Negotiation Using Logic Programming with Consistency Restoring Rules*

Tran Cao Son
Department of Computer Science
New Mexico State University
Las Cruces, NM 88003, USA
tson@cs.nmsu.edu

Chiaki Sakama
Computer and Communication Sciences
Wakayama University
Wakayama 640-8510, Japan
sakama@sys.wakayama-u.ac.jp

Abstract

We formalize negotiations using logic programming with consistency restoring rules (or CR-Prolog) [Balduccini and Gelfond, 2003]. Our formulation deals with incomplete information, preferences, and changing goals. We assume that each agent is equipped with a knowledge base for negotiation which consists of a CR-program, a set of possible assumptions, and a set of ordered goals. We use the notion of an answer set as a means to formalize the basic notions of negotiation such as proposal, response, negotiation, negotiation tree (protocol), etc. and discuss their properties.

1 Introduction

An intelligent agent situated in the real world often has to negotiate with others to achieve his/her objectives. Consider a typical negotiation between a seller and a buyer:

- *Seller:* Would you like to have this PC for $1000?
- *Buyer:* Can I get it for $900?
- *Seller:* Only if you pay by cash.
- *Buyer:* Done.

This simple negotiation shows that a negotiation between agents often involves reasoning with incomplete information and preferences. Here, the seller prefers to sell his/her PC for $1000 but he/she could accept $900 if the buyer pays by cash. The seller, at the beginning of the negotiation, does not have the information about the method of payment of the buyer. He/she would learn about this information during the negotiation. Furthermore, due to his/her preference, it is natural to expect that the seller would discuss the method of payment only if the buyer requests for a discount.

The above discussion implies that any formalism for reasoning about negotiation needs to have the capability to deal with preferences and incomplete information. Surprisingly, there have been only a few attempts to formalize negotiation using logic programming despite the fact that logic programming is known as a knowledge representation language suitable for reasoning with incomplete information and preferences (e.g., [Baral, 2003; Gelfond and Leone, 2002]). These attempts use logic programming under the answer set semantics but rely on techniques in belief revision to compute deals between agents (e.g., [Chen *et al.*, 2006]).

The example also shows that agents might change their goals during the negotiation. This is also a key issue in formalizing negotiation, which seems to prefer argumentation-based negotiation [Rahwan *et al.*, 2003]. Recent proposals on formalizing negotiation (see, e.g., [Amgoud *et al.*, 2006; Kakas and Moraitis, 2006; Rahwan *et al.*, 2003]) seem to be in-line with this conclusion. This raises a question on the suitability of logic programming in formalizing negotiation. Our goal in this paper is to address this question.

In this paper, we use logic programming with consistency restoring rules (or CR-Prolog) [Balduccini and Gelfond, 2003] to formalize negotiation. We represent the knowledge (for negotiation) of each agent as a CR-Prolog program extended with a set of assumptions and a set of ordered goals and use answer sets as a means to define proposals and their acceptability as well as a negotiation and related notions. We will begin with a short review of CR-Prolog. After that, we define the notion of a negotiation knowledge base, proposal, and discuss our proposal classification. We then present our formalization of negotiation based on these notions and discuss how it can be extended to accommodate agents' changes in goals. Finally, we relate our work to others and discuss future work.

2 LP with Consistency Restoring Rules

A logic program Π is a set of normal rules of the form

$$c_1 \mid \ldots \mid c_k \leftarrow a_1, \ldots, a_m, not\ a_{m+1}, \ldots, not\ a_n \qquad (1)$$

where $0 \leq m \leq n$, $0 \leq k$, each a_i or c_j is a literal of a propositional language[1] and not represents *negation-as-failure*. A negation as failure literal (or naf-literal) is of the form $not\ a$ where a is a literal. For a rule of the form (1), the left and right hand sides of the rule are called the *head* and the *body*, respectively. Both the head and the body can be empty. When the head is empty, the rule is called a *constraint*. When the body is empty, the rule is called a *fact*.

For a rule r of the form (1), $H(r)$ and $B(r)$ denote the left and right hand side of $\leftarrow$, respectively; $head(r)$ denotes the

*Partially supported by NSF grants IIS-0812267, MII-0220590, CREST-0420407, and a JSPS award.

[1] Rules with variables are viewed as a shorthand for the set of its ground instances.

set $\{c_1, \ldots, c_k\}$; and $pos(r)$ and $neg(r)$ denote $\{a_1, \ldots, a_m\}$ and $\{a_{m+1}, \ldots, a_n\}$, respectively.

Consider a set of ground literals X. X is consistent if there exists no atom a such that both a and $\neg a$ belong to 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 nonempty 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.

For a consistent set of ground literals S and a program Π, the *reduct* of Π w.r.t. S, denoted by Π^S, is the program obtained from the set of all ground instances of Π by deleting (**i**) each rule that has a naf-literal *not* a in its body with $a \in S$, and (**ii**) all naf-literals in the bodies of the remaining rules.

