Negotiation Based Framework for Attribute-Based Access Control Policy Evaluation

Edward Caprin and Yan Zhang
Artificial Intelligence Research Group
School of Computing, Engineering and Mathematics.
University of Western Sydney, Kingswood, Australia.
[e.caprin][y.zhang]@uws.edu.au

ABSTRACT

Attribute-Based Access Control (ABAC) involves the mapping of various characteristics of a principal, objects and the environment to authorisations. With ABAC well suited to web services which focus on user collaboration [2, 10] and the recent draft guidelines from NIST [4] there is a clear indication of growing popularity of the model. However, as noted by Sandhu [7] ABAC faces various challenges, some of which relate to policy evaluation. In this paper we present a novel framework for ABAC policy evaluation based on negotiations and formalised in *Answer Set Programming*.

1. INTRODUCTION

Attribute-Based Access Control (ABAC) involves the mapping of various characteristics of a principal, objects or the environment to authorisations. With ABAC well suited to web services which focus on user collaboration [2, 10] and the recent draft guidelines from NIST [4] there is a clear indication of growing popularity of the model. However, as noted by Sandhu [7] ABAC faces various challenges, some of which relate to policy evaluation.

Attributes are facts and details about some principal or the environment. As principals may consider certain attributes sensitive they may wish to dictate the circumstance under which they are revealed during the policy evaluation process using *attribute disclosure rules* [6]. In turn these rules must be taken into consideration during policy evaluation. This results in ABAC policy evaluation being almost adversarial in nature as the resource owner and requester have opposing goals. The owner, intuitively, wishes to maximise the number of attributes the requester reveals to establish trust. While on the other hand, the requester would wish to minimise the number of attributes revealed. Interestingly, this holds certain parallels to the typical buyer/seller scenario used to illustrate negotiations. As the seller wishes to maximise price while the buyer wishes to minimise.

Existing research on the evaluation of ABAC policies has suggested the use of first order logic, or variant, expression such as *Answer Set Programming* (ASP) [10]. With this in mind we propose a novel approach to ABAC policy evaluation based on the negotiation framework developed by Son and Sakama [8] and ASP. This paper serves as a formal introduction to our framework, its properties and provides comments on a prototype implementation.

2. ANSWER SET PROGRAMMING (ASP)

ASP is a form of declarative programming based on the *stable model semantics* well suited to representing domain specific knowledge [1]. In our framework ASP serves a dual purpose; 1) it provides both a syntactical and semantic base for a ABAC policy language and 2) allows for the use of off-the-shelf ASP solvers, such as DLV [5, 9], to perform certain computations.

Given the numerous subclasses of ASP programs [1] we limit our framework's policy language to *Normal Logic Programs with Nonmonotonic Negation* (NLP). These programs are a set of rules of the forms:

$$A_0 \leftarrow A_1, \cdots, A_m, not\ A_{m+1}, \cdots, not\ A_n. \qquad (1)$$

Where each A_i is an atom and reads as A_0 can safely assumed to be true if and only if $A_1, \cdots, A_m$ are true and $A_{m+1}, \cdots, A_n$ can safely be assumed to be false. For our purposes, A_0 is an authorisation or attribute, while A_1 to A_n denote the conditions where A_0 holds.

These programs are used in conjunction with the aforementioned solvers to generate *answer sets*, sets of conclusions which can be inferred from the program. For some program $\mathcal{P}$ we denote the collection of all answer sets by by $Ans(\mathcal{P})$.

3. PRELIMINARIES

For a set of atoms L and a atom $l \in L$, we denote the set of rules $\{\ l_i. \mid l_i \in L\ \}$ by $\{L \leftarrow\}$. $Goal(l)$ denotes the constraint $\leftarrow not\ l.$, while $Goal^-(l)$ denotes the constraint $\leftarrow l.$. For a set of rules R, $Head(R)$ is the set of all atoms in the rule heads. $Body(R)^+$ denotes all positive atoms in rule bodies in R, and $Body(R)^-$ denotes all negation as failure (NAF) atoms. Finally, $Body(R) = Body(R)^+ \cup Body(R)^-$ denoting the set of all atoms in rules in R.

For clarity throughout this paper we refer to the principals' participating in a negotiation relative to each other using the terms *agent* and *opponent*. For instance, consider the principals Alice and Bob. Alice refers to herself as the agent, while Bob is her opponent. Conversely, from Bob's perspective he is the agent while Alice is his opponent.

4. NEGOTIATION KNOWLEDGE BASE

Negotiation is an exchange of offers between intelligent agents to reach a mutually acceptable conclusion on some goal. In order to do this agents must have some source of information on which to base these offers. With the focus of this framework ABAC policy evaluation we exploit the policies themselves as a information source. Besides their policy an agent requires another set of information on which to base their offers; assumptions about their opponent's attributes. We formalise all this information as a 3-tuple called a *Negotiation Knowledge Base* (NKB).

