Negotiation by Abduction and Relaxation

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

Katsumi Inoue
National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-ku
Tokyo 101 8430, Japan
ki@nii.ac.jp

ABSTRACT

This paper studies a logical framework for automated negotiation between two agents. We suppose an agent who has a knowledge base represented by a logic program. Then, we introduce methods of constructing counter-proposals in response to proposals made by an agent. To this end, we combine the techniques of *extended abduction* in artificial intelligence and *relaxation* in cooperative query answering for databases. These techniques are respectively used for producing *conditional proposals* and *neighborhood proposals* in the process of negotiation. We provide a negotiation protocol based on the exchange of these proposals and develop procedures for computing new proposals.

Categories and Subject Descriptors

F.4.1 [**Mathematical Logic**]: Logic and constraint programming;; I.2.11 [**Distributed Artificial Intelligence**]: Multiagent systems

General Terms

Theory

Keywords

negotiation, extended abduction, relaxation, logic programming

1. INTRODUCTION

Automated negotiation has been received increasing attention in multi-agent systems, and a number of frameworks have been proposed in different contexts ([1, 2, 3, 5, 10, 11, 13, 14], for instance). Negotiation usually proceeds in a series of rounds and each agent makes a proposal at every round. An agent that received a proposal responds in two ways. One is a *critique* which is a remark as to whether or not (parts of) the proposal is accepted. The other is a *counter-proposal* which is an alternative proposal made in response to a previous proposal [13].

AAMAS'07 May 14–18 2007, Honolulu, Hawai'i, USA.

To see these proposals in one-to-one negotiation, suppose the following negotiation dialogue between a buyer agent B and a seller agent S. (B_i (or S_i) represents an utterance of B (or S) in the i-th round.)

B_1: I want to buy a personal computer of the brand b_1, with the specification of CPU:1GHz, Memory:512MB, HDD: 80GB, and a DVD-RW driver. I want to get it at the price under 1200 USD.
S_1: We can provide a PC with the requested specification if you pay for it by cash. In this case, however, service points are not added for this special discount.
B_2: I cannot pay it by cash.
S_2: In a normal price, the requested PC costs 1300 USD.
B_3: I cannot accept the price. My budget is under 1200 USD.
S_3: We can provide another computer with the requested specification, except that it is made by the brand b_2. The price is exactly 1200 USD.
B_4: I do not want a PC of the brand b_2. Instead, I can downgrade a driver from DVD-RW to CD-RW in my initial proposal.
S_4: Ok, I accept your offer.

In this dialogue, in response to the opening proposal B_1, the counter-proposal S_1 is returned. In the rest of the dialogue, B_2, B_3, S_4 are critiques, while S_2, S_3, B_4 are counter-proposals.

Critiques are produced by evaluating a proposal in a knowledge base of an agent. In contrast, making counter-proposals involves generating an alternative proposal which is more favorable to the responding agent than the original one. It is known that there are two ways of producing counter-proposals: extending the initial proposal or amending part of the initial proposal. According to [13], the first type appears in the dialogue: A: "I propose that you provide me with service X". B: "I propose that I provide you with service X if you provide me with service Z". The second type is in the dialogue: A: "I propose that I provide you with service Y if you provide me with service X". B: "I propose that I provide you with service X if you provide me with service Z". A negotiation proceeds by iterating such "give-and-take" dialogues until it reaches an agreement/disagreement. In those dialogues, agents generate (counter-)proposals by reasoning on their own goals or objectives. The objective of the agent A in the above dialogues is to obtain service X. The agent B proposes conditions to provide the service. In the process of negotiation, however, it may happen that agents are obliged to weaken or change their initial goals to reach a negotiated compromise. In the dialogue of

a buyer agent and a seller agent presented above, a buyer agent changes its initial goal by downgrading a driver from DVD-RW to CD-RW. Such behavior is usually represented as specific meta-knowledge of an agent or specified as negotiation protocols in particular problems. Currently, there is no computational logic for automated negotiation which has general inference rules for producing (counter-)proposals.

The purpose of this paper is to mechanize a process of building (counter-)proposals in one-to-one negotiation dialogues. We suppose an agent who has a knowledge base represented by a logic program. We then introduce methods for generating three different types of proposals. First, we use the technique of *extended abduction* in artificial intelligence [8, 15] to construct a *conditional proposal* as an extension of the original one. Second, we use the technique of *relaxation* in cooperative query answering for databases [4, 6] to construct a *neighborhood proposal* as an amendment of the original one. Third, combining extended abduction and relaxation, *conditional neighborhood proposals* are constructed as amended extensions of the original proposal. We develop a negotiation protocol between two agents based on the exchange of these counter-proposals and critiques. We also provide procedures for computing proposals in logic programming.

This paper is organized as follows. Section 2 introduces a logical framework used in this paper. Section 3 presents methods for constructing proposals, and provides a negotiation protocol. Section 4 provides methods for computing proposals in logic programming. Section 5 discusses related works, and Section 6 concludes the paper.

2. PRELIMINARIES

Logic programs considered in this paper are *extended disjunctive programs* (EDP) [7]. An EDP (or simply a *program*) is a set of rules of the form:

$$L_1 ; \cdots ; L_l \leftarrow L_{l+1}, \ldots, L_m, \mathit{not}\ L_{m+1}, \ldots, \mathit{not}\ L_n$$

($n \geq m \geq l \geq 0$) where each L_i is a positive/negative literal, i.e., A or $\neg A$ for an atom A, and *not* is *negation as failure* (NAF). $\mathit{not}\, L$ is called an *NAF-literal*. The symbol ";" represents disjunction. The left-hand side of the rule is the *head*, and the right-hand side is the *body*. For each rule r of the above form, $\mathit{head}(r)$, $\mathit{body}^+(r)$ and $\mathit{body}^-(r)$ denote the sets of literals $\{L_1, \ldots, L_l\}$, $\{L_{l+1}, \ldots, L_m\}$, and $\{L_{m+1}, \ldots, L_n\}$, respectively. Also, $\mathit{not\_body}^-(r)$ denotes the set of NAF-literals $\{\mathit{not}\, L_{m+1}, \ldots, \mathit{not}\, L_n\}$. A disjunction of literals and a conjunction of (NAF-)literals in a rule are identified with its corresponding sets of literals. A rule r is often written as $\mathit{head}(r) \leftarrow \mathit{body}^+(r), \mathit{not\_body}^-(r)$ or $\mathit{head}(r) \leftarrow \mathit{body}(r)$ where $\mathit{body}(r) = \mathit{body}^+(r) \cup \mathit{not\_body}^-(r)$. A rule r is *disjunctive* if $\mathit{head}(r)$ contains more than one literal. A rule r is an *integrity constraint* if $\mathit{head}(r) = \emptyset$; and r is a *fact* if $\mathit{body}(r) = \emptyset$. A program is *NAF-free* if no rule contains NAF-literals. Two rules/literals are identified with respect to variable renaming. A *substitution* is a mapping from variables to terms $\theta = \{x_1/t_1, \ldots, x_n/t_n\}$, where $x_1, \ldots, x_n$ are distinct variables and each t_i is a term distinct from x_i. Given a conjunction G of (NAF-)literals, $G\theta$ denotes the conjunction obtained by applying θ to G. A program, rule, or literal is *ground* if it contains no variable. A program P with variables is a shorthand of its *ground instantiation* $\mathit{Ground}(P)$, the set of ground rules obtained from P by substituting variables in P by elements of its Herbrand universe in every possible way.