S is an *answer set (or a stable model)* of Π [Gelfond and Lifschitz, 1991] if it satisfies the following conditions: (**i**) If Π does not contain any naf-literal (i.e. $m = n$ in every rule of Π) then S is a minimal consistent set of literals that satisfies all the rules in Π; and (**ii**) If Π does contain some naf-literal ($m < n$ in some rule of Π), then S is an answer set of Π if S is the answer set of Π^S. (Note that Π^S does not contain naf-literals, its answer set is defined in the first item.) A program Π is said to be *consistent* if it has a consistent answer set. Otherwise, it is inconsistent.

CR-Prolog introduces an additional type of rules, called *consistency restoring rules* (or *cr-rules*), of the form

$$r: \quad c_1 \mid \ldots \mid c_k \stackrel{+}{\leftarrow} a_1, \ldots, a_m, not\ a_{m+1}, \ldots, not\ a_n \quad (2)$$

where r is the name of the rule and c_i's and a_j's are literals as in the rule (1). Observe that a cr-rule can be viewed as a normal rule (of the form (1)) by dropping its name and replacing the connective $\stackrel{+}{\leftarrow}$ with $\leftarrow$.

A CR-program P is given by a pair (P^r, P^c) where P^r is a set of rules of the form (1) and P^c is a set of rules of the form (2). Let C be a subset of P^c. By $P^r \cup C$ we denote the program consisting of rules in P^r and the cr-rules in C viewed as normal rules.

Answer sets of P are defined as follows. If P^r is consistent, then any answer set of P^r is an answer set of P. Otherwise, an answer set of P is an answer set of $P^r \cup C$ where C is a minimal subset of P^c such that $P^r \cup \{H(r) \leftarrow B(r) \mid H(r) \stackrel{+}{\leftarrow} B(r) \in C\}$ is consistent.

Example 1. Consider the program $P = (P^r, P^c)$ where

$$P^r = \{s \leftarrow . \quad \leftarrow not\ p, not\ q.\}$$

and $$P^c = \{r_1: \ p \stackrel{+}{\leftarrow} not\ r. \quad r_2: \ q \stackrel{+}{\leftarrow} not\ r.\}$$

P^r is inconsistent. P has two answer sets $\{p, s\}$ and $\{q, s\}$, which are obtained by adding $\{r_1\}$ or $\{r_2\}$ to P^r correspondingly.

Although adding $\{r_1, r_2\}$ to P^r also creates a consistent program, its answer set $\{p, q, s\}$ is not considered as an answer set of P since the set $\{r_1, r_2\}$ is not a minimal subset of P^c with this property. □

When multiple rules can be used in restoring the consistency of a program, a *preference* relation in the form of $prefer(r_1, r_2)$ can be added to the program to force the application of more preferred cr-rules in restoring the consistency of the program. It is assumed that $prefer$ is a transitive and anti-symmetric relation among cr-rules of a program. The semantics of CR-programs ensures that if $prefer(r_1, r_2)$ is specified then the rule r_2 should be used in restoring the consistency of the program only if no solution containing r_1 is possible and the two rules are never used at the same time [Balduccini, 2007]. A CR-program (P^r, P^c) is said to be *consistent* if it has at least one answer set. For example, if we add $prefer(r_1, r_2)$ to P^r then $P = (P^r, P^c)$ from the previous example will have only one answer set $\{p, s\}$.

3 Negotiation Knowledge Bases, Proposals, and Proposal Classification

For a set of literals L, $Goal(L)$ denotes the set of constraint $\{\leftarrow not\ l \mid l \in L\}$, $\{L \stackrel{+}{\leftarrow}\}$ denotes the set of cr-rules $\{r_l : l \stackrel{+}{\leftarrow} \mid l \in L\}$, and $lit(L)$ the set $\{l, \neg l \mid l \in L \text{ or } \neg l \in L\}$. A negotiation knowledge base is defined as follows.

Definition 1 (Negotiation KB)**.** *A* negotiation knowledge base *K (or* n-KB*, for short) is a tuple $\langle P^r, P^c, H, N^{\prec}\rangle$ where*

- *(P^r, P^c) is a CR-program,*
- *$N^{\prec}$ is a set of* negotiated literals *associated with a strict partial order $\prec$ on its elements, and*
- *H is a set of literals (called* assumptions*) such that $H \cap head(P^r) = \emptyset$ and $\{H \stackrel{+}{\leftarrow}\} \subseteq P^c$.*

The n-KB is consistent *if (P^r, P^c) is consistent.*

Intuitively, the program (P^r, P^c) serves as a means for the agent to negotiate about a predefined objective[2] where P^r consists of normal rules defining the domain-specific knowledge of the agent and P^c consists of cr-rules defining possible negotiation strategies. $N^{\prec}$ contains literals expressing the desired properties of outcome for which the n-KB is developed. $\prec$ represents a preference order of the agent with respect to the negotiated literals; $a \prec b$ means that b is preferred to a. The set of assumptions H represents information that the agent might not have at the beginning of the negotiation and some of this information might be known to him/her during the negotiation. Since the CR-program (P^r, P^c) serves as a means for the agent to generate (counter)-supporting arguments, hypotheses, etc. in his/her negotiation, we will often require that (P^r, P^c) is consistent, i.e., the n-KB is consistent.