Definition 1 (Negotiation Knowledge Base). *A 3-tuple $K = \langle \Pi, H^+, H^- \rangle$ is called an NKB where:*

- Π *is an ASP program representing a principals' policy which contains the agent's attributes along with their attribute disclosure and authorisation rules.*
- H^+ *is the set of atoms called* Positive Assumptions *which the agent safely assume to be true such that* $H^+ \cap Head(\Pi) = \emptyset$ *and* $H^+ \cap H^- = \emptyset$.
- H^- *is the set of atoms called* Negative Assumptions *which the agent safely assume to be false such that* $H^- \cap Head(\Pi) = \emptyset$ *and* $H^- \cap H^+ = \emptyset$.

Π is an ASP program containing rules of the form (1) which encodes and functions as a agent's policy. NKBs where $Body(\Pi) = \emptyset$ are said to contain a *trivial policy.*

Property 1 (Trivial Policy). *An NKB $\langle \Pi, H^+, H^- \rangle$ is said to contain a* trivial policy *if $Body(\Pi) = \emptyset$.*

Assumptions are attributes an agent believes, or hopes, their opponent may or may not have. These are divided into *positive* and *negative assumptions.* H^+ is the set of *positive assumptions.* These are the attributes the agent believes their opponent holds, derived from the atoms in $Body(\Pi)^+$, i.e. $A_1, \cdots, A_m$ in eq. (1). While H^-, the set of *negative assumptions* are attributes the agent believes their opponent does not hold. These are derived from the atoms in $Body(\Pi)^-$, i.e. $A_{m+1}, \cdots, A_n$ in eq. (1). $H = H^+ \cup H^-$ is the set of both positive and negative assumptions.

Negotiations are said to conclude *successfully* if the requester is granted their request, while the negotiation concludes *unsuccessfully* if they are not. We will specify the conditions under which a negotiation concludes either successfully or unsuccessful as they arise throughout this paper. First lets consider negotiations which involve trivial policies.

Interestingly, for NKBs which contain a trivial policy when presented with a request for some resource, denoted by the *goal* atom g, if $\Pi \models g$ then then the negotiation will always conclude successfully. Since assumptions are derived from $Body(\Pi)$ NKBs which contain a trivial policy also have a empty assumption set; $H^+ = \emptyset$, $H^- = \emptyset$.

On the other hand, NKBs containing non-trivial policies generate their assumptions independent of any particular opponent. As such, there is a chance that these assumptions to do reflect an opponent's actual attributes. NKBs where this case are said to be *misaligned.*

Property 2 (Misaligned NKB). *For the NKB of agent A, $K_A = \langle \Pi_A, H_A^+, H_A^- \rangle$ and the NKB of agent B, $K_B = \langle \Pi_B, H_B^+, H_B^- \rangle$ both of which contain non-trivial policies. We say that the NKBs K_A and K_B are* misaligned *if either of the following holds:*

$$\Pi_B \cup \{H_B^+ \leftarrow\} \not\models h_A^+, \forall h_A^+ \in H_A^+ \quad (2)$$

$$\Pi_A \cup \{H_A^+ \leftarrow\} \not\models h_B^+, \forall h_B^+ \in H_B^+ \quad (3)$$

Property (2) states that K_A *misaligns with* K_B, indicating A's assumptions about B's attributes are incorrect. Similarly, (3) states that K_B *misaligns with* K_A which indicates B's assumptions about A are wrong. This property provides useful insights as to how a negotiation concludes.

Theorem 1. *For a negotiation between agents A and B over a request g from B to A, who have NKBs which contain non-trivial policies $K_A = \langle \Pi_A, H_A^+, H_A^- \rangle$ and $K_B = \langle \Pi_B, H_B^+, H_B^- \rangle$ respectively. If the NKB of A misaligns to B's and $\Pi_A \not\models g$ the negotiation will always conclude unsuccessfully.*

Proof. Suppose we have the NKBs of agents' A and B, respectively, $K_A = \langle \Pi_A, H_A^+, H_A^- \rangle$ and $K_B = \langle \Pi_B, H_B^+, H_B^- \rangle$. Since H_B^+ is derived from $Body(\Pi_B)^+$ agent B can derive all of its own attributes; $\Pi_B \cup \{H_B^+ \leftarrow\} \models Head(\Pi_B)$. Similarly, since H_A^+ is derived from $Body(\Pi_A)^+$ we can we write equation (2) as $\Pi_B \cup \{H_B^+ \leftarrow\} \not\models Body(\Pi_A)^+$. As such, for the NKB of A to misalign with B's, $Body(\Pi)^+ \cap Head(\Pi_B) = \emptyset$. Therefore, misalignment of the two NKBs implies that $Body(\Pi_A)^+$ and $Head(\Pi_B)$ are disjoint. Meaning that B holds no attributes which fulfil *any* of rules in A's policy.

