma_b \text{ is acceptable without disinformation if } M \cap (L \cup \text{comp}(B)) = \emptyset, \text{ with disinformation, otherwise.} \]

\[ \gamma_b \text{ is rejectable if } Q \text{ is inconsistent.} \]

\[ \gamma_b \text{ is negotiable, otherwise.} \]

Intuitively, \( Q \) encodes the possibility of satisfying \( b \)'s goal \( G \) in the ALD-program \( P \) of the agent \( a \). If \( Q \) is inconsistent then \( \gamma_b \) is rejectable. \( \gamma_b \) is acceptable if \( Q \) has a belief set \( M \) such that: (i) \( M \) is compatible with the assumptions \( S \) about the agent \( a \) (\( S \subseteq M \)); and (ii) if there are assumptions made by the receiving agent \( a \) about the proposer \( b \), then these must be compatible with the information revealed by the proposer (i.e., \( M \cap \text{comp}(H) \subseteq R \cap \text{comp}(H) \)). The first condition is needed, since a negotiated goal is acceptable to both parties only if their supports agree. The second condition implies that a proposal is based on the same set of shared assumptions. A proposal is negotiable if it is neither acceptable nor rejectable. Note that when an agent considers a proposal acceptable or negotiable, he/she may use disinformation included in his/her NKB.

**Example 3.7.** For \( K_s \) and \( K_b \) from Examples 3.2 and 3.3, we can check the following:

- The proposal \( \{\text{high}\}, \{\text{prod}_A, \text{qual}_A\}, \emptyset \) is acceptable with disinformation w.r.t. \( K_s \), since \( P_s \cup \text{Goal}([\text{high}]) \) has a belief set \( M \) containing \text{high}, \text{prod}_A, and \text{qual}_A. Any belief set that allows the seller to accept this proposal contains the BS \text{qual}_A.

- The proposal \( \{\text{low}\}, \{\text{prod}_A, \text{mak}_D, \text{qual}_B\}, \emptyset \) is a negotiable proposal w.r.t. \( K_s \), since \( P_s \cup \text{Goal}([\text{high}]) \) has a belief set containing its assumptions but requires at least one of the sets \{\text{student}\} or \{\text{cash}\}.

- The proposal \( \{\text{high}\}, \emptyset, \{\text{prod}_A, \text{mak}_C, \text{qual}_A\} \) is a rejectable proposal w.r.t. \( K_b \) because \( P_b \cup \text{Goal}([\text{high}]) \) has no belief set containing \text{high}.

An agent can employ a stronger condition for accepting a proposal than what used in Definition 3.6, e.g., by restricting the belief set \( M \) to be \( M \cap (L \cup \text{comp}(B)) = \emptyset. \) This may lead an agent to consider \( \gamma \) acceptable only if it is acceptable without disinformation. With this stronger condition, the proposal \( \{\text{high}\}, \{\text{prod}_A, \text{mak}_C, \text{qual}_A\}, \emptyset \) is unacceptable without disinformation but negotiable for the seller if he/she wants to be honest. On the other hand, it is acceptable if the seller uses disinformation. We will return to this issue in the next section.
Proposition 3.8. For any NKB $K$ and a proposal $\gamma$ posted to $K$, then $\gamma$ is either acceptable, negotiable, or rejectable with respect to $K$.

Proof. This proposition follows immediately from the definition of acceptability. \qed

4. Negotiation Using NKBS

We will now present a model of negotiation between two agents $a$ and $b$ who use NKBS $K_a$ and $K_b$ respectively. We assume that $K_a$ and $K_b$ share the same language, and the set $H$ of assumptions in $K_a$ is disjoint from the one in $K_b$.

4.1. Negotiation

We envision a negotiation to be composed of several rounds. In each round (see Figure 1), one of the agents, say $x (x \in \{A, B\})$, puts forward a proposal $\gamma_x = (G, S, R)$, that includes the goal, the assumptions that $x$ made about his opponent $y \in \{A, B\} \setminus \{x\}$ ($Assumptions_{x>y}$), and the information about his/herself ($Information_x$). The opponent $y$ will respond with a proposal with the same structure.

**Fig. 1.** Proposal and Response

We will begin with the definition of a response to a proposal. In this subsection, a response could be an arbitrary proposal, an acceptance or a rejection of the current proposal.

Definition 4.1 (Response). Let $K_a = (P, H, N^a)$ be a NKB of agent $a$, and $\gamma_b = (G, S, R)$ a proposal by $b$ w.r.t. its NKB $K_b$. A response to $\gamma_b$ by $a$ is either

- a proposal $\gamma_a = (G', S', R')$ w.r.t. $K_a$; or
- $\langle \top, \emptyset, \emptyset \rangle$, denoting acceptance of the proposal, if $\gamma_b$ is acceptable w.r.t. $K_a$; or
- $\langle \bot, \emptyset, \emptyset \rangle$, denoting rejection of the proposal.

The set of all responses (by $a$ w.r.t. $K_a$) to a proposal $\gamma_b$ is denoted by $\beta(K_a, \gamma_b)$.

A negotiation is a series of responses between two agents, who, in alternation, take into consideration the other agent’s response and put forward a new response; this can be either accept, reject, or a new proposal that may involve explanations of why the latest proposal (of the other agent) was not acceptable. A possible infinite sequence of responses $\omega_1, \ldots, \omega_i, \ldots$ is denoted by $\langle \omega_i \rangle_{i>0}$.

Definition 4.2 (Negotiation). Let $a$ and $b$ be two agents, and $K_a$ and $K_b$ be their NKBS respectively. A negotiation between two agents $a$ and $b$, starting with $a$, is a possible infinite sequence of responses $\langle \omega_i \rangle_{i>0}$, where $\omega_i = (G_i, S_i, R_i)$ and

- $\omega_{2k+1}$ is a proposal w.r.t. $K_a$ ($k \geq 0$) and is a response to $\omega_{2k}$ by $a$ for $k \geq 1$; and
- $\omega_{2k}$ is a proposal w.r.t. $K_b$ and is a response to $\omega_{2k-1}$ by $b$ for $k \geq 1$.
A negotiation ends at $i$ if $\omega_i = (\top, \emptyset, \emptyset)$ or $\omega_i = (\bot, \emptyset, \emptyset)$. A negotiation contains a goal change if $G_i \neq G_{i+2}$ for some $i$. In this case, we say that the agent, who responded with $\omega_i$, changes the goal of the negotiation.

Observe that the definition of a response indicates that a negotiation can end with $(\top, \emptyset, \emptyset)$ only if the preceding proposal is acceptable with the agent who utters the acceptance. We classify negotiations as follows.

**Definition 4.3 (Successful and Unsuccessful Negotiations).** A negotiation is successful (resp. unsuccessful) if it is finite and ends with $\omega_i = (\top, \emptyset, \emptyset)$ (resp. $\omega_i = (\bot, \emptyset, \emptyset)$). We call $\omega_{i-1}$ the accepted (resp. rejected) proposal of the negotiation.