The semantics of an EDP is defined by the *answer set semantics* [7]. Let Lit be the set of all ground literals in the language of a program. Suppose a program P and a set of literals $S (\subseteq \mathit{Lit})$. Then, the *reduct* P^S is the program which contains the ground rule $\mathit{head}(r) \leftarrow \mathit{body}^+(r)$ iff there is a rule r in $\mathit{Ground}(P)$ such that $\mathit{body}^-(r) \cap S = \emptyset$. Given an NAF-free EDP P, $Cn(P)$ denotes the smallest set of ground literals which is (i) *closed* under P, i.e., for every ground rule r in $\mathit{Ground}(P)$, $\mathit{body}(r) \subseteq Cn(P)$ implies $\mathit{head}(r) \cap Cn(P) \neq \emptyset$; and (ii) *logically closed*, i.e., it is either consistent or equal to Lit. Given an EDP P and a set S of literals, S is an *answer set* of P if $S = Cn(P^S)$. A program has none, one, or multiple answer sets in general. An answer set is *consistent* if it is not Lit. A program P is *consistent* if it has a consistent answer set; otherwise, P is *inconsistent*.

Abductive logic programming [9] introduces a mechanism of hypothetical reasoning to logic programming. An abductive framework used in this paper is the *extended abduction* introduced by Inoue and Sakama [8, 15]. An *abductive program* is a pair $\langle P, H \rangle$ where P is an EDP and H is a set of literals called *abducibles*. When a literal $L \in H$ contains variables, any instance of L is also an abducible. An abductive program $\langle P, H \rangle$ is *consistent* if P is consistent. Throughout the paper, abductive programs are assumed to be consistent unless stated otherwise. Let $G = L_1, \ldots, L_m, \mathit{not}\, L_{m+1}, \ldots, \mathit{not}\, L_n$ be a conjunction, where all variables in G are existentially quantified at the front and *range-restricted*, i.e., every variable in $L_{m+1}, \ldots, L_n$ appears in $L_1, \ldots, L_m$. A set S of ground literals *satisfies* the conjunction G if $\{ L_1\theta, \ldots, L_m\theta \} \subseteq S$ and $\{ L_{m+1}\theta, \ldots, L_n\theta \} \cap S = \emptyset$ for some ground instance $G\theta$ with a substitution θ. Let $\langle P, H \rangle$ be an abductive program and G a conjunction as above. A pair (E, F) is an *explanation* of an *observation* G in $\langle P, H \rangle$ if[1]

1. $(P \setminus F) \cup E$ has an answer set which satisfies G,
2. $(P \setminus F) \cup E$ is consistent,
3. E and F are sets of ground literals such that $E \subseteq H \setminus P$ and $F \subseteq H \cap P$.

When $(P \setminus F) \cup E$ has an answer set S satisfying the above three conditions, S is called a *belief set* of an abductive program $\langle P, H \rangle$ satisfying G (with respect to (E, F)). Note that if P has a consistent answer set S satisfying G, S is also a belief set of $\langle P, H \rangle$ satisfying G with respect to $(E, F) = (\emptyset, \emptyset)$. Extended abduction introduces/removes hypotheses to/from a program to explain an observation. Note that "normal" abduction (as in [9]) considers only introducing hypotheses to explain an observation. An explanation (E, F) of an observation G is called *minimal* if for any explanation (E', F') of G, $E' \subseteq E$ and $F' \subseteq F$ imply $E' = E$ and $F' = F$.

EXAMPLE 2.1. Consider the abductive program $\langle P, H \rangle$:

$$\begin{aligned} P: \quad & \mathit{flies}(x) \leftarrow \mathit{bird}(x), \mathit{not}\, \mathit{ab}(x)\,, \\ & \mathit{ab}(x) \leftarrow \mathit{broken\text{-}wing}(x)\,, \\ & \mathit{bird}(\mathit{tweety}) \leftarrow, \ \ \mathit{bird}(\mathit{opus}) \leftarrow, \\ & \mathit{broken\text{-}wing}(\mathit{tweety}) \leftarrow\ . \\ H: \quad & \mathit{broken\text{-}wing}(x)\,. \end{aligned}$$

The observation $G = \mathit{flies}(\mathit{tweety})$ has the minimal explanation $(E, F) = (\emptyset, \{\mathit{broken\text{-}wing}(\mathit{tweety})\})$.

[1]This defines *credulous* explanations [15]. *Skeptical* explanations are used in [8].

3. NEGOTIATION

3.1 Conditional Proposals by Abduction

We suppose an agent who has a knowledge base represented by an abductive program $\langle P, H \rangle$. A program P consists of two types of knowledge, *belief* B and *desire* D, where B represents objective knowledge of an agent, while D represents subjective knowledge in general. We define $P = B \cup D$, but do not distinguish B and D if such distinction is not important in the context. In contrast, abducibles H are used for representing permissible conditions to make a compromise in the process of negotiation.

DEFINITION 3.1. A *proposal* G is a conjunction of literals and NAF-literals:

$$L_1, \ldots, L_m, not\ L_{m+1}, \ldots, not\ L_n$$

where every variable in G is existentially quantified at the front and range-restricted. In particular, G is called a *critique* if $G = accept$ or $G = reject$ where *accept* and *reject* are the reserved propositions. A *counter-proposal* is a proposal made in response to a proposal.

DEFINITION 3.2. A proposal G is *accepted* in an abductive program $\langle P, H \rangle$ if P has an answer set satisfying G.

When a proposal is not accepted, abduction is used for seeking conditions to make it acceptable.

DEFINITION 3.3. Let $\langle P, H \rangle$ be an abductive program and G a proposal. If (E, F) is a minimal explanation of $G\theta$ for some substitution θ in $\langle P, H \rangle$, the conjunction G':

$$G\theta,\ E,\ not\ F$$

is called a *conditional proposal* (for G), where "E, *not* F" represents the conjunction: $A_1, \ldots, A_k, not\ A_{k+1}, \ldots, not\ A_l$ for $E = \{A_1, \ldots, A_k\}$ and $F = \{ A_{k+1}, \ldots, A_l \}$.