We will now present two typical n-KBs, one of the seller and one of the buyer, that we will use as running examples. Intuitively, the seller agent will have a KB for him/her to negotiate a *sale* while a buyer agent will have a KB for him/her to negotiate a *purchase*. The buyer will want to get the best (lowest) price for the purchase that he/she is negotiating for while the seller would prefer to obtain the best (highest) price for the sale. For simplicity, we assume that the two knowledge bases discuss the same product.

Example 2 (Seller n-KB)**.** A seller agent S uses the following n-KB in negotiating with his customers about the possible prices (of a certain product) that he can offer. $K_S = \langle P^r_S, P^c_S, H_S, N^{\prec}_S\rangle$ where[3]

- P^r_S consists of the following rules:

[2] For generality, this objective can be specified as a parameter of the knowledge base. This makes the notations a little more complicated. To focus on the main points, we opt out of this option.

[3] Arithmetic predicates are written in infix notation.

$$
\begin{array}{rcl}
whole\_sale\_customer & \leftarrow & registered. \\
student\_customer & \leftarrow & student. \\
senior\_customer & \leftarrow & age \geq 65. \\
sale & \leftarrow & \mathbf{sale\_price}. \\
 & \leftarrow & not\ sale. \\
prefer(r_1, r_i) & \leftarrow & (\text{for } i > 1) \\
prefer(r_i, r_5) & \leftarrow & (\text{for } i \in \{2, 3, 4\})
\end{array}
$$

where $\mathbf{sale\_price} \in \{high\_pr, low\_pr, lowest\_pr\}$ and the set of facts $\{made\_in\_L,\ maker\_A,\ \neg maker\_C,\ \neg maker\_B\}$. Here, P_S^r defines various types of customers, the predicate $sale$, and the preferences among the cr-rules of the KB. It also states some facts about the product at hand (e.g., it is made in location L and is a product of $maker\_A$).

- H_S is a set of assumptions representing information that the seller needs to verify about his clients during the negotiation. It is the set of literals that can be built from the atoms[4]

$$
\left\{
\begin{array}{l}
registered,\ student,\ age \geq 65, \\
good\_credit,\ quantity \geq 100,\ pay\_cash
\end{array}
\right\}
$$

- $N_S^{\prec}$ is the set of atoms that the agent should be negotiated about and is defined by $N_S^{\prec} = \{high\_pr,\ low\_pr,\ lowest\_pr\}$ with $\prec = \{lowest\_pr \prec low\_pr \prec high\_pr\}$. It says that the seller prefers the $high\_pr$ over low_pr and $lowest\_pr$.
- P_S^c consists of $\{H_S \stackrel{+}{\leftarrow}\}$ and

$$
\begin{array}{lll}
r_1: & high\_pr & \stackrel{+}{\leftarrow} \\
r_2: & low\_pr & \stackrel{+}{\leftarrow} senior\_customer \\
r_3: & low\_pr & \stackrel{+}{\leftarrow} student\_customer, \\
 & & good\_credit. \\
r_4: & low\_pr & \stackrel{+}{\leftarrow} student\_customer, pay\_cash. \\
r_5: & lowest\_pr & \stackrel{+}{\leftarrow} whole\_sale\_customer, \\
 & & quantity \geq 100.
\end{array}
$$

P_S^c specifies different pricing scenarios. Intuitively, r_1 says that any customer who agrees to buy the product with $high\_pr$ is welcome! A senior citizen is entitled to a discount (low_pr) and the same holds for a student with $good\_credit$ history. Whole sale customers are entitled to the biggest discount if they buy at least 100 units.

It is easy to see that (P_S^r, P_S^c) is consistent, i.e., K_S is a meaningful n-KB.

Finally, it is worth noting that the semantics of CR-Prolog ensures that no answer set of (P_S^r, P_S^c) would contain two possible prices, i.e., guaranteeing that the choice of a price is unique for the seller. □

An n-KB of a buyer may look as follows.

Example 3 (Buyer n-KB)**.** Let B be a buyer agent with the n-KB $K_B = \langle P_B^r, P_B^c, H_B, N_B^{\prec} \rangle$ where

- H_B is the set of literals constructed from atoms in $\{maker\_A, maker\_B, maker\_C, made\_in\_L\}$.
- P_B^r consists of the set of facts $\{age = 25,\ student,\ pay\_cash,\ \neg good\_credit,\ quantity = 1\}$ and the following rules

$$
\begin{array}{rcl}
purchase & \leftarrow & \mathbf{purchase\_price}. \\
 & \leftarrow & not\ purchase. \\
prefer(r_i, r_1) & \leftarrow & (i > 1) \\
prefer(r_4, r_i) & \leftarrow & (i \in \{2, 3\})
\end{array}
$$

where $\mathbf{purchase\_price} \in \{high\_pr, low\_pr, lowest\_pr\}$.