**Example 4.4.** The negotiation dialogue between the seller $s$ and the buyer $b$ in the introduction is realized using NKBs of Examples 3.2 and 3.3 as follows:

- $b_1: (\{\text{low}\}, \{\text{prod}_A, \text{qual}_A, \text{mak}_C\}, \emptyset)$
- $s_1: (\{\text{low}\}, \{\text{student}\}, \{\text{prod}_A, \text{qual}_A, \text{mak}_C\})$
- $b_2: (\{\text{low}\}, \{\text{prod}_A, \text{qual}_A, \text{mak}_C\}, \{\neg \text{student}\})$
- $s_2: (\{\text{low}\}, \{\text{cash}\}, \{\text{prod}_B, \text{mak}_D, \text{qual}_B\})$
- $b_3: (\{\text{lowest}\}, \{\text{prod}_B, \text{mak}_D, \text{qual}_B\}, \{\text{cash}\})$
- $s_3: (\{\text{lowest}\}, \{\text{cash, mailing}\}, \{\text{prod}_B, \text{mak}_D, \text{qual}_B\})$
- $b_4: (\top, \emptyset, \emptyset)$

In this negotiation, the seller uses a BS in $s_1$, by indicating that $\text{qual}_A$ is true, and lies in $s_2$ by claiming that $\text{qual}_B$ is true. The buyer uses a BS in $b_4$, because he accepts it under the assumption that $\text{mailing}$ is true. A goal change has occurred at $b_3$ (for the buyer) and $s_3$ (for the seller).

A negotiation (Definition 4.2) can be infinite. In real-life, however, a negotiation always terminates. We next characterize a termination condition of negotiation. We need the following definition.

**Definition 4.5 (Constructive Negotiation).** A negotiation $\langle \omega_i \rangle_{i \geq 0}$ is constructive if for every pair of indices $1 \leq i < j$, if $\omega_i = \omega_j$ then $\omega_{i+1} \neq \omega_{j+1}$.

In a constructive negotiation, agents do not repeat the same response to the same proposal, which leads to finiteness (because of the finiteness of the underlying propositional language).

**Theorem 4.6.** Every constructive negotiation is finite.

**Proof.** The structure of a proposal or a response includes three components (goals $G$, assumptions $S$, and information $R$) and each is a subset of a finite set, due to the fact that a NKB is finite. Therefore, there can only be a finite number of different proposals/responses generated by the NKBs of the two negotiating agents. Since agents do not repeat the same response to the same proposal in a constructive negotiation, and the number of different proposals/responses is finite, we can conclude that every constructive negotiation is finite.

A negotiation represents one possible way for two agents to attempt to reach an agreement. In the course of reaching an agreement, two agents might have different alternatives. We will represent all of the possible negotiations between two agents using a negotiation tree.

**Definition 4.7 (Negotiation Tree).** Let $a$ and $b$ be two agents with the NKBs $K_a$ and $K_b$, respectively. A negotiation tree between agents $a$ and $b$, starting with $a$, is a labeled tree $T_{a,b}$ where

- The root of $T_{a,b}$ is $\Box$;
- Each child of the root node has a label of the form $(K_a, \gamma_a, K_b)$ where $\gamma_a$ is a proposal w.r.t. $K_a$;
- If $\eta_i = (K, \gamma, K')$ is a node at level $i \geq 1$ in the tree, then every child of $\eta_i$ has a label of the form $(K', \gamma', K)$ where $\gamma' \in \beta(K', \gamma)$; and
- Nodes labeled by $(K, (\top, \emptyset, \emptyset), K')$ or $(K, (\bot, \emptyset, \emptyset), K')$ are leaves and have no children.
Observe that a negotiation tree is analogous to the notion of a protocol as used in [Kakas and Moraitis 2006]. Given a negotiation tree $T_{a,b}$, it is easy to see that each path from a node $n_i$ of $T_{a,b}$ to a leaf encodes a negotiation between $a$ and $b$. Since a negotiation tree represents all possible negotiations between $a$ and $b$, starting with $a$, it can be used to predict the possible results of a negotiation between the two agents. If $T_{a,b}$ contains a leaf labeled $(K, (\top, \emptyset, \emptyset), K')$, there is a possibility of reaching an agreement; otherwise, no agreement can be reached. We classify negotiation trees as follows.

**Definition 4.8 (Classification of Negotiation Trees).** A negotiation tree is *finite* if it has a finite number of nodes. A finite tree is *successful* if it has a leaf whose label is of the form $(K, (\top, \emptyset, \emptyset), K')$; it is *unsuccessful* if all of its leaves have a label of the form $(K, (\bot, \emptyset, \emptyset), K')$.

The idea of constructive negotiation is extended to negotiation trees.

**Theorem 4.9 (Constructive Negotiation Tree).** A negotiation tree is *constructive* if every path of the tree encodes a constructive negotiation.

**Proof.** This theorem follows from Theorem 4.6 and the definition of a negotiation tree. \qed

### 4.2. Negotiation Strategies

The previous section provides the basic definitions for modeling negotiation. In practice, agents engaging in a negotiation commonly employ strategies in determining which branch of the negotiation tree to follow. We next formalize this notion and use it to classify agents. We will begin with a generic notion of *strategy*. In this subsection, by $(\omega)^1$ we denote the finite negotiation $\omega_1, \ldots, \omega_i$ with the final proposal $\omega_i$. Furthermore, $(\omega)^1 \circ \omega$ is the result of the concatenation of $\omega$ to $(\omega)^1$, i.e., $\omega_1, \ldots, \omega_i, \omega$.

**Definition 4.11 (Strategy).** Given an agent $a$ with a NKB $K$, a *negotiation strategy* for $a$ is a function $F$ that maps a finite negotiation $(\omega)^1$, whose last proposal $\omega_i$ is not from $a$, to a proposal that satisfies the following properties:

1. $F((\omega)^1) \in \beta(K, \omega_i)$ (i.e., the strategy selects a response to $\omega_i$), and
2. $(\omega)^1 \circ F((\omega)^1)$ is a negotiation.

The definition states that the strategy should provide an agent with a response, given a proposal and a negotiation. Given two agents $a$ and $b$ that use the strategies $F_a$ and $F_b$, respectively, the outcome of a negotiation between them now depends on $F_a$ and $F_b$. We characterize this in the next definition.

**Definition 4.12 (Negotiation with Strategy).** Let $a$ and $b$ be two agents using strategies $F_a$ and $F_b$, respectively. A negotiation $(\omega_i)_{i>0}$ between $a$ and $b$, started by $a$, is said to be $(F_a, F_b)$-*induced* if:

- $\omega_1$ is a proposal w.r.t. $K_a$ and
- for every $i > 0$, $\omega_{2i+1} = F_a((\omega)^{2i}_1)$ and $\omega_{2i} = F_b((\omega)^{2i-1}_1)$.