PROPOSITION 3.1. *Let $\langle P, H \rangle$ be an abductive program and G a proposal. If G' is a conditional proposal, there is a belief set S of $\langle P, H \rangle$ satisfying G'.*

PROOF. When $G' = G\theta,\ E,\ not\ F$, $(P \setminus F) \cup E$ has a consistent answer set S satisfying $G\theta$ and $E \cap F = \emptyset$. In this case, S satisfies $G\theta, E, not\ F$. □

A conditional proposal G' provides a minimal requirement for accepting the proposal G. If $G\theta$ has multiple minimal explanations, several conditional proposals exist accordingly. When $(E, F) \neq (\emptyset, \emptyset)$, a conditional proposal is used as a new proposal made in response to the proposal G.

EXAMPLE 3.1. An agent seeks a position of a research assistant at the computer department of a university with the condition that the salary is at least 50,000 USD per year. The agent makes his/her request as the proposal:[2]

$$G = assist(compt\_dept),\ salary(x),\ x \geq 50,000.$$

The university has the abductive program $\langle P, H \rangle$:

$$\begin{aligned} P:\quad & salary(40,000) \leftarrow assist(compt\_dept),\ not\ has\_PhD,\\ & salary(60,000) \leftarrow assist(compt\_dept),\ has\_PhD,\\ & salary(50,000) \leftarrow assist(math\_dept),\\ & salary(55,000) \leftarrow system\_admin(compt\_dept),\\ & employee(x) \leftarrow assist(x),\\ & employee(x) \leftarrow system\_admin(x),\\ & assist(compt\_dept)\,;\ assist(math\_dept)\\ & \qquad ;\ system\_admin(compt\_dept) \leftarrow,\\ H:\quad & has\_PhD, \end{aligned}$$

where available positions are represented by disjunction. According to P, the base salary of a research assistant at the computer department is 40,000 USD, but if he/she has PhD, it is 60,000 USD. In this case, $(E, F) = (\{has\_PhD\}, \emptyset)$ becomes the minimal explanation of $G\theta = assist(compt\_dept)$, $salary(60,000)$ with $\theta = \{\, x/60,000 \,\}$. Then, the conditional proposal made by the university becomes

$$assist(compt\_dept),\ salary(60,000),\ has\_PhD\,.$$

3.2 Neighborhood Proposals by Relaxation

When a proposal is unacceptable, an agent tries to construct a new counter-proposal by weakening constraints in the initial proposal. We use techniques of *relaxation* for this purpose. Relaxation is used as a technique of *cooperative query answering* in databases [4, 6]. When an original query fails in a database, relaxation expands the scope of the query by relaxing the constraints in the query. This allows the database to return "neighborhood" answers which are related to the original query. We use the technique for producing proposals in the process of negotiation.

DEFINITION 3.4. Let $\langle P, H \rangle$ be an abductive program and G a proposal. Then, G is *relaxed* to G' in the following three ways:

Anti-instantiation: Construct G' such that $G'\theta = G$ for some substitution θ.

Dropping conditions: Construct G' such that $G' \subset G$.

Goal replacement: If G is a conjunction "G_1, G_2", where G_1 and G_2 are conjunctions, and there is a rule $L \leftarrow G_1'$ in P such that $G_1'\theta = G_1$ for some substitution θ, then build G' as $L\theta, G_2$. Here, $L\theta$ is called a *replaced literal*.

In each case, every variable in G' is existentially quantified at the front and range-restricted.

Anti-instantiation replaces constants (or terms) with fresh variables. Dropping conditions eliminates some conditions in a proposal. Goal replacement replaces the condition G_1 in G with a literal $L\theta$ in the presence of a rule $L \leftarrow G_1'$ in P under the condition $G_1'\theta = G_1$. All these operations generalize proposals in different ways. Each G' obtained by these operations is called a *relaxation* of G. It is worth noting that these operations are also used in the context of *inductive generalization* [12]. The relaxed proposal can produce new offers which are neighbor to the original proposal.

DEFINITION 3.5. Let $\langle P, H \rangle$ be an abductive program and G a proposal.

1. Let G' be a proposal obtained by anti-instantiation. If P has an answer set S which satisfies $G'\theta$ for some substitution θ and $G'\theta \neq G$, $G'\theta$ is called a *neighborhood proposal by anti-instantiation*.
2. Let G' be a proposal obtained by dropping conditions. If P has an answer set S which satisfies $G'\theta$ for some substitution θ, $G'\theta$ is called a *neighborhood proposal by dropping conditions*.

[2]For notational convenience, we often include mathematical (in)equations in proposals/programs. They are written by literals, for instance, $x \geq y$ by $geq(x, y)$ with a suitable definition of the predicate geq.

3. Let G' be a proposal obtained by goal replacement. For a replaced literal $L \in G'$ and a rule $H \leftarrow B$ in P such that $L = H\sigma$ and $(G' \setminus \{L\}) \cup B\sigma \neq G$ for some substitution σ, put $G'' = (G' \setminus \{L\}) \cup B\sigma$. If P has an answer set S which satisfies $G''\theta$ for some substitution θ, $G''\theta$ is called a *neighborhood proposal by goal replacement*.

EXAMPLE 3.2. (cont. Example 3.1) Given the proposal $G = assist(compt\_dept)$, $salary(x)$, $x \geq 50,000$,

- $G'_1 = assist(w)$, $salary(x)$, $x \geq 50,000$ is produced by substituting $compt\_dept$ with a variable w. As
$$G'_1\theta_1 = assist(math\_dept),\ salary(50,000)$$
with $\theta_1 = \{\, w/math\_dept \,\}$ is satisfied by an answer set of P, $G'_1\theta_1$ becomes a neighborhood proposal by anti-instantiation.
- $G'_2 = assist(compt\_dept)$, $salary(x)$ is produced by dropping the salary condition $x \geq 50,000$. As
$$G'_2\theta_2 = assist(compt\_dept),\ salary(40,000)$$
with $\theta_2 = \{\, x/40,000 \,\}$ is satisfied by an answer set of P, $G'_2\theta_2$ becomes a neighborhood proposal by dropping conditions.
- $G'_3 = employee(compt\_dept)$, $salary(x)$, $x \geq 50,000$ is produced by replacing $assist(compt\_dept)$ with $employee(compt\_dept)$ using the rule $employee(x) \leftarrow assist(x)$ in P. By G'_3 and the rule $employee(x) \leftarrow system\_admin(x)$ in P, $G''_3 = sys\_admin(compt\_dept)$, $salary(x)$, $x \geq 50,000$ is produced. As
$$G''_3\theta_3 = sys\_admin(compt\_dept),\ salary(55,000)$$
with $\theta_3 = \{\, x/55,000 \,\}$ is satisfied by an answer set of P, $G''_3\theta_3$ becomes a neighborhood proposal by goal replacement.