- P_B^c is the union of $\{H_B \stackrel{+}{\leftarrow}\}$ and the following rules:

$$
\begin{array}{lll}
r_1: & high\_pr & \stackrel{+}{\leftarrow} make\_A, not\ made\_in\_L \\
r_2: & low\_pr & \stackrel{+}{\leftarrow} make\_A, made\_in\_L \\
r_3: & low\_pr & \stackrel{+}{\leftarrow} maker\_B \\
r_4: & lowest\_pr & \stackrel{+}{\leftarrow} maker\_C
\end{array}
$$

- $N_B^{\prec} = \{high\_pr, low\_pr, lowest\_pr\}$ with $\prec = \{high\_pr \prec low\_pr \prec lowest\_pr\}$.

The n-KB of the buyer agent is different from that of the seller. It represents information about the buyer (e.g., a student, age 25) and his/her goal (e.g., $quantity = 1$). Naturally, the buyer has a different priority in its negotiated atoms: he/she prefers to pay the lowest price.

Again, we can check that (P_B^r, P_B^c) is consistent, and hence, K_B is a consistent n-KB. □

We will now define various notions that will allow us to formalize a negotiation protocol between two agents. We start with the notion of a proposal. Intuitively, the notion of a proposal must be able to capture utterances like the following:

- (Seller) I would like to sell you this product with the $high\_pr$.
- (Buyer) I can afford the $high\_pr$ if it is made by $maker\_A$ and it is not manufactured in L (i.e., not $made\_in\_L$).
- (Seller) We do have the product of $maker\_A$ and it is not manufactured in L ($made\_in\_L$) but you need to be a student and pay in cash to get low_pr.
- etc.

Each utterance usually contains some properties of the goal and the conditions for their acceptance. We therefore define a proposal as follows.

Definition 2 (Proposal)**.** *Let $K = \langle P^r, P^c, H, N^{\prec} \rangle$ be an n-KB of an agent, say A, and G be a set of goals $G \subseteq N^{\prec}$. Let M be an answer set of $(P^r \cup Goal(G), P^c)$, $S = M \cap H$, and $R \subseteq M \setminus H$.*

We call $\langle G, S \rangle$ a proposal *for G by A (w.r.t. K and M) and $\langle G, S, R \rangle$ an* extended proposal *for G by A (w.r.t. K and M).*

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

Intuitively, a proposal $\langle G, S \rangle$ states that the goal of A is to negotiate for G and the reason that A proposes the goal G is that he/she has a supporting argument for it, in which he/she assumes S to be true. An extended proposal provides further information supporting the goal G. We refer to G and S as *goal* and *support* of $\langle G, S \rangle$. In the following, a proposal (an extended proposal) for G by A w.r.t. K and M is often shortened to a proposal (an extended proposal) for G by A when K is clear from the context and the presence of M is unimportant for our discussion.

Observe that the semantics of a CR-program requires that the set of cr-rules to be used in the construction of its answer sets is minimal. The next proposition is obvious.

[4]For a set of atoms X, $\{a, \neg a \mid a \in X\}$ is the set of literals that can be built from X.

Proposition 1. *If $\langle G, S\rangle$ is a proposal w.r.t. K then there exists no other proposal $\langle G, S'\rangle$ w.r.t. K such that $S' \subsetneq S$.*

Example 4. For S (Exp. 2), $\langle\{high\_pr\}, \emptyset\rangle$ and $\langle\{low\_pr\}, \{age{\geq}65\}\rangle$ are two possible proposals. For B (Exp. 3), two possible proposals are: $\langle\{lowest\_pr\}, \{maker\_C\}\rangle$ and $\langle\{low\_pr\}, \{maker\_B\}\rangle$.

Given an agent A and a proposal $\gamma = \langle G, S\rangle$ from another agent, say B, we can see one of the following cases:

- A accepts γ: This means that γ could be a proposal of A. In doing so, A must consider whether he/she can achieve his/her goal by accepting γ and whether the assumptions in S is consistent with his/her knowledge and assumptions.
- A rejects γ: This means that there is no possible way that A can view γ as his/her proposal.
- A sees some alternative proposals for the goal of γ, yet γ is not suitable for A, i.e., A considers γ a negotiatable proposal.

This leads us to the following definition.