Agents are interested in different types of strategies. For example, strategies that guarantee the termination of a negotiation, strategies that do not use disinformation, strategies that guarantee the success of a negotiation, etc. In the following, we will discuss some possible strategies. For two finite negotiations $(\omega_i)$ and $(\omega_j)$, we write $(\omega_i) \sim (\omega_j)$ if $\omega_i$ is a proper prefix of $\omega_j$.

**Definition 4.13 (Observant Strategy).** A strategy $F$ is *observant* if $F((\omega)^1) \neq F((\omega)^1)$ for every pair of negotiations $(\omega)^1$ and $(\omega)^1_j$ such that $(\omega)^1 \sim (\omega)^1_j$.

The observant strategy states that an agent cannot repeat the same response to the same proposal within the same negotiation. If at least one agent’s strategy is observant, then the negotiation will terminate. This is because an agent does not have infinitely many responses and thus the negotiation will eventually fail/succeed.
Proposition 4.14. Let us consider two agents a and b with strategies $F_a$ and $F_b$, respectively. If either $F_a$ or $F_b$ is observant then every negotiation between a and b terminates.

Proof. Similarly to the proof of Theorem 4.6, we can observe that the agent using the observant strategy does not repeat the same response to the same proposal, and the number of different proposals/responses is finite; thus, this agent will eventual need to accept or reject the proposal, leading to the termination of the negotiation. □

So far, our discussion focused on the development of a general framework for negotiation and does not distinguish the type of information that an agent uses in achieving his/her goals. In practice, an agent might prefer to be honest before he/she uses disinformation in achieving his/her goals. Other agents might disregard 'ethical issues' of using disinformation and would prefer to put forward one of his/her proposal for a best possible goal. Some agents might insist on getting the best goal and keep repeating the same proposal until the other agent either rejects or accepts it. We will next employ strategies in characterizing agents.

We will begin with an extension of the preference relation between belief sets of an ALD-program (Subsection 2.2), to define a preference relation among belief sets of NKBs that takes into consideration the ordering between goals in the NKBs. Given a NKB $K = (P, H, N^\prec)$, let $\prec^*$ denote the upward preference relation over subsets of goals. Similarly to Proposition 2.4, we can show that $\prec^*$ is anti-symmetric. When two sets of goals $X, Y \subseteq N^\prec$ are incompatible with respect to $\prec^*$, we write $X \approx Y$. We define two preference relations over belief sets of a NKBs. Let us recall that Definition 2.11 provides a preference relation among belief sets that values the lack of disinformation.

Definition 4.15. Let $K = (P, H, N^\prec)$ be a NKB and $S_1$ and $S_2$ be two belief sets of $K$.

- $S_1$ is preferred to $S_2$ with respect to goals in $K$, denoted by $S_2 \ll^g S_1$, if
  - $S_2 \cap N^\prec \prec^* S_1 \cap N^\prec$; or
  - $S_2 \cap N^\prec \approx S_1 \cap N^\prec$ and $S_2 \ll S_1$ according to Definition 2.11.
- $S_1$ is preferred to $S_2$ with respect to disinformation in $K$, denoted by $S_2 \ll^h S_1$, if
  - $S_2 \ll S_1$ with respect to Definition 2.11; or
  - $S_1 \not\ll S_2$ and $S_2 \not\ll S_1$ with respect to Definition 2.11 and $S_2 \cap N^\prec \prec^* S_1 \cap N^\prec$.

The $\ll^g$ relation indicates that belief sets containing the best possible goals are preferred and there is no distinction between disinformation and true information. On the other hand, it follows from Proposition 2.12 that belief sets without disinformation are preferred according to $\ll^h$.

Observe that other ordering relations could be developed to further characterize agents. For instance, one could define different preference relations between two subsets of goals and use them in specifying $\ll^g$-based strategies. Another direction is to eliminate the distinction between the ordering between goals $\prec$ and the preference between the assumptions $\prec$. Although this is an interesting research topic, we will leave this for future studies. In the end, this change will create another preference relation over belief sets of a NKB.

For a set $\Omega$ of belief sets of a NKB $K$ and $i \in \{h, g\}$, we define the maximal elements of $\Omega$ as:

$$\max(\Omega, \ll^i) = \{S \mid S \in \Omega, \text{there is no } V \in \Omega \text{ such that } S \ll^i V\}.$$  

It is easy to see that $\max(\Omega, \ll^i)$ is the set of most preferred belief sets in $\Omega$ with respect to the ordering $\ll^i$.

We will discuss next how such preference relations can be employed by the agents. Let $K$ be a NKB and $\gamma$ be a proposal. We define

$$\Sigma(K, \gamma) = \{M \mid M \text{ supports some } \omega \in \beta(K, \gamma)\}.$$  

Definition 4.16 (Preference Based Strategy). A strategy $F$ of an agent with the NKB $K$ is a $\ll^i$-based strategy where $i \in \{g, h\}$ if for every proposal $\gamma$ and negotiation $\langle\omega\rangle^i_1$,

- If $F(\langle\omega\rangle^i_1) \neq (\top, \emptyset, \emptyset)$ and $\Sigma(K, \omega_j) \neq \emptyset$ then $F(\langle\omega\rangle^i_1)$ is supported by a belief set in
  $$\max(\Sigma(K, \omega_j), \ll^i);$$
If \( F(\omega_j^1) = (\top,\emptyset,\emptyset) \) then \( \omega_j \) is supported by a belief set in \( \max(\Sigma(K,\omega_j), \ll ^i) \);  
Otherwise \( F(\omega_j^1) = (\bot,\emptyset,\emptyset) \).

A \( \ll ^i \)-based strategy, which is also observant, is called a \( \ll ^i \)-best-practice strategy.

An agent who employs a \( \ll ^h \)-based (respectively, \( \ll ^g \)-based) strategy is called a deliberative (respectively, greedy) agents.

Observe that a \( \ll ^i \)-based strategy does not guarantee termination of a negotiation. However, it follows from Proposition 4.14 that a \( \ll ^i \)-best-practice strategy does. It is easy to see that a deliberate agent with a best-practice strategy may accept a less preferred outcome in a negotiation, even though he/she might obtain a better outcome had he/she used disinformation. Similarly, the agent may sometimes reject a proposal even though this might be negotiable and further steps might lead to a successful negotiation, had the agent lied or used a BS. This can be seen in the next example.

**Example 4.17.** Consider the negotiation in Example 4.4. A deliberate seller will respond to \( b_1 \) with  
\[
s_1' = \{ \{ \text{low} \}, \{ \text{student} \}, \{ \text{prod}_A, \text{mak}_C \} \}
\]
rather than \( s_1 \), since he/she has a belief set without disinformation (it does not contain \( \text{qual}_A \)) that supports \( s_1' \).