Finally, extended abduction and relaxation are combined to produce conditional neighborhood proposals.

DEFINITION 3.6. Let $\langle\, P, H \,\rangle$ be an abductive program and G a proposal.

1. Let G' be a proposal obtained by either anti-instantiation or dropping conditions. If (E, F) is a minimal explanation of $G'\theta (\neq G)$ for some substitution θ, the conjunction $G'\theta$, E, *not* F is called a *conditional neighborhood proposal by anti-instantiation/dropping conditions*.
2. Let G' be a proposal obtained by goal replacement. Suppose G'' as in Definition 3.5(3). If (E, F) is a minimal explanation of $G''\theta$ for some substitution θ, the conjunction $G''\theta$, E, *not* F is called a *conditional neighborhood proposal by goal replacement*.

A conditional neighborhood proposal reduces to a neighborhood proposal when $(E, F) = (\emptyset, \emptyset)$.

3.3 Negotiation Protocol

A *negotiation protocol* defines how to exchange proposals in the process of negotiation. This section presents a negotiation protocol in our framework. We suppose one-to-one negotiation between two agents who have a common ontology and the same language for successful communication.

DEFINITION 3.7. A proposal $L_1, ..., L_m, not\ L_{m+1}, ..., not\ L_n$ *violates* an integrity constraint $\leftarrow body^+(r), not\_body^-(r)$ if for any substitution θ, there is a substitution σ such that $body^+(r)\sigma \subseteq \{\, L_1\theta, \ldots, L_m\theta \,\}$, $body^-(r)\sigma \cap \{\, L_1\theta, \ldots, L_m\theta \,\} = \emptyset$, and $body^-(r)\sigma \subseteq \{\, L_{m+1}\theta, \ldots, L_n\theta \,\}$.

Integrity constraints are conditions which an agent should satisfy, so that they are used to explain why an agent does not accept a proposal.

A negotiation proceeds in a series of *rounds*. Each i-th round ($i \geq 1$) consists of a proposal G_1^i made by one agent Ag_1 and another proposal G_2^i made by the other agent Ag_2.

DEFINITION 3.8. Let $\langle\, P_1, H_1 \,\rangle$ be an abductive program of an agent Ag_1 and G_2^i a proposal made by Ag_2 at the i-th round. A *critique set* of Ag_1 (at the i-th round) is a set
$$CS_1^i(P_1, G_2^j) = CS_1^{i-1}(P_1, G_2^{j-1}) \cup \{\, r \mid r \text{ is an integrity constraint in } P_1 \text{ and } G_2^j \text{ violates } r \,\}$$
where $j = i - 1$ or i, and $CS_1^0(P_1, G_2^0) = CS_1^1(P_1, G_2^0) = \emptyset$.

A critique set of an agent Ag_1 accumulates integrity constraints which are violated by proposals made by another agent Ag_2. $CS_2^i(P_2, G_1^j)$ is defined in the same manner.

DEFINITION 3.9. Let $\langle\, P_k, H_k \,\rangle$ be an abductive program of an agent Ag_k and G^j a proposal, which is not a critique, made by any agent at the $j(\leq i)$-th round. A *negotiation set* of Ag_k (at the i-th round) is a triple $NS_k^i = (S_c^i, S_n^i, S_{cn}^i)$, where S_c^i is the set of conditional proposals, S_n^i is the set of neighborhood proposals, and S_{cn}^i is the set of conditional neighborhood proposals, produced by G^j and $\langle\, P_k, H_k \,\rangle$.

A negotiation set represents the space of possible proposals made by an agent. S_x^i ($x \in \{c, n, cn\}$) accumulates proposals produced by G^j ($1 \leq j \leq i$) according to Definitions 3.3, 3.5, and 3.6. Note that an agent can construct counter-proposals by modifying its own previous proposals or another agent's proposals. An agent Ag_k accumulates proposals that are made by Ag_k but are rejected by another agent, in the *failed proposal set* FP_k^i (at the i-th round), where $FP_k^0 = \emptyset$.

Suppose two agents Ag_1 and Ag_2 who have abductive programs $\langle\, P_1, H_1 \,\rangle$ and $\langle\, P_2, H_2 \,\rangle$, respectively. Given a proposal G_1^1 which is satisfied by an answer set of P_1, a negotiation starts. In response to the proposal G_1^i made by Ag_1 at the i-th round, Ag_2 behaves as follows.

1. If $G_1^i = accept$, an agreement is reached and negotiation ends in success.
2. Else if $G_1^i = reject$, put $FP_2^i = FP_2^{i-1} \cup \{G_2^{i-1}\}$ where $\{G_2^0\} = \emptyset$. Proceed to the step 4(b).
3. Else if P_2 has an answer set satisfying G_1^i, Ag_2 returns $G_2^i = accept$ to Ag_1. Negotiation ends in success.
4. Otherwise, Ag_2 behaves as follows. Put $FP_2^i = FP_2^{i-1}$.
   (a) If G_1^i violates an integrity constraint in P_2, return the critique $G_2^i = reject$ to Ag_1, together with the critique set $CS_2^i(P_2, G_1^i)$.
   (b) Otherwise, construct NS_2^i as follows.
   (i) Produce S_c^i. Let $\mu(S_c^i) = \{\, p \mid p \in S_c^i \setminus FP_2^i$ and p satisfies the constraints in $CS_1^i(P_1, G_2^{i-1})\}$. If $\mu(S_c^i) \neq \emptyset$, select one from $\mu(S_c^i)$ and propose it as G_2^i to Ag_1; otherwise, go to (ii).
   (ii) Produce S_n^i. If $\mu(S_n^i) \neq \emptyset$, select one from $\mu(S_n^i)$ and propose it as G_2^i to Ag_1; otherwise, go to (iii).
   (iii) Produce S_{cn}^i. If $\mu(S_{cn}^i) \neq \emptyset$, select one from $\mu(S_{cn}^i)$ and propose it as G_2^i to Ag_1; otherwise, negotiation ends in failure. This means that Ag_2 can make no counter-proposal or every counter-proposal made by Ag_2 is rejected by Ag_1.