Definition 3 (Acceptable/Rejectable/Negotiatable Proposal)**.** *Let $K = \langle P^r, P^c, H, N\rangle$ be an n-KB and $\gamma = \langle G, S\rangle$ be a proposal from another agent. Let $Q = (P^r \cup Goal(G), P^c)$.*

- *γ is* acceptable *w.r.t. K if Q has an answer set M such that $M \cap H = S \cap H$ and $M \cup S$ is consistent.*
- *γ is* rejectable *if Q is inconsistent.*
- *γ is* negotiatable*, otherwise.*

Intuitively, Q encodes the set of possible proposals for G by the agent with the n-KB K. Thus, if Q is inconsistent then the proposal is rejectable. γ is acceptable if Q has an answer set M (representing a proposal for G w.r.t. K) such that M is consistent with S and $M \cap H = S \cap H$. The first condition is needed since a negotiated item is acceptable to both parties only if their supports are consistent. The second condition implies that both proposals need to use the same set of shared assumptions as well. This is necessary mainly because of the requirement used in the definition of an answer set of CR-programs. The third item of the definition is clear: a proposal is negotiatable if it is neither acceptable nor rejectable.

Let K be an n-KB and $\Gamma_a(K)$, $\Gamma_n(K)$, and $\Gamma_r(K)$ be the set of proposals that are acceptable, negotiatable, and rejectable w.r.t. K, respectively. It is easy to see

Proposition 2. *Let K be an arbitrary n-KB. Then, $\Gamma_a(K)$, $\Gamma_u(K)$, and $\Gamma_r(K)$ are pairwise disjoint. Furthermore, $\gamma \in \Gamma_a(K) \cup \Gamma_u(K) \cup \Gamma_r(K)$ for every proposal γ.*

We illustrate the above definitions in the next example.

Example 5. Consider K_S from Example 2. We have that

- $\langle\{high\_pr\}, \emptyset\rangle$ is acceptable w.r.t. K_S since $(P_S^r \cup Goal(\{high\_pr\}), P_S^c)$ has an answer set M, $M \cap H_S = \emptyset$.
- $\langle\{low\_pr\}, \emptyset\rangle$ is a negotiatable proposal w.r.t. K_S. There are three answer sets of $(P_S^r \cup Goal(\{low\_pr\}), P_S^c)$: M_1 contains the set of assumptions $\{age \geq 65\}$, M_2 contains $\{student, good\_credit\}$, and M_3 contains $\{student, pay\_cash\}$. None of these answer sets satisfies the conditions in the first Item of Def. 3.
- $\langle\{lowest\_pr\}, \{quantity = 1\}\rangle$ is a rejectable proposal w.r.t K_S. The only answer set of $(P_S^r \cup Goal(\{lowest\_pr\}), P_S^c)$ contains $quantity{\geq}100$ which contradicts with $quantity{=}1$. □

4 Negotiation Using n-KBs: Keeping the Goal

We will now present a model of negotiation between two parties A and B who use n-KBs K_A and K_B respectively in their negotiation. We assume that (*i*) K_A and K_B, the n-KBs of A and B, respectively, share the same language; (*ii*) Agents are willing to accept new assumptions during their negotiations as long as they do not cause inconsistency in their n-KBs. (*iii*) Agents do not provide wrong assumptions as explanation for not accepting a proposal, i.e., they are honest.

One of the most important questions that an agent has during a negotiation is what can he/she do about a current proposal, say $\langle G, S\rangle$ directed to him/her by another agent. Naturally, the agent can either accepts, rejects, or responds by putting forward a new proposal. In doing so, Def. 3 should be used. However, Def. 3 does not provide an answer for the third case, when the agent needs to respond by a new proposal. In the next definition, we address this issue. As we expect that agents will negotiate in more than one rounds and in each round, an agent, besides his/her proposal, could indicate some facts, say R, that cannot be accepted by him/her. As such, a response will be defined given an extended proposal. For a set of literals R and then n-KB $K = \langle P^r, P^c, H, N^{\prec}\rangle$, by $K \ominus R$ we denote the n-KB $\langle P^r, P^c, H \setminus lit(R), N^{\prec}\rangle$, which is obtained from K by eliminating the literals constructible from R from the set of assumptions of K. Observe that any proposal w.r.t. $K \ominus R$ cannot contain an assumption belonging to $lit(R)$.

Definition 4 (Response)**.** *Let $K_A{=}\langle P^r, P^c, H, N^{\prec}\rangle$ be an n-KB of an agent A and $\omega_B{=}\langle G, S, R\rangle$ be an extended proposal by B w.r.t. its n-KB K_B. A* response *to ω_B by A w.r.t. K_A is an extended proposal by A and is defined as follows.*

- *If $\langle G, S\rangle$ is acceptable w.r.t. $K_A{\ominus}R$, then the response is $\langle\top, \emptyset, \emptyset\rangle$, representing* `accept`.
- *If $\langle G, S\rangle$ is rejectable w.r.t. $K_A \ominus R$, then the response is $\langle\bot, \emptyset, \emptyset\rangle$, representing* `reject`.
- *If $\langle G, S\rangle$ is negotiatable w.r.t. $K_A \ominus R$, then a response is an extended proposal $\langle G, S', F\rangle$ by A w.r.t. $K_A \ominus R$ for which there exists an answer set M of $(P^r \cup Goal(G), P^c \setminus lit(R))$ such that*
  - *$S' = M \cap H$.*
  - *$\{\neg l \mid \neg l \in M, l \in S\} \subseteq F \subseteq \{\neg l \mid l \in S \setminus H\}$.*

We say that the response is constructive *if it is either $\langle\top, \emptyset, \emptyset\rangle$, $\langle\bot, \emptyset, \emptyset\rangle$, or satisfying that $F = \emptyset$ implies that $S \subseteq M$.*

By $\beta(K_A, \langle G, S, R\rangle)$ we denote the set of all responses to $\langle G, S, R\rangle$ from A w.r.t. K_A.