On the other hand, a greedy seller  will respond to \( b_1 \) with the proposal  
\[
s_1'' = \{ \{ \text{high} \}, \emptyset, \{ \text{prod}_A, \text{qual}_A \} \}
\]
since this proposal is supported by a most preferred belief set of \( K_s \) (with respect to \( \ll ^g \)).

Similarly, a deliberate buyer will not accept the proposal \( s_3 \) since there is no belief set without disinformation supporting its acceptance while there exists a response  
\[
b_4' = \{ \{ \text{lowest} \}, \{ \text{cash} \}, \{ \text{prod}_B, \text{qual}_B, \text{mak}_D \} \}
\]
which is supported by a belief set without disinformation. The response \( b_4 \) would be used, however, by a greedy buyer.

We will now prove that, when both agents employ preference-based strategies, the outcome of a successful negotiation will be preferred by both agents.

**Proposition 4.18.** Let \( a \) and \( b \) be two agents with the NKBs \( K_a \) and \( K_b \) and \( \ll ^i \)- and \( \ll ^j \)-preference based strategies, respectively, where \( i, j \in \{ h, g \} \). Furthermore, let \( w_1, \ldots, w_n, (\top,\emptyset,\emptyset) \) be a successful negotiation between \( a \) and \( b \), and let \( b \) be the agent that utters \( (\top,\emptyset,\emptyset) \). Then,

- \( w_n \) is supported by a belief set in \( \max(\Sigma(K_b, w_n), \ll ^j) \); and
- \( w_n \) is supported by a belief set in \( \max(\Sigma(K_a, w_n), \ll ^i) \).

**Proof.** Since \( b \) is the one that finalizes the negotiation, Definition 4.16 implies that \( w_n \) must be supported by \( \max(\Sigma(K_b, w_n), \ll ^j) \). Since \( a \) proposes \( w_n \), Definition 4.16 implies that \( w_n \) must be supported by \( \max(\Sigma(K_a, w_n), \ll ^i) \). \( \Box \)

The significance of the above proposition is that both agents believe that the final outcome of the negotiation is one of the best possible proposals that they can achieve. In other words, both agents are satisfied that they have achieved their objectives.

We conclude the section with the observation that a non-observant strategy might not guarantee termination of the negotiation. One disadvantage of the observant strategy is that it requires an agent to memorize the full history of the negotiation. This might not be desirable for automated agents with limited resources (e.g., time or memory). We next consider an alternative way to avoid this issue.

Observe that a proposal \( (G, S, R) \) consists of three parts, the goal \( G \), the assumptions \( S \), and the self-revelation \( R \). Among these parts, the goal \( G \) and the assumptions \( S \) could be changed from one round to
the next round due to the fact that the agent might attempt to negotiate for a more preferred goal or some assumptions might be incorrect. On the other hand, for a rational agent, the content of the condition part of the agent is releasing about him/herself should be consistent during a negotiation. For example, if the buyer already confirms that he/she is a student, it would be more natural to him/her to maintain that he/she is a student. This leads to the observation that if the agents employ a strategy that monotonically updates the condition part of their proposals, then any negotiation between them will terminate.

Let us use this intuition to introduce the notion of self-persistent strategy.

Definition 4.19 (Self-Persistent Strategy). Let \( K_a = (P_a, H_a, N^+_a) \) and \( K_b = (P_b, H_b, N^+_b) \) be the NKBs of an agent \( a \) and an agent \( b \), respectively. A self-persistent strategy \( FP \) (of \( a \) towards \( b \)) is a mapping from the set of rounds of negotiation between \( a \) and \( b \), starting with \( a \), to the set of proposals, i.e.,

\[
FP : \bigcup_{G \in \mathbb{N}^+_2} \left( \alpha(K_a, G) \times \bigcup_{\gamma \in \alpha(K_a, G)} \beta(K_b, \gamma) \right) \rightarrow \bigcup_{G \in \mathbb{N}^+_2} \alpha(K_a, G)
\]

where for each proposal \( \gamma_1 = (G_1, S_1, R_1) \) of \( a \) and response \( \gamma_2 = (G_2, S_2, R_2) \) by \( b \) to \( \gamma_1 \), we have that \( FP((\gamma_1, \gamma_2)) = \gamma \) such that:

- \( \gamma = (G, S, R) \) is a response to \( \gamma_2 \) and
- \( R_1 \subseteq R \) if \( \gamma \neq (\top, \emptyset, \emptyset) \) and \( \gamma \neq (\bot, \emptyset, \emptyset) \).

A self-persistent agent is an agent that makes use of a self-persistent strategy during negotiation. Let \( FP \) be a self-persistent strategy of an agent \( a \) with a NKB \( K_a \). \( FP \) induces the negotiation strategy \( F_{FP} \) as follows:

\[
F_{FP}(\langle \omega \rangle_1) = \begin{cases} 
\gamma' \in \beta(K_a, \omega_1) & \text{if } i = 1 \\
FP(\langle \omega_{i-1}, \omega_i \rangle) & \text{if } i > 1 
\end{cases}
\]

It is easy to see that the following lemma holds.

**Lemma 4.20.** For every self-persistent strategy \( FP \), \( F_{FP} \) is an observant strategy.

**Proof.** Consider the two negotiations \( \langle \omega_i \rangle \) and \( \langle \omega_j \rangle \) such that \( \omega_i \sim \omega_j \). Clearly, we have that \( F_{FP}(\langle \omega_i \rangle) \neq F_{FP}(\langle \omega_j \rangle) \) because of the transitivity of \( \sim \). \( \square \)

The following proposition is a consequence of the above lemma.

**Proposition 4.21.** Every negotiation between self-persistent agents terminates.

**Proof.** Trivially follows from Lemma 4.20 and Proposition 4.14. \( \square \)

Observe that a self-persistent strategy differs from an observant strategy in that it only requires the agent to “memorize” his/her last proposal, thus avoiding the possible issue of limited resource of agents.

### 4.3. Adaptive Negotiations

In this section, we define the notion of an adaptive negotiation that provides the theoretical foundation for the development of the prototypical negotiated agents presented in the next section. We start with a discussion on what should a reasonable response between rational agents contain. Let us consider a rational agent who receives a proposal and needs to develop his response. Being rational, the agent should consider

(i) The assumptions (about him/her) that have been made by the opponent;
(ii) The information that the opponent reveals about him/herself.

With regards to (i), the response should identify the false assumptions and inform his/her opponent in the response. The agent can, of course, lie about it if he/she decides to do so. For instance, if the seller assumes that the buyer is a student but the buyer is not, then the buyer should identify this and inform the seller, as far as if he/she does not disguise him/herself as a student. In Example 4.4, the buyer responds to \( S_1 \) by...
informing the seller that he is not a student (the true information); on the other hand, the buyer responds to
s3 with an acceptance, indicating that he is accepting to be on the mailing list (disinformation). As for (ii),
the response needs to conform to this information—e.g., if the seller says that s/he does not have the product
A, then the buyer should not assume that prodA is available, even though prodA is a possible assumption in
his/her KB. These considerations lead to more sophisticated responses.