In the step 4(a), Ag_2 rejects the proposal G_1^i and returns the reason of rejection as a critique set. This helps for Ag_1

in preparing a next counter-proposal. In the step 4(b), Ag_2 constructs a new proposal. In its construction, Ag_2 should take care of the critique set $CS_1^i(P_1, G_2^{i-1})$, which represents integrity constraints, if any, accumulated in previous rounds, that Ag_1 must satisfy. Also, FP_2^i is used for removing proposals which have been rejected. Construction of $S_x^i\,(x \in \{c, n, cn\})$ in NS_2^i is incrementally done by adding new counter-proposals produced by G_1^i or G_2^{i-1} to S_x^{i-1}. For instance, S_n^i in NS_2^i is computed as

$$S_n^i = S_n^{i-1} \cup \{\, p \mid p \text{ is a neighborhood proposal made by } G_1^i \,\} \cup \{\, p \mid p \text{ is a neighborhood proposal made by } G_2^{i-1} \,\},$$

where $S_n^0 = \emptyset$. That is, S_n^i is constructed from S_n^{i-1} by adding new proposals which are obtained by modifying the proposal G_1^i made by Ag_1 at the i-th round or modifying the proposal G_2^{i-1} made by Ag_2 at the $(i-1)$-th round. S_c^i and S_{cn}^i are obtained as well.

In the above protocol, an agent produces S_c^i at first, secondly S_n^i, and finally S_{cn}^i. This strategy seeks conditions which satisfy the given proposal, prior to neighborhood proposals which change the original one. Another strategy, which prefers neighborhood proposals to conditional ones, is also considered. Conditional neighborhood proposals are to be considered in the last place, since they differ from the original one to the maximal extent. The above protocol produces the candidate proposals in S_x^i for each $x \in \{c, n, cn\}$ at once. We can consider a variant of the protocol in which each proposal in S_x^i is constructed one by one (see Example 3.3).

The above protocol is repeatedly applied to each one of the two negotiating agents until a negotiation ends in success/failure. Formally, the above negotiation protocol has the following properties.

THEOREM 3.2. *Let Ag_1 and Ag_2 be two agents having abductive programs $\langle P_1, H_1 \rangle$ and $\langle P_2, H_2 \rangle$, respectively.*

1. *If $\langle P_1, H_1 \rangle$ and $\langle P_2, H_2 \rangle$ are function-free (i.e., both P_i and H_i contain no function symbol), any negotiation will terminate.*
2. *If a negotiation terminates with agreement on a proposal G, both $\langle P_1, H_1 \rangle$ and $\langle P_2, H_2 \rangle$ have belief sets satisfying G.*

PROOF. 1. When an abductive program is function-free, abducibles and negotiation sets are both finite. Moreover, if a proposal is once rejected, it is not proposed again by the function μ. Thus, negotiation will terminate in finite steps.

2. When a proposal G is made by Ag_1, $\langle P_1, H_1 \rangle$ has a belief set satisfying G. If the agent Ag_2 accepts the proposal G, it is satisfied by an answer set of P_2 which is also a belief set of $\langle P_2, H_2 \rangle$. □

EXAMPLE 3.3. Suppose a buying-selling situation in the introduction. A seller agent has the abductive program $\langle P_s, H_s \rangle$ in which P_s consists of belief B_s and desire D_s:

$$\begin{aligned}
B_s:\ & pc(b_1, 1G, 512M, 80G)\,;\, pc(b_2, 1G, 512M, 80G) \leftarrow, && (1)\\
& dvd\text{-}rw\,;\, cd\text{-}rw \leftarrow, && (2)\\
D_s:\ & normal_price(1300) \leftarrow \\
& \quad pc(b_1, 1G, 512M, 80G),\ dvd\text{-}rw, && (3)\\
& normal_price(1200) \leftarrow \\
& \quad pc(b_1, 1G, 512M, 80G),\ cd\text{-}rw, && (4)\\
& normal_price(1200) \leftarrow \\
& \quad pc(b_2, 1G, 512M, 80G),\ dvd\text{-}rw, && (5)\\
& price(x) \leftarrow normal_price(x),\ add_point, && (6)\\
& price(x * 0.9) \leftarrow \\
& \quad normal_price(x),\ pay_cash,\ not\ add_point, && (7)\\
& add_point \leftarrow, && (8)\\
H_s:\ & add_point,\ \ pay_cash.
\end{aligned}$$

Here, (1) and (2) represent selection of products. The atom $pc(b_1, 1G, 512M, 80G)$ represents that the seller agent has a PC of the brand b_1 such that CPU is 1GHz, memory is 512MB, and HDD is 80GB. Prices of products are represented as desire of the seller. The rules (3) – (5) are normal prices of products. A normal price is a selling price on the condition that service points are added (6). On the other hand, a discount price is applied if the paying method is cash and no service point is added (7). The fact (8) represents the addition of service points. This service would be withdrawn in case of discount prices, so *add_point* is specified as an abducible.

A buyer agent has the abductive program $\langle P_b, H_b \rangle$ in which P_b consists of belief B_b and desire D_b:

$$\begin{aligned}
B_b:\ & drive \leftarrow dvd\text{-}rw, && (9)\\
& drive \leftarrow cd\text{-}rw, && (10)\\
& price(x) \leftarrow, && (11)\\
D_b:\ & pc(b_1, 1G, 512M, 80G) \leftarrow, && (12)\\
& dvd\text{-}rw \leftarrow, && (13)\\
& cd\text{-}rw \leftarrow not\ dvd\text{-}rw, && (14)\\
& \leftarrow pay_cash, && (15)\\
& \leftarrow price(x),\ x > 1200, && (16)\\
H_b:\ & dvd\text{-}rw.
\end{aligned}$$

Rules (12) – (16) are the buyer's desire. Among them, (15) and (16) impose constraints for buying a PC. A DVD-RW is specified as an abducible which is subject to concession.

(1st round) First, the following proposal is given by the buyer agent:

$$G_b^1:\ pc(b_1, 1G, 512M, 80G),\ dvd\text{-}rw,\ price(x),\ x \le 1200.$$

As P_s has no answer set which satisfies G_b^1, the seller agent cannot accept the proposal. The seller takes an action of making a counter-proposal and performs abduction. As a result, the seller finds the minimal explanation $(E, F) = (\{\, pay\_cash \,\}, \{\, add\_point \,\})$ which explains $G_b^1\theta_1$ with $\theta_1 = \{\, x/1170 \,\}$. The seller constructs the conditional proposal:

$$G_s^1:\ pc(b_1, 1G, 512M, 80G),\ dvd\text{-}rw,\ price(1170),\ pay\_cash,\ not\ add\_point$$

and offers it to the buyer.