We assume that R contains literals that should not be considered by A. Therefore, the response is computed w.r.t. the n-KB $K_A \ominus R$. Under this assumption, the first two items are straightforward but the last item deserves some explanations. Clearly, a response should be an extended proposal by A w.r.t. $K_A{\ominus}R$. Furthermore, it should contain information that supports A's new proposal and contradicts with the assumptions made by B. This leads to the construction of $\langle G, S', F\rangle$ with the proposed properties.

Example 6. Consider $\omega_B^1 = \langle\{low\_pr\}, \{maker\_B\}, \emptyset\rangle$. This is an acceptable proposal w.r.t. K_B and is a negotiatable proposal w.r.t. K_S. The seller has three possible responses:

ω_S^1=$\langle\{low\_pr\}, \{age \geq 65\}, \{\neg maker\_B\}\rangle$
ω_S^2=$\langle\{low\_pr\}, \{student, good\_credit\}, \{\neg maker\_B\}\rangle$
ω_S^3=$\langle\{low\_pr\}, \{student, pay\_cash\}, \{\neg maker\_B\}\rangle$
The three responses correspond to the answer sets M_1, M_2, and M_3 in Example 5. Observe that $\neg maker\_B$ is a fact in K_S and $maker\_B$ is an assumption in ω_B^1. □

We will now define the notion of a negotiation.

Definition 5 (Negotiation)**.** *Let A and B be two agents and K_A and K_B be their n-KBs respectively. A* negotiation *between A and B for G, starting with A, is a possible infinite sequence of extended proposals $\omega_1, \ldots, \omega_n, \ldots$ where $\omega_i = \langle G_i, S_i, F_i\rangle$ and*

- *$\omega_1 = \langle G, S, \emptyset\rangle$ and $\langle G, S\rangle \in \alpha(K_A, G)$.*
- *$\omega_{i+1} \in \beta(K_{i-1}, \omega_i)$ for every $i > 1$ where*
  - *$K_1 = K_A$ and $K_{2k+1} = K_{2k-1} \ominus F_{2k}$ for $k > 0$; and*
  - *$K_0 = K_B$ and $K_{2k+2} = K_{2k} \ominus F_{2k+1}$ for $k \geq 0$.*

A negotiation ends *at i if ω_i is either $\langle\top, \emptyset, \emptyset\rangle$ or $\langle\bot, \emptyset, \emptyset\rangle$.*

A negotiation is a series of responses between two agents, who, in alternation, takes into consideration the other's response and puts forward a new response, which is either `accept` ($\langle\top, \emptyset, \emptyset\rangle$), `reject` ($\langle\bot, \emptyset, \emptyset\rangle$), or a new proposal with explanations why the last proposal (of the other agent) were not acceptable to him/her.

Definition 6 (Un/Successful Negotiation)**.** *A negotiation is* successful *(resp.* unsuccessful*) if it is finite and ends with $(K, \langle\top, \emptyset, \emptyset\rangle)$. (resp. $(K, \langle\bot, \emptyset, \emptyset\rangle)$). A negotiation is* constructive *if it contains only constructive responses.*

Some negotiations between the buyer B (Example 3) and the seller S (Example 2) are given next.

Example 7. The following is an unsuccessful negotiation.
$S : \langle\{lowest\_pr\}, \{registered, quantity \geq 100\}, \emptyset\rangle$
$B : \langle\{lowest\_pr\}, \{maker\_C\}, \{quantity = 1\}\rangle$
$S : \langle\bot, \emptyset, \emptyset\rangle$
This corresponds to the following:

- (Seller): I can offer you the lowest price if you register and buy at least 100 units.
- (Buyer): How about if it is from $maker\_C$, but anyway I would only buy one unit.
- (Seller): Sorry, I could not offer it at this time. □

Example 8. The following is a successful negotiation.
$B_1 : \langle\{low\_pr\}, \{maker\_B\}, \emptyset\rangle$
$S_1 : \langle\{low\_pr\}, \{age \geq 65\}, \{\neg maker\_B\}\rangle$
$B_2 : \langle\{low\_pr\}, \{maker\_A, made\_in\_L\}, \{age = 25\}\rangle$
$S_2 : \langle\{low\_pr\}, \{student, good\_credit\}, \emptyset\rangle$
$B_3 : \langle\{low\_pr\}, \{maker\_A, made\_in\_L\}, \{\neg good\_credit\}\rangle$
$S_3 : \langle\{low\_pr\}, \{student, pay\_cash\}, \emptyset\rangle$
$B_4 : \langle\top, \emptyset, \emptyset\rangle$
S_1 responds to B_1 and indicates that he/she cannot assume $maker\_B$ (as $\neg maker\_B$ is a fact in K_S). The seller wants to offer low_pr to the buyer but assumes that $age \geq 65$. This assumption does not work well for the buyer as $age = 25$ is true in K_B. Hence, the buyer puts forwards another proposal (B_2). This time, the assumptions made by the buyer are not conflict with K_S but the seller requires a new set of assumptions (S_2). The negotiation continues until the buyer accepts the offer and agrees to pay by cash. □