**Definition 4.22 (Adaptive Response).** Let KA = (Pa, Ha, Na) and KB = (Pb, Hb, Nb) be NKBs of
an agent a and an agent b, respectively. Let γb = (G, S, R) be a proposal from b to a. A response γa =
(G1, S1, R1), with respect to KA, whose support is M, is an adaptive response to γb with respect to M by a
if one of the following conditions is satisfied:

1. γa = (T, ∅, ∅),
2. γa = (⊥, ∅, ∅), or
3. R ∩ H_a ⊆ S1, R^c ∩ S1 = ∅, and S^c ∩ M ⊆ R1.

In the above definition, the third item indicates that agent a, in the construction of the response, takes into
consideration

- The information that b has provided about his/herself (R ∩ H_a ⊆ S1).
- It does not assume anything contradictory to it (R^c ∩ S1 = ∅); and
- It corrects b by providing information about him/herself that b falsely assumed (S^c ∩ M ⊆ R1).

In the following, we say that γa is an adaptive response to γb if there exists a support M such that γa is an
adaptive response to γb with respect to M. We illustrate the definition in the next example.

**Example 4.23.** Let us consider again the seller and buyer from Examples 3.2 and 3.3. The proposal

γ1 = ({low}, {prodA, qualA, makC}, ∅)

by the buyer ("Can I have the prodA, made by makC, and with good quality for low price?")
is negotiable with respect to KA, since KA has a belief set (with disinformation) M containing
{low, prodA, qualA, makC} that also contains the assumption student. As such, the seller could produce
the following response:

γ1' = ({low}, {student}, {prodA, qualA, makC})

(i.e., "Only if you are a student.")

The NKB of the seller has another belief sets M2 containing {high, prodA, qualA, makC}. Observe that
low ≺ high for the seller. As such, the seller could respond with the proposal

γ1'' = ({high}, ∅, {prodA, qualA, makC})

(i.e., "That goes for the high price?").

Each response is an adaptive response to γ1.

We are now ready to define a notion called adaptive strategy. Unlike the strategies proposed in the previous
section, an adaptive strategy not only computes a response, it also updates the knowledge bases of negotiated
agents.

**Definition 4.24 (Adaptive Strategy).** Let KBa = (Pa, Ha, Na) and K_b = (Pb, Hb, Nb) be NKBs of
an agent a and an agent b, respectively. An adaptive strategy A (of a towards b) is a mapping from pairs
of rounds of negotiation between a and b, starting by a, and NKBs into pairs of proposals and NKBs, i.e., for
each proposal γ1 = (G1, S1, R1) of a and a response γ2 = (G2, S2, R2) by b to γ1,

A((γ1, γ2), KBa) = (γ, KBa')

where

1. γ = (G, S, R) is an adaptive response to γ2, R1 ⊆ R,
   (a) If γ2 is acceptable with respect to KBa then γ = (T, ∅, ∅) or G2 ≺* G;

   ACM Transactions on Computational Logic, Vol. V, No. N, Article A, Publication date: YYYY.
persistent strategy; thus, termination of negotiations between adaptive agents is guaranteed. An agent employing an adaptive strategy in negotiation will be called an adaptive agent. An adaptive agent imports the information received during the negotiation into his/her NKB, keeps this information for the next round of negotiation, and removes non-achievable goals from the set of goals. Furthermore, an adaptive agent prefers to accept a proposal if a better outcome cannot be achieved.

**Proposition 4.25.** Every negotiation among adaptive agents terminates.

**Proof.** This result derives directly from the simple observation that an adaptive strategy is also a self-persistent strategy; thus, termination of negotiations between adaptive agents is guaranteed. □

We illustrate the definition of adaptive strategy in the following example.

**Example 4.26.** Let us reconsider the seller and buyer with the NKBs from Examples 3.2 and 3.3. Let us assume that both agents are adaptive. Suppose that the seller starts with the (dishonest) proposal of Example 4.23:

$$\{(\text{low}), \{\text{cash}\}, \{\text{prod}_A, \text{qual}_A, \text{mak}_C\}\}.$$

The NKB of the buyer has two belief sets $M_1$ and $M_2$ containing $\text{prod}_A, \text{qual}_A, \text{mak}_C$ and low or lowest where

- $M_1$ contains lowest (application of $n_1$ of $K_b$);
- $M_2$ contains low (application of $n_3$ of $K_b$).

The buyer could accept this proposal (since $M_2$ supports it). It can also attempt to negotiate for a better price. The buyer first changes his/her KB to be the following NKB: $K_b' = (P_b', H_b', N_b')$ where $P_b' = (\Pi_b', A_b', L_b, B_b)$ with $(L_b, B_b) = (\emptyset, \{\text{mailing}\})$, $H_b' = \{\text{qual}_B, \text{prod}_B, \text{mak}_D\}$, and

$$\Pi_b' = \begin{cases} 
\text{purchase}_x \leftarrow \text{prod}_x, \text{qual}_x, \text{price}_3 \\
\neg\text{student} \leftarrow \\
\text{cash} \leftarrow \\
\text{qual}_A \leftarrow \\
\text{prod}_A \leftarrow \\
\text{mak}_C \leftarrow \\
\text{low}, \text{lowest} \leftarrow \\
2 < n_1 \quad 3 < n_2 \quad 3 < n_1 
\end{cases}$$

where $x \in \{A, B\}$ and $\text{price}_3 \in \{\text{low}, \text{lowest}\}$ and

$$A_b = \begin{cases} 
n_1 : \text{lowest} \leftarrow \text{mak}_C & n_2 : \text{lowest} \leftarrow \text{mak}_D & n_3 : \text{low} \leftarrow \text{mak}_C \\
n_5 : \text{qual}_B & n_5' : \neg\text{qual}_B & n_7 : \text{mak}_D \\
n_7' : \neg\text{mak}_D & n_9 : \text{prod}_B & n_9' : \neg\text{prod}_B 
\end{cases}$$

In this update, the buyer adds the facts $\text{mak}_C, \text{prod}_A$, and $\text{qual}_A$ to his/her program and removes these facts and their negations from the set of abducibles.

The new KB has a belief set containing $\{\text{lowest}, \text{mak}_C, \text{prod}_A, \text{qual}_A, \text{cash}\}$. This means that the buyer could respond with the proposal

$$\{(\text{lowest}), \{\text{mak}_C, \text{prod}_A, \text{qual}_A\}, \{\text{cash}\}\}$$

that explores whether he/she could get the product at the lowest price.
5. A PLATFORM FOR NEGOTIATION SYSTEMS

In this section, we describe an ASP-Prolog [Pontelli et al. 2011] based platform for the development of negotiation agents which employs ALD in their negotiation. The organization of the platform is illustrated in Figure 2. At the lowest level, we develop an ASP-Prolog layer that provides an implementation of abductive logic programming. This is used, in turn, to enable the representation of NKBs and for the basic handling of negotiation proposals. At the top level, we have Prolog programs that can interact and coordinate the execution of agents performing negotiations. The layers of this architecture are discussed in the following subsections.