(2nd round) The buyer does not accept G_s^1 because he/she cannot pay it by cash (15). The buyer then returns the critique $G_b^2 = reject$ to the seller, together with the critique set $CS_b^2(P_b, G_s^1) = \{(15)\}$. In response to this, the seller tries to make another proposal which satisfies the constraint in this critique set. As G_s^1 is stored in FP_s^2 and no other conditional proposal satisfying the buyer's requirement exists, the seller produces neighborhood proposals. He/she relaxes G_b^1 by dropping $x \le 1200$ in the condition, and produces

$$pc(b_1, 1G, 512M, 80G),\ dvd\text{-}rw,\ price(x).$$

As P_s has an answer set which satisfies

$$G_s^2:\ pc(b_1, 1G, 512M, 80G),\ dvd\text{-}rw,\ price(1300),$$

the seller offers G_s^2 as a new counter-proposal.

(3rd round) The buyer does not accept G_s^2 because he/she cannot pay more than 1200USD (16). The buyer again returns the critique $G_b^3 = reject$ to the seller, together with the critique set $CS_b^3(P_b, G_s^2) = CS_b^2(P_b, G_s^1) \cup \{(16)\}$. The seller then considers another proposal by replacing b_1 with a variable w, G_b^1 now becomes

$$pc(w, 1G, 512M, 80G),\ dvd\text{-}rw,\ price(x),\ x \leq 1200.$$

As P_s has an answer set which satisfies

$$G_s^3:\ pc(b_2, 1G, 512M, 80G),\ dvd\text{-}rw,\ price(1200),$$

the seller offers G_s^3 as a new counter-proposal.

(4th round) The buyer does not accept G_s^3 because a PC of the brand b_2 is out of his/her interest and P_b has no answer set satisfying G_s^3. Then, the buyer makes a concession by changing his/her original goal. The buyer relaxes G_b^1 by goal replacement using the rule (9) in P_b, and produces

$$pc(b_1, 1G, 512M, 80G),\ drive,\ price(x),\ x \leq 1200.$$

Using (10), the following proposal is produced:

$$pc(b_1, 1G, 512M, 80G),\ cd\text{-}rw,\ price(x),\ x \leq 1200.$$

As $P_b \setminus \{\, dvd\text{-}rw \,\}$ has a consistent answer set satisfying the above proposal, the buyer proposes the conditional neighborhood proposal

$$G_b^4:\ pc(b_1, 1G, 512M, 80G),\ cd\text{-}rw,\ not\ dvd\text{-}rw,\ price(x),\ x \leq 1200$$

to the seller agent. Since P_s also has an answer set satisfying G_b^4, the seller accepts it and sends the message $G_s^4 = accept$ to the buyer. Thus, the negotiation ends in success.

4. COMPUTATION

In this section, we provide methods of computing proposals in terms of answer sets of programs. We first introduce some definitions from [15].

DEFINITION 4.1. Given an abductive program $\langle\, P, H \,\rangle$, the set UR of *update rules* is defined as:

$$\begin{aligned} UR \;=\; & \{\, L \leftarrow not\ \overline{L},\ \ \overline{L} \leftarrow not\ L \mid L \in H \,\} \\ \cup\; & \{\, +L \leftarrow L \mid L \in H \setminus P \,\} \\ \cup\; & \{\, -L \leftarrow not\ L \mid L \in H \cap P \,\}, \end{aligned}$$

where $\overline{L}$, $+L$, and $-L$ are new atoms uniquely associated with every $L \in H$. The atoms $+L$ and $-L$ are called *update atoms*.

By the definition, the atom $\overline{L}$ becomes true iff L is not true. The pair of rules $L \leftarrow not\ \overline{L}$ and $\overline{L} \leftarrow not\ L$ specify the situation that an abducible L is true or not. When $p(x) \in H$ and $p(a) \in P$ but $p(t) \notin P$ for $t \neq a$, the rule $+L \leftarrow L$ precisely becomes $+p(t) \leftarrow p(t)$ for any $t \neq a$. In this case, the rule is shortly written as $+p(x) \leftarrow p(x),\ x \neq a$. Generally, the rule becomes $+p(x) \leftarrow p(x),\ x \neq t_1, \ldots, x \neq t_n$ for n such instances. The rule $+L \leftarrow L$ derives the atom $+L$ if an abducible L which is not in P is to be true. In contrast, the rule $-L \leftarrow not\ L$ derives the atom $-L$ if an abducible L which is in P is not to be true. Thus, update atoms represent the change of truth values of abducibles in a program. That is, $+L$ means the introduction of L, while $-L$ means the deletion of L. When an abducible L contains variables, the associated update atom $+L$ or $-L$ is supposed to have exactly the same variables. In this case, an update atom is semantically identified with its ground instances. The set of all update atoms associated with the abducibles in H is denoted by UH, and $UH = UH^+ \cup UH^-$ where UH^+ (resp. UH^-) is the set of update atoms of the form $+L$ (resp. $-L$).

DEFINITION 4.2. Given an abductive program $\langle\, P, H \,\rangle$, its *update program* UP is defined as the program

$$UP = (P \setminus H) \cup UR.$$

An answer set S of UP is called *U-minimal* if there is no answer set T of UP such that $T \cap UH \subset S \cap UH$.

By the definition, U-minimal answer sets exist whenever UP has answer sets. Update programs are used for computing (minimal) explanations of an observation. Given an observation G as a conjunction of literals and NAF-literals possibly containing variables, we introduce a new ground literal O together with the rule $O \leftarrow G$. In this case, O has an explanation (E, F) iff G has the same explanation. With this replacement, an observation is assumed to be a ground literal without loss of generality. In what follows, $E^+ = \{\, +L \mid L \in E \,\}$ and $F^- = \{\, -L \mid L \in F \,\}$ for $E \subseteq H$ and $F \subseteq H$.

PROPOSITION 4.1. *([15]) Let $\langle\, P, H \,\rangle$ be an abductive program, UP its update program, and G a ground literal representing an observation. Then, a pair (E, F) is an explanation of G iff $UP \cup \{\, \leftarrow not\ G \,\}$ has a consistent answer set S such that $E^+ = S \cap UH^+$ and $F^- = S \cap UH^-$. In particular, (E, F) is a minimal explanation iff S is a U-minimal answer set.*

EXAMPLE 4.1. To explain the observation $G = flies(t)$ in the program P of Example 2.1, first construct the update program UP of P:[3]

$$\begin{aligned} UP:\quad & flies(x) \leftarrow bird(x), not\ ab(x), \\ & ab(x) \leftarrow broken\text{-}wing(x), \\ & bird(t) \leftarrow,\ \ bird(o) \leftarrow, \\ & broken\text{-}wing(x) \leftarrow not\ \overline{broken\text{-}wing(x)}, \\ & \overline{broken\text{-}wing(x)} \leftarrow not\ broken\text{-}wing(x), \\ & +broken\text{-}wing(x) \leftarrow broken\text{-}wing(x),\ x \neq t, \\ & -broken\text{-}wing(t) \leftarrow not\ broken\text{-}wing(t). \end{aligned}$$

Next, consider the program $UP \cup \{\, \leftarrow not\ flies(t) \,\}$. It has the single U-minimal answer set: $S = \{\, bird(t), bird(o), flies(t), flies(o), \overline{broken\text{-}wing(t)}, \overline{broken\text{-}wing(o)}, -broken\text{-}wing(t) \,\}$. The unique minimal explanation $(E, F) = (\emptyset, \{broken\text{-}wing(t)\})$ of G is expressed by the update atom $-broken\text{-}wing(t)$ in $S \cap UH^-$.