We will show next that if the agents are constructive in their negotiation then their negotiation will eventually terminate. Proof relies on the fact that agents accumulate assumptions and the n-KBs are finite and is omitted to save space.

Theorem 1. *Every constructive negotiation is finite.*

A negotiation represents one possible way for two agents to reach an agreement (or disagreement). In the course of reaching an agreement, two agents might have different alternatives. The notion of a negotiation tree, to be defined next, will account for all possible negotiations for a goal between two agents. We will make use of the well-known notation of a tree. By the level of a node in a tree we mean the number of links lying on the path connecting the root to the node. Also, for a response $\omega = \langle G, S, F\rangle$, we use $\omega.G$, $\omega.S$, and $\omega.F$ to denote G, S, and F respectively.

Definition 7 (Negotiation Tree)**.** *Let A and B be two agents with the n-KB K_A and K_B respectively. A* negotiation tree *between A and B for G, starting with A, is a labeled tree $T_{A,B,G}$ where*

- *the root of $T_{A,B,G}$ is G;*
- *every child of G has the label of the form $(K_A, \langle G, S, \emptyset\rangle)$ where $\langle G, S\rangle \in \alpha(K_A, G)$;*
- *if $\eta = (K_A, \omega)$ is a node at level 1, then every child of η has the label of the form (K_B, ω') where $\omega' \in \beta(K_B, \omega)$;*
- *if $\eta = (K, \omega)$ is a node at level i, $i \geq 2$, whose parent has the label (K', ω'), then every child of η has the label of the form $(K' \ominus \omega.F, \omega'')$ where $\omega'' \in \beta(K', \omega)$; and*
- *$(K, \langle\top, \emptyset, \emptyset\rangle)$ and $(K, \langle\bot, \emptyset, \emptyset\rangle)$ do not have children.*

We classify negotiation tree as follows.

Definition 8 (Classification of Negotiation Tree)**.** *A negotiation tree is* finite *if it has finite number of nodes; it is* successful *if it has a leaf whose label is of the form $(K, \langle\top, \emptyset, \emptyset\rangle)$; and it is* unsuccessful *if all of its leaves have a label of the form $(K, \langle\bot, \emptyset, \emptyset\rangle)$.*

The notion of constructive negotiation is extended to negotiation tree as follows.

Definition 9 (Constructive Negotiation Tree)**.** *A negotiation tree is constructive if for every node $\eta = (K, \omega)$ at the level $i \geq 2$, whose parent has the label (K', ω'), the set $\{\omega' \mid (K, \omega')$ is a child of $\eta\}$ consists of all and only constructive responses to $\langle\omega.G, \omega.S\rangle$ w.r.t. $K' \ominus \omega.F$.*

We can prove the following theorem.

Theorem 2. *Every constructive negotiation tree is finite.*

Algorithm 1 allows us to predict whether or not a successful negotiation for a goal G between A and B exists.
Using Theorems 1-2, we can show that Algorithm 1 always terminates. This algorithm can be used in developing for a negotiation protocol in the style presented in [Kakas and Moraitis, 2006]. We omit this here due to space limitation.

5 Negotiation for the Best Possible Result

A negotiation tree details possible negotiations between two agents who maintain their goal during the negotiation. In reality, agents might change their goals during the negotiation.

As it turns out, this can be easily accommodated in our framework. To do so, we only need to introduce the notion of a relaxation (or strengthening) of a goal.

Definition 10 (Relaxation/Strengthening)**.** *Let* $K = \langle P^r, P^c, H, N^{\prec}\rangle$ *be an n-KB and* G *and* G' *be two set of literals* $G, G' \subseteq N^{\prec}$*. We say that* G' *is a* relaxation *of* G *(or* G *is a* strengthening *of* G'*) if either* $G' \subsetneq G$ *or* $G' \neq G$ *and for every* $l \in G' \setminus G$ *there exists some* $l' \in G$ *such that* $l \prec l'$*.*

A *goal change* is either a relaxation or a strengthening of the goal. With the introduction of a goal change, Definitions 4, 5, 7, and Algorithm 1 can be modified to allow for responses that take into consideration a new goal. Space limitation prevents us from giving the full details of this development but they are rather straightforward. The key idea is to consider a change in a goal (*i*) as the starting of a new negotiation (as in Definition 5); and (*ii*) only necessary if the preceding negotiation leads to a rejection by either agent. Theorems 1, 2, and the termination of the modified algorithm can be proved accordingly.