![Negotiation Architecture Diagram](image)

Fig. 2. Negotiation Architecture

5.1. ASP-Prolog

The ASP-Prolog system [Pontelli et al. 2011] is an extension of a modular Prolog system which enables the integration of Prolog-style reasoning with Answer Set Programming (ASP) [Niemelä 1999; Marek and Truszczyński 1999]. The first implementation of ASP-Prolog dates back to 2004 [El-Khatib et al. 2004], and we have recently embarked in a redesign and re-implementation of the system using more modern Prolog and ASP technology, as discussed in [Pontelli et al. 2011].

An ASP-Prolog program is composed of a hierarchy of modules, where each module can be declared to contain either Prolog code or ASP code. Each module provides an interface which allows the module to export predicate definitions and import definitions from other modules. In the current implementation, the root of the module hierarchy is expected to be a Prolog module, that can be interacted with using the traditional Prolog-style query-answering mechanism.

Each module is provided with the ability to access the intended semantics of other modules; in particular, the intended models of each module (i.e., the least Herbrand model of a (pure) Prolog module and the answer sets of an ASP module) are themselves automatically realized as individual modules, that can be separately queried. The interactions among modules can be realized using the following built-in constructs:

- \( m : \text{model}(X) \) succeeds if \( X \) is the name of a module representing one of the intended models of the module \( m \) — i.e., a module representing the least Herbrand model (resp. an answer set) of a Prolog (resp. ASP) module \( m \).
- \( m : p \) succeeds if the atom \( p \) is entailed in all the intended models of the module \( m \); in particular, if \( m \) represents an answer set of another module, then this test will simply verify whether \( p \) is entailed by that particular answer set.
- The content of modules can be retrieved (using the predicate \( m : \text{clause}(R) \)) and modified by other modules. The predicates \text{assert} and \text{retract} can be applied to add or remove clauses from a module. For example, let us assume we have one Prolog module \( p_1 \) and two ASP modules \( q_1 \) and \( q_2 \):
Observe that the only answer set of \( q_1 \) is \( \{r, h\} \). Thus, \( q_1 : h \) is true. This implies that \( q_2 \) has two answer sets \( \{s, q\} \) and \( \{a, q\} \). Adding \( s \) to \( q_1 \) changes its answer set to \( \{h\} \).

The queries \( p_1 : p \), \( p_1 : t \), and \( q_1 : q \) are successful, while \( p_1 : s \) fails. On the other hand, the execution of \( p_1 : v \) would fail, adding the fact \( s \) to the module \( q_1 \).

### 5.2. ALP Modules

Since the semantics of ALP and ALP with preferences is defined by answer sets of logic programs, it is natural to view ALPs as another type of modules that can be accessed and used in the same way as other modules in ASP-Prolog. To achieve this goal, we extend ASP-Prolog with the following predicates:

- \( \text{use_alp}(+\text{Name}, +\Pi, +\mathcal{A}, +\text{Options}) \): this predicate has the same effect as the predicate \( \text{use_module}(P) \) for a Prolog or an ASP-module. It compiles the ALP program \( \langle \Pi, \mathcal{A} \rangle \) with the corresponding options specified in the list \( \text{Options} \) into a new ASP module, named \( \text{Name} \). The compilation process is illustrated in Algorithm 1. Similarly to the case of ASP modules, the ALP belief sets of \( \langle \Pi, \mathcal{A} \rangle \) are created as separate modules of the module produced by the algorithm. Accessing a belief set of the program or its content is done in the same way as for ASP modules. Some shorthands are also provided, e.g., \( \text{use_alp}(+\Pi, +\mathcal{A}) \) where the module name is specified as \( \Pi \).

- \( \text{most_preferred}(+\text{Name}, ?\mathcal{M}) \): this predicate allows users to query or check for a most preferred belief set of the ALP program referred to by \( \text{Name} \).

- \( \text{more_preferred}(+\text{Name}, ?\mathcal{M}_1, ?\mathcal{M}_2) \): this predicate allows users to compare two belief sets w.r.t. the \( \ll \) relation.

**ALGORITHM 1: ALP Compile**

**Require:** An abductive logic program \( P = \langle \Pi, \mathcal{A} \rangle \); a name \( \text{Name} \)

**Ensure:** An ASP module \( \text{Name} \).

Let \( P_0 = \emptyset \).

for all rules \( n \in \mathcal{A} \) do

Let \( n \) be of the form \( \text{head} \leftarrow \text{body} \);

Add to \( P_0 \) the rule \( \text{head} \leftarrow \text{body}, \text{ok}(n) \);

Add to \( P_0 \) the rule \( \{\text{ok}(n)\} \leftarrow \)

end for

Generate an ASP module containing the rules \( P' = \Pi \cup P_0 \).

**Example 5.1.** Consider the ALP \( P = \langle \Pi, \mathcal{A} \rangle \) with preferences:

\[
\Pi = \{ n_2 < n_1, \quad s \leftarrow, \quad \text{not } p, \text{ not } q \},
\]

\[
\mathcal{A} = \{ n_1 : p \leftarrow \text{ not } r, \quad n_2 : q \leftarrow \text{ not } r \}.
\]

\[4\]The plus/question mark preceding an argument in the specification of a predicate indicates whether the argument is an input/output argument of the predicate.
### Table I. Agent Specification

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Example: Seller agent</th>
</tr>
</thead>
<tbody>
<tr>
<td>declare_pr:</td>
<td>declare_pr: The program $P_s$ (Example 3.2)</td>
</tr>
<tr>
<td>The program $P$</td>
<td>The program $A_s$ (Example 3.2)</td>
</tr>
<tr>
<td>declare_pa:</td>
<td>declare_pa:</td>
</tr>
<tr>
<td>The program $A$</td>
<td></td>
</tr>
<tr>
<td>declare_lying:</td>
<td>declare_lying:</td>
</tr>
<tr>
<td>Literals in $L$</td>
<td>qual$_B$.</td>
</tr>
<tr>
<td>declare_bs:</td>
<td>declare_bs: qual$_A$.</td>
</tr>
<tr>
<td>Literals in $B$</td>
<td>student, cash, mailing.</td>
</tr>
<tr>
<td>declare_hypotheses:</td>
<td>declare_hypotheses:</td>
</tr>
<tr>
<td>Literals in $H$</td>
<td>high, low, lowest, prefer(low, lowest).</td>
</tr>
<tr>
<td>declare_goal:</td>
<td>declare_goal: prefer(high, low).</td>
</tr>
<tr>
<td>Literals in $N$ and the preference order $&lt;$</td>
<td>prefer(high, lowest).</td>
</tr>
</tbody>
</table>

Source: This is a table sourcenote. This is a table sourcenote. This is a table sourcenote.