PROPOSITION 4.2. *Let $\langle\, P, H \,\rangle$ be an abductive program and G a ground literal representing an observation. If $P \cup \{\, \leftarrow not\ G \,\}$ has a consistent answer set S, G has the minimal explanation $(E, F) = (\emptyset, \emptyset)$ and S satisfies G.*

Now we provide methods for computing (counter-)proposals. First, conditional proposals are computed as follows.

input : an abductive program $\langle\, P, H \,\rangle$, a proposal G;
output : a set S_c of proposals.

If G is a ground literal, compute its minimal explanation (E, F) in $\langle\, P, H \,\rangle$ using the update program. Put "$G, E, not\ F$" in S_c. Else if G is a conjunction possibly containing variables, consider the abductive program

[3] t represents tweety and o represents opus.

$\langle P \cup \{ O \leftarrow G \}, H \rangle$ with a ground literal O. Compute a minimal explanation of O in $\langle P \cup \{ O \leftarrow G \}, H \rangle$ using its update program. If O has a minimal explanation (E, F) with a substitution θ for variables in G, put "$G\theta, E, not\ F$" in S_c.

Next, neighborhood proposals are computed as follows.

input : an abductive program $\langle P, H \rangle$, a proposal G;
output : a set S_n of proposals.

% neighborhood proposals by anti-instantiation;
Construct G' by anti-instantiation. For a ground literal O, if $P \cup \{ O \leftarrow G' \} \cup \{ \leftarrow not\ O \}$ has a consistent answer set satisfying $G'\theta$ with a substitution θ and $G'\theta \neq G$, put $G'\theta$ in S_n.

% neighborhood proposals by dropping conditions;
Construct G' by dropping conditions. If G' is a ground literal and the program $P \cup \{ \leftarrow not\ G' \}$ has a consistent answer set, put G' in S_n. Else if G' is a conjunction possibly containing variables, do the following. For a ground literal O, if $P \cup \{ O \leftarrow G' \} \cup \{ \leftarrow not\ O \}$ has a consistent answer set satisfying $G'\theta$ with a substitution θ, put $G'\theta$ in S_n.

% neighborhood proposals by goal replacement;
Construct G' by goal replacement. If G' is a ground literal and there is a rule $H \leftarrow B$ in P such that $G' = H\sigma$ and $B\sigma \neq G$ for some substitution σ, put $G'' = B\sigma$. If $P \cup \{ \leftarrow not\ G' \}$ has a consistent answer set satisfying $G''\theta$ with a substitution θ, put $G''\theta$ in S_n. Else if G' is a conjunction possibly containing variables, do the following. For a replaced literal $L \in G'$, if there is a rule $H \leftarrow B$ in P such that $L = H\sigma$ and $(G' \setminus \{L\}) \cup B\sigma \neq G$ for some substitution σ, put $G'' = (G' \setminus \{L\}) \cup B\sigma$. For a ground literal O, if $P \cup \{ O \leftarrow G'' \} \cup \{ \leftarrow not\ O \}$ has a consistent answer set satisfying $G''\theta$ with a substitution θ, put $G''\theta$ in S_n.

THEOREM 4.3. *The set S_c (resp. S_n) computed above coincides with the set of conditional proposals (resp. neighborhood proposals).*

PROOF. The result for S_c follows from Definition 3.3 and Proposition 4.1. The result for S_n follows from Definition 3.5 and Proposition 4.2. □

Conditional neighborhood proposals are computed by combining the above two procedures. Those proposals are computed at each round. Note that the procedure for computing S_n contains some nondeterministic choices. For instance, there are generally several candidates of literals to relax in a proposal. Also, there might be several rules in a program for the usage of goal replacement. In practice, an agent can prespecify literals in a proposal for possible relaxation or rules in a program for the usage of goal replacement.

5. RELATED WORK

As there are a number of literature on automated negotiation, this section focuses on comparison with negotiation frameworks based on logic and argumentation.

Sadri et al. [14] use abductive logic programming as a representation language of negotiating agents. Agents negotiate using common dialogue primitives, called *dialogue moves*. Each agent has an abductive logic program in which a sequence of dialogues are specified by a program, a dialogue protocol is specified as constraints, and dialogue moves are specified as abducibles. The behavior of agents is regulated by an *observe-think-act cycle*. Once a dialogue move is uttered by an agent, another agent that observed the utterance thinks and acts using a proof procedure. Their approach and ours both employ abductive logic programming as a platform of agent reasoning, but the use of it is quite different. First, they use abducibles to specify dialogue primitives of the form $tell(utterer, receiver, subject, identifier, time)$, while we use abducibles to specify arbitrary permissible hypotheses to construct conditional proposals. Second, a program pre-specifies a plan to carry out in order to achieve a goal, together with available/missing resources in the context of resource-exchanging problems. This is in contrast with our method in which possible counter-proposals are newly constructed in response to a proposal made by an agent. Third, they specify a negotiation policy inside a program (as integrity constraints), while we give a protocol independent of individual agents. They provide an operational model that completely specifies the behavior of agents in terms of agent cycle. We do not provide such a complete specification of the behavior of agents. Our primary interest is to mechanize construction of proposals.

Bracciali and Torroni [2] formulate abductive agents that have knowledge in abductive logic programs. To explain an observation, two agents communicate by exchanging integrity constraints. In the process of communication, an agent can revise its own integrity constraints according to the information provided by the other agent. A set IC' of integrity constraints *relaxes* a set IC (or IC *tightens* IC') if any observation that can be proved with respect to IC can also be proved with respect to IC'. For instance, $IC' : \leftarrow a, b, c$ relaxes $IC : \leftarrow a, b$. Thus, they use relaxation for weakening the constraints in an abductive logic program. In contrast, we use relaxation for weakening proposals and three different relaxation methods, anti-instantiation, dropping conditions, and goal replacement, are considered. Their goal is to explain an observation by revising integrity constraints of an agent through communication, while we use integrity constraints for communication to explain critiques and help other agents in making counter-proposals.

Meyer et al. [11] introduce a logical framework for negotiating agents. They introduce two different modes of negotiation: *concession* and *adaptation*. They provide rational postulates to characterize negotiated outcomes between two agents, and describe methods for constructing outcomes. They provide logical conditions for negotiated outcomes to satisfy, but they do not describe a process of negotiation nor negotiation protocols. Moreover, they represent agents by classical propositional theories, which is different from our abductive logic programming framework.