6 Discussion and Conclusion

We formalize negotiation using CR-Prolog and define the basic concepts of negotiation using answer sets. The main features of our formalism are that it (*a*) includes support (as explanation) in a proposal/response; (*b*) can deal with incomplete information, preference, and changes in goal; (*c*) computes proposals/responses (and their support) on a case-by-case basis. Proposals/responses can be computed using available implementation of CR-Prolog. We provide an algorithm for predicting the result of a negotiation between two agents for a given goal. One of our immediate future goals is to develop a system for negotiated agents which rely on the available implementation of CR-Prolog.

Logic programming is used in [Chen *et al.*, 2006]. In their framework, two agents exchange answer sets to produce a common belief set. Their goal is coordinating belief sets of two agents, and it has no mechanism of constructing new proposals. We use CR-Prolog and specify a way for computing new proposals as part of the response to a given proposal.

Our work is similar to the work in [Sadri *et al.*, 2002; Sakama and Inoue, 2007] where abductive logic programming (ALP) is used to model negotiation. In our framework, the assumptions can be used in conjunction with predefined strategies, represented in the consistency restoring rules, to generate proposals whereas ALP uses only assumptions. The system in [Sakama, 2008] uses induction to construct proposals but does not consider preferences while ours does not use induction and considers preferences.

Finally, our work is similar in the spirit of approaches to argumentation-based negotiation (ABN) [Amgoud *et al.*, 2006; Kakas and Moraitis, 2006; Rahwan *et al.*, 2003] in that it considers explanations as a part of a proposal/response. The main difference between our work and ABN lies in our use of CR-Prolog, a non-monotonic logic, and ABN's logic is monotonic. Our framework does not compute explanations for accepting/rejecting a proposal in advance as in [Amgoud *et al.*, 2006] and allows negotiators to nonmonotonically changing their belief by incoming information.

Algorithm 1 Negotiation$(K_A,\ K_B,\ G)$

Ensure: Accept/Reject
  Initialize NQ := [ ] {Negotiation Queue}
  for each $\langle G, S\rangle \in \alpha(K_A, G)$ **do**
   InsertFrontQueue(NQ, $(K_A, \langle G, S, \emptyset\rangle, K_B)$)
  end for
  while NQ is not empty **do**
   (K_1, ω, K_2) := Extract(NQ) {get 1^{st} element}
   Compute $Z = \beta(K_2, \omega)$
   if $\langle \top, \emptyset, \emptyset\rangle \in Z$ **return** Accept **end if**
   if $\langle \bot, \emptyset, \emptyset\rangle \notin Z$ **then**
    for each $\omega \in Z$ **do**
     InsertFrontQueue(NQ, $(K_2, \omega, K_1 \ominus \omega.F)$) (Deepth First)
    end for
   end if
  end while
  return Reject

References

[Amgoud *et al.*, 2006] L. Amgoud, Y. Dimopoulos, and P. Moraitis. A unified and general framework for argumentation-based negotiation. In *AAMAS*, 1018–1025.

[Balduccini and Gelfond, 2003] M. Balduccini and M. Gelfond. Logic Programs with Consistency-Restoring Rules. *AAAI Spring Symposium*, 9–18, 2003.

[Balduccini, 2007] M. Balduccini. CR-MODELS: An Inference Engine for CR-Prolog. LPNMR 2007.

[Baral, 2003] C. Baral. *Knowledge Representation, reasoning, and declarative problem solving with Answer sets*. Cambridge University Press, Cambridge, MA, 2003.

[Chen *et al.*, 2006] W. Chen, M. Zhang, and N. Foo. Repeated negotiation of logic programs. In *NRAC.* , 2006.

[Gelfond and Leone, 2002] M. Gelfond and N. Leone. Logic programming and knowledge representation – the A-Prolog perspective. *AIJ*, 138(1-2):3–38, 2002.

[Gelfond and Lifschitz, 1991] M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing* 9:365–385, 1991.

[Kakas and Moraitis, 2006] A. Kakas and P. Moraitis. Adaptive agent negotiation via argumentation. *AAMAS*.

[Meyer *et al.*, 2004] T. Meyer, N. Foo, R. Kwok, and D. Zhang. Logical foundation of negotiation: outcome, concession and adaptation. In *AAAI*, 293–298, 2004.

[Rahwan *et al.*, 2003] I. Rahwan et al. Argumentation-based negotiation. *The Knowledge Eng. Review*, 18:343–375.

[Sadri *et al.*, 2002] F. Sadri, F. Toni, and P. Torroni. An abductive logic programming architecture for negotiating agents. In *JELIA*, 419–431, 2002.

[Sakama and Inoue, 2007] C. Sakama and K. Inoue. Negotiation by abduction and relaxation. In *AAMAS*, 1018–1025.

[Sakama, 2008] C. Sakama. Inductive Negotiation in Answer Set Programming. *DALT'08*, LNAI 5397, 143–160.