Note: This is a table footnote. This is a table footnote. This is a table footnote.

*This is a table footnote. This is a table footnote. This is a table footnote.*

It is easy to see that $P$ has three belief sets $M_1 = \{p, s, n_2 < n_1\}$, $M_2 = \{q, s, n_2 < n_1\}$, and $M_3 = \{p, q, s, n_2 < n_1\}$ which are belief sets of $P$. Suppose that $\Pi$ and $A$ are stored in the files `pr.lp` and `pa.lp`, respectively. The command

```prolog
?- use_alp(m, pr, pa)
```

compiles $P = \langle \Pi, A \rangle$ into the module $m$. It also creates three modules $m_1$, $m_2$, and $m_3$ which correspond to $M_1$, $M_2$, and $M_3$, respectively. The next table displays some queries to $q$ and the corresponding answers.

<table>
<thead>
<tr>
<th>Query</th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>$?- m:\text{model}(X)$.</td>
<td>$X = p1; X = p2; X = p3$</td>
</tr>
<tr>
<td>$?- \text{findall}(E, \langle \text{current_predicate}(m:\text{P}_\text{A}), \text{functor}(E, P, A), m:E\rangle, L)$.</td>
<td>$L = [s, \text{prefer}(n1, n2)]$</td>
</tr>
<tr>
<td>$?- m:\text{model}(X), X:p$.</td>
<td>$X = p1; X = p3$</td>
</tr>
<tr>
<td>$?- \text{most_preferred}(m, X)$.</td>
<td>$X = p1$</td>
</tr>
</tbody>
</table>

The first query asks for belief sets of $m$ (the program $P$). The second query asks for all atoms that are true in $m$, i.e., those belonging to all belief sets of $P$. The third query identifies models in which $p$ (the atom) is true. The last query asks for a most preferred belief set of $m$.

#### 5.3. Negotiation Agent

The NKB layer of Fig. 2 allows the high-level description of the NKB of a negotiation agent and its mapping to a corresponding ALP module. A negotiation agent is described by a NKB $K = \langle P, H, N^\sim \rangle$, where $P = \langle \Pi, A, L, B \rangle$, $(L, B)$ is a disinformation with respect to $P$, $H$ is set of assumptions, and $N^\sim$ is set of negotiated conditions. For convenience, we introduce a simple specification language for agents as shown in Table I (left). The complete description of a NKB is provided in a file, with different sections corresponding to the different components of the NKB.

An agent is compiled into ALP (and, in turn, into ASP as discussed in the previous section) by the command

```prolog
load_agent(+Agent, +File, +[Options]),
```

where $Agent$ is the name of the agent, $File$ is file encoding the agent specification, and the list of options $\langle \text{Options} \rangle$ for use with the program $P = \langle \Pi, A \rangle$. The command will compile the NKB into a module, named $Agent$, linked to modules corresponding to the belief sets of the ALD-program $\langle \Pi, A, L, B \rangle$. This allows the users to access the belief sets, compute the
most preferred belief sets, as well as compare belief sets. The overall structure of the compiler is sketched in Algorithm 2.

**ALGORITHM 2: NKB Compilation**

**Require:** An Agent name  
**Require:** A NKB \( K = \langle P, H, N \rangle \) where \( P = (\Pi, A, L, B) \) and \( D = (L, B) \) is disinformation w.r.t. \( \langle \Pi, A \rangle \)  
**Ensure:** An ALP module representing \( \langle \Pi, A, D \rangle \).

Compute \( I = \{ r \mid r \in \Pi \} \) and \( head(r) \cap L = \emptyset \)

\( P_1 = A \cup \{ I \} \cup B \)

Assign labels to the rules in \( P_1 \)

Compute

\[
\Phi = \{ \text{prefer}(n_i, n_j) \mid n_i \in I \text{ and } n_j \in A \cup H \} \cup \\
\{ \text{prefer}(n_j, n_i) \mid n_j \in A \cup H \cup I \text{ and } n_i \in \{ L \cup B \} \}
\]

\( P_1 = P_1 \setminus I \cup \Phi \)

Compile the program \( P_1 = (P_1, P_1) \) using the use alp predicate.

**Example 5.2.** Consider the NKB \( K_s \) in Example 3.2. Suppose that the NKB is stored in seller.lp. Its description is shown on the right in Table I. The query `?- load_agent(s,seller)` will compile the NKB into an ALP module named s, denoting the agent s. The module is associated to the modules corresponding to the belief sets of the ALD program, whose content can be accessed using the ASP-Prolog module interface discussed earlier.

**5.4. Computing and Evaluating Proposals**

The proposal evaluation layer (Fig. 2) provides a collection of predicates to generate general proposals and to evaluate proposals for acceptability, rejectability, and negotiability. Let us discuss the key predicates; all examples refer to s and b as the seller and buyer whose NKBs are specified in Examples 3.2 and 3.3.

We assume that the NKBs are already compiled into modules s and b, respectively, using the load_agent command.

- **proposal(+Agent,[?G,_S,?R]):** this predicate succeeds if \( (G, S, R) \) is a proposal for the agent Agent. Note that this predicate can be used to generate as well as test a proposal. For example, the query `?- proposal(s,[G,S,R])` generates an arbitrary proposal that the seller agent \( s \) can create, given her NKB (Example 5.2); one possible answer is

  \( G = \{ \text{high} \}, S = \emptyset, R = \{ \text{prod}_A \} \).

- **proposal(+Agent,?M,[?G,?S,?R]):** this predicate succeeds if \( (G, S, R) \) is a proposal for the agent Agent with supporting belief set \( M \). For example, assume that the module \( s \) has a model represented by module \( s_1 \) containing \( \text{low} \), \( \text{student} \), and \( \text{prod}_B \); the query

  `?- proposal(s,s1,[G,S,R]).`

asks for a possible proposal for agent \( s \), supported by the belief set described by module \( s_1 \); it returns the answer \( G = \{ \text{low} \}, S = \{ \text{student} \}, R = \{ \text{prod}_B \} \). A most preferred proposal for \( s \) can be obtained using the query:

  `?- most_preferred(s, M), proposal(s,M,[G,S,R])`.  

- **acceptable(+Agent,[+G,+S,+R]):** this predicate succeeds if \( (G, S, R) \) is an acceptable proposal for the agent Agent. For example, the query

  `?- acceptable(s, [[high],[prod_A], []])`.  

asks if \( \{ \text{high} \}, \{ \text{prod}_A \}, \emptyset \) is acceptable for agent \( s \); this query succeeds.