Foo et al. [5] model one-to-one negotiation as a one-time encounter between two extended logic programs. An agent offers an answer set of its program, and their mutual deal is regarded as a trade on their answer sets. Starting from the initial agreement set $S \cap T$ for an answer set S of an agent and an answer set T of another agent, each agent extends this set to reflect its own demand while keeping consistency with demand of the other agent. Their algorithm returns new programs having answer sets which are consistent with each other and keep the agreement set. The work is extended to repeated encounters in [3]. In their framework, two agents exchange answer sets to produce a common belief set, which is different from our framework of exchanging proposals.

There are a number of proposals for negotiation based

on *argumentation*. An advantage of argumentation-based negotiation is that it constructs a proposal with arguments supporting the proposal [1]. The existence of arguments is useful to convince other agents of reasons why an agent offers (counter-)proposals or returns critiques. Parsons et al. [13] develop a logic of argumentation-based negotiation among BDI agents. In one-to-one negotiation, an agent A generates a proposal together with its arguments, and passes it to another agent B. The proposal is evaluated by B which attempts to build arguments against it. If it conflicts with B's interest, B informs A of its objection by sending back its attacking argument. In response to this, A tries to find an alternative way of achieving its original objective, or a way of *persuading* B to drop its objection. If either type of argument can be found, A will submit it to B. If B finds no reason to reject the new proposal, it will be accepted and the negotiation ends in success. Otherwise, the process is iterated. In this negotiation processes, the agent A never changes its original objective, so that negotiation ends in failure if A fails to find an alternative way of achieving the original objective. In our framework, when a proposal is rejected by another agent, an agent can weaken or change its objective by abduction and relaxation. Our framework does not have a mechanism of argumentation, but reasons for critiques can be informed by responding critique sets.

Kakas and Moraitis [10] propose a negotiation protocol which integrates abduction within an argumentation framework. A proposal contains an offer corresponding to the negotiation object, together with supporting information representing conditions under which this offer is made. Supporting information is computed by abduction and is used for constructing conditional arguments during the process of negotiation. In their negotiation protocol, when an agent cannot satisfy its own goal, the agent considers the other agent's goal and searches for conditions under which the goal is acceptable. Our present approach differs from theirs in the following points. First, they use abduction to seek conditions to support arguments, while we use abduction to seek conditions for proposals to accept. Second, in their negotiation protocol, counter-proposals are chosen among candidates based on preference knowledge of an agent at meta-level, which represents policy under which an agent uses its object-level decision rules according to situations. In our framework, counter-proposals are newly constructed using abduction and relaxation. The method of construction is independent of particular negotiation protocols. As [2, 10, 14], abduction or abductive logic programming used in negotiation is mostly based on normal abduction. In contrast, our approach is based on extended abduction which can not only introduce hypotheses but remove them from a program. This is another important difference.

Relaxation and neighborhood query answering are devised to make databases cooperative with their users [4, 6]. In this sense, those techniques have the spirit similar to cooperative problem solving in multi-agent systems. As far as the authors know, however, there is no study which applies those technique to agent negotiation.

6. CONCLUSION

In this paper we proposed a logical framework for negotiating agents. To construct proposals in the process of negotiation, we combined the techniques of extended abduction and relaxation. It was shown that these two operations are used for general inference rules in producing proposals. We developed a negotiation protocol between two agents based on exchange of proposals and critiques, and provided procedures for computing proposals in abductive logic programming. This enables us to realize automated negotiation on top of the existing answer set solvers. The present framework does not have a mechanism of selecting an optimal (counter-)proposal among different alternatives. To compare and evaluate proposals, an agent must have preference knowledge of candidate proposals. Further elaboration to maximize the utility of agents is left for future study.

7. REFERENCES

[1] L. Amgoud, S. Parsons, and N. Maudet. Arguments, dialogue, and negotiation. In: *Proc. ECAI-00*, pp. 338–342, IOS Press, 2000.

[2] A. Bracciali and P. Torroni. A new framework for knowledge revision of abductive agents through their interaction. In: *Proc. CLIMA-IV, Computational Logic in Multi-Agent Systems*, LNAI 3259, pp. 159–177, 2004.

[3] W. Chen, M. Zhang, and N. Foo. Repeated negotiation of logic programs. In: *Proc. 7th Workshop on Nonmonotonic Reasoning, Action and Change*, 2006.

[4] W. W. Chu, Q. Chen, and R.-C. Lee. Cooperative query answering via type abstraction hierarchy. In: *Cooperating Knowledge Based Systems*, S. M. Deen ed., pp. 271–290, Springer, 1990.

[5] N. Foo, T. Meyer, Y. Zhang, and D. Zhang. Negotiating logic programs. In: *Proc. 6th Workshop on Nonmonotonic Reasoning, Action and Change*, 2005.

[6] T. Gaasterland, P. Godfrey, and J. Minker. Relaxation as a platform for cooperative answering. *Journal of Intelligence Information Systems* 1(3/4):293–321, 1992.

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

[8] K. Inoue and C. Sakama. Abductive framework for nonmonotonic theory change. In: *Proc. IJCAI-95*, pp. 204–210, Morgan Kaufmann.

[9] A. C. Kakas, R. A. Kowalski, and F. Toni, The role of abduction in logic programming. In: *Handbook of Logic in AI and Logic Programming*, D. M. Gabbay, et al. (eds), vol. 5, pp. 235–324, Oxford University Press, 1998.

[10] A. C. Kakas and P. Moraitis. Adaptive agent negotiation via argumentation. In: *Proc. AAMAS-06*, pp. 384–391, ACM Press.

[11] T. Meyer, N. Foo, R. Kwok, and D. Zhang. Logical foundation of negotiation: outcome, concession and adaptation. In: *Proc. AAAI-04*, pp. 293–298, MIT Press.

[12] R. S. Michalski. A theory and methodology of inductive learning. In: *Machine Learning: An Artificial Intelligence Approach*, R. S. Michalski, et al. (eds), pp. 83–134, Morgan Kaufmann, 1983.

[13] S. Parsons, C. Sierra and N. Jennings. Agents that reason and negotiate by arguing. *Journal of Logic and Computation*, 8(3):261–292, 1988.

[14] F. Sadri, F. Toni, and P. Torroni, An abductive logic programming architecture for negotiating agents. In: *Proc. 8th European Conf. on Logics in AI*, LNAI 2424, pp. 419–431, Springer, 2002.

[15] C. Sakama and K. Inoue. An abductive framework for computing knowledge base updates. *Theory and Practice of Logic Programming* 3(6):671–715, 2003.