- **negotiable(+Agent,[+G,+S,+R]):** this predicate succeeds if \( (G, S, R) \) is a negotiable proposal for the agent Agent. The query

  `?- most_preferred(s, M), proposal(s,M,[G,S,R]), acceptable(s,M,[G,S,R]).`
cess is an iterative process (implemented by the predicate agent. The proposal identified by 
main(Name1, Agent1, Name2, Agent2, [G,S,R]) :-

• load_agent(Name1, Agent1), load_agent(Name2, Agent2),
negotiation(Name1, Name2, [G,S,R]).

The negotiation starts with an agent identifying a most preferred proposal and proposing it to the other
agent. The proposal identified by [G, S, R] is the final outcome of a negotiation. The actual negotiation pro-
cess is an iterative process (implemented by the predicate round), which alternates generation of responses
between the two agents. In each round, the receiving agent computes a counter proposal and updates her NKB. The coordinator agent updates the history of the negotiation to ensure that agents do not repeat their answers. Multiple traces can be obtained by asking for different answers.

1: negotiation(A, B, [G1,S1,R1]) :-
2: most_preferred(A,M), proposal(A,M,[G,S,R]), print_prop(A,G,S,R),
3: round(A,B,[G,S,R],[A,[],[G,S,R]], [G1,S1,R1]).
5: response(B,[G1,S1,R1],[G,S,R]),
6: check_repeated((B,[G,S,R],[G1,S1,R1]),History),
7: ([G1,S1,R1]==[true,[],[]] -> write('Accepted') ;
8: ([G1,S1,R1]==[false,[],[]]-> write('Rejected') ;
9: append([B,[G,S,R],[G1,S1,R1]],History,History1),
10: agent(B, Hyp, _, _, _), intersection(Hyp, R, SH1),
11: negated(Hyp, HypN), intersection(HypN, R, SH2),
12: union(SH1, SH2, SH), update_KB(B, pr, true, SH),
13: round(B, A, [G1,S1,R1],History1,[G1,S1,R1])
14:).

where agent(·) provides the components of the agent and negated(S,S') is true for S' = S''. The subgoals in lines 7–12 are used to update the agent (see definition of adaptive agent), while line 6 avoids repetitions of proposals. The auxiliary predicate print_prop displays the current proposal (and the generating agent) on the screen.

**Example 5.3.** The following is an example of some negotiations when running the query

```
?- main(s, seller, b, buyer, [G,S,R]).
```

Agent s: proposal {[high],,[],[]}
G=[high], S=[], R=[]
Rejected yes ? ;
Agent b: proposal {[low],[qual_B, mak_C, prod_B],[]}
Agent s: proposal {[low],[qual],-[qual_B]}]
Agent b: proposal {[low],[cash],[]}
Agent s: proposal {[low],[cash],[]}
Agent b: proposal {[low],[qual_A, mak_C, prod_A], [cash]}
Accepted yes ?

The seller starts with the proposal {high,pr},∅,∅), which the buyer rejects. The buyer proposes {low},{qual_B,makC,prod_B},∅). This does not work for the seller, as he does not want to lie about qualB. The buyer proposes the alternative {low},{qual_A,makC,prod_A},∅) for which the seller wants cash. The buyer agrees and the seller agrees to sell the product, with BS about the quality of product A. □

6. DISCUSSIONS AND CONCLUSIONS
In this paper, we developed an abstract framework for negotiation based on logic programming. It can deal with incomplete information, preferences, disinformation, and compute proposals and conditions on a case-by-case basis using different negotiation strategies. We also introduced the design of a logic programming platform to implement negotiating agents. The architecture has been entirely developed using the ASP-Prolog system, taking advantage of the ability of combining Prolog and ASP modules. We believe this architecture is quite unique in the level of flexibility provided and in its ability to support easy extensions to capture, e.g., different agent strategies and behaviors.

Logic programming has been used to formulate negotiation by many researchers. In [Chen et al. 2006], the authors use answer sets of logic programs as a means for negotiation between agents. Their goal is to coordinate answer sets of two programs, and it has no mechanisms for developing proposals for a particular goal. Sadri et al. realize negotiation using abductive programs [Sadri et al. 2002]. In their framework, a pro-
gram specifies a negotiation plan to achieve a goal, and the behavior of each agent is operationally specified by an observe-think-act cycle. In our framework, an agent can generate proposals using abductive assumptions, and the behavior of agents are flexibly changed by strategies. Son and Sakama use logic programs with consistency restoring rules (CR-Prolog) to formulate negotiation [Son and Sakama 2009]. Unlike abductive programs, a CR-Prolog program considers the most preferred answer sets only. This limits flexibility in building proposals that are supported by less preferred belief sets.

Our work is in the same spirit as the approaches to argumentation-based negotiation (ABN) [Kakas and Moraitis 2006; Amgoud et al. 2006], in that it considers explanations as a part of a proposal/response. The main difference between our work and ABN lies in the flexibility of our approach. Our framework does not compute explanations for accepting/rejecting a proposal in advance as in [Amgoud et al. 2006], and it allows negotiators to non-monotonically modify their beliefs using incoming information. Kakas et al. [Kakas and Moraitis 2006] introduce priorities over arguments and use abduction to seek conditions to support arguments. They do not integrate abduction and preferences as done in this paper.

There have also been a number of proposals to formalize negotiation with multiple issues or incomplete information (e.g., [Fatima et al. 2004; 2005]). The key difference between these approaches and ours is that they rely on the use of utility functions and deadlines in the construction of a counter-offer (or a response). Our approach does not consider deadlines. It provides agents with a way to construct their responses which, together with their strategies, can take into consideration the agents’ preferences and disinformation.

It is important to note that all studies mentioned above model negotiation between honest agents, which is not always a realistic scenario. There are a few studies which provide a formal logic for negotiation between dishonest agents. Zlotkin et al. study negotiation in which agents may lie in [Zlotkin and Rosenschein 1991]. The study focuses on multi-agents in a dynamic environment, where agents act and interact to achieve their individual or cooperative goals. However, the study of Zlotkin et al. does not provide any computational method to construct (dis)honest proposals. The issue of dishonest negotiation has also been discussed in the context of game theory (e.g., [Ettinger and Jehiel 2010]). The approach differs from most of the works we have discussed so far, including our own.

The future focus of this work will move along several directions. For the framework to be applicable in more realistic situations, it might be better to allow agents to omit their own goal in a proposal. We are investigating possible ways to make this possible. In addition, we are currently extending the architecture as follows:

— We are implementing within our architecture several built-in agent strategies (e.g., to represent agents with different levels of dishonesty),
— We are moving our implementation platform to a truly concurrent model, where agents are concurrently active and they explicitly interact and communicate (e.g., through a Linda-style blackboard);
— We are exploring the modeling and implementation of real-world scenarios.

Let us also observe that this paper does not argue the issue of detecting and dealing with disinformation made by a dishonest agent. The investigation of these issues is also left for future work.

REFERENCES


ACM Transactions on Computational Logic, Vol. V, No. N, Article A, Publication date: YYYY.


Received October 2012; revised May 2013; accepted August 2013