If $\Pi_A \not\models g$ then A's policy does not contain a rule with an empty body which for g, leading to an unsuccessful negotiation. □

The condition $\Pi_A \not\models g$ takes into account that NKBs with non-trivial polices may contain rules with empty bodies, known as *facts.*

For the duration of this paper we will consider a negotiation scenario between two agents; Alice and Bob. Alice and Bob have the NKBs $K_{Alice} = \langle \Pi_{Alice}, H_{Alice}^+, H_{Alice}^- \rangle$ and $K_{Bob} = \langle \Pi_{Bob}, H_{Bob}^+, H_{Bob}^- \rangle$, respectively.

Example 1 (K_{Alice}). Π_{Alice} *contains:*

```
1 allow(A, view, "cats.jpg") :- memberof(A, "
    UoL Lacrosse"), not memberof(A, "UoL Tennis
    "), A != alice.
2 allow(A, view, "dogs.jpg") :- memberof(A, "
    UoL Lacrosse"), not memberof(A, "UoL Coffee
     Lovers"), A != alice.
3 enrolled(alice, "UoL", "Computer Science") :-
     enrolled(A, "UoL", "Computer Science"),
    not memberof(A, "UoL Robotics"), A != alice
    .
4 memberof(alice, "UoL Lacrosse").
```

In the above lines 1 and 2 are authorisation rules. Alice allows other to view the image "cats.jpg" if they are a member of the UoL Lacrosse club, but not a member of the UoL Tennis club. She also allows others to view the image "dogs.jpg" if they are a member of the Lacrosse club and not members of the Coffee club. In both rules Alice does not allow herself to be the one granted permission to view.

Line 3 is a attribute disclosure rule stating Alice is a enrolled in UoL Computer Science and will reveal this to other who are also enrolled in Computer Science and not a member of the Robotics club. Finally, line 4 is simply an attribute stating Alice is a member of the UoL Lacrosse club.

H^+_{Alice} *contains:*

```
memberof(bob, "UoL Lacrosse"), enrolled(bob,
  "UoL", "Computer Science"),
```

Where Alice assumes Bob is a member of the Lacrosse club and enrolled in UoL Computer Science. While H^-_{Alice} *contains:*

```
memberof(bob, "UoL Coffee Lovers"), memberof(
  bob, "UoL Tennis"), memberof(bob, "UoL
  Robotics")
```

Indicating Alice hopes Bob is not a member of the Coffee, Tennis or Robotics club. Note with all of these assumptions they are instances of the atoms in Body(Π_{Alice} *grounded w.r.t. Alice's current negotiation opponent, Bob.*

Example 2 (K_{Bob}). Π_{Bob} *contains:*

```
memberof(bob, "UoL Lacrosse") :- enrolled(A,
  "UoL", _), A != bob.
memberof(bob, "UoL Coffee Lovers") :-
  memberof(A, "UoL Lacrosse"), A != bob.
enrolled(bob, "UoL", "Computer Science").
```

Above shows the policy of agent Bob. He is a member of UoL Lacrosse club and is willing to reveal this fact to any enrolled at UoL. Bob is also a member of the UoL Coffee club, revealing this to other Lacrosse club members. The final line is an attribute indicating that Bob is enrolled at UoL in Computer Science.

H^+_{Bob} *contains:*

```
enrolled(alice, "UoL", "Mathematics"),
  enrolled(alice, "UoL", "Computer Science"),
memberof(alice, "UoL Lacrosse")
```

With Bob's positive assumptions we see in above in line 1 Bob limits which courses he considers with his "enrolled in any UoL course" to Mathematics and Computer Science. This is done purely for the practicality of the example. If we had included the full range of courses of a typical comprehensive university space would quickly become a concern.

$H^-_{Bob} = \emptyset$

Neither K_{Alice} or K_{Bob} contain trivial policies nor do they misalign to each other.

5. PROPOSALS

Proposals are offers from one agent to another over access to some resource, called the *goal*, and are derived from an agent's NKB.

Definition 2 (Proposal). *For the NKB of agent A,* $K_A = \langle \Pi, H^+, H^- \rangle$ *and an atom representing the* goal g *we denote minimal subsets of* H^+ *and* H^-*, respectively, by* H^+_* *and* H^-_**. Such that there exists the answer sets:*

- $M^+ \in Ans(\Pi \cup \{H^+_* \leftarrow\} \cup Goal(g))$
- $M^- \in Ans(\Pi \cup \{H^-_* \cup (M^+ \cap H^+) \leftarrow\} \cup Goal^-(g))$.

For which there is a set of atoms $S = (M^+ \cap H^+) \cup (M^- \cap H^-)$ *called* support *which leads to a conclusion on* g*. The tuple* $\langle g, S \rangle$ *is called a* proposal *for* g *by* A *w.r.t.* K*. We denote the set of all proposals for* g *by* A *w.r.t.* K *by* $\alpha(K, g)$*.*

Intuitively, S contains all assumptions which the agent needs to confirm with the opponent to reach a rational conclusion on the opponents request for g. It can be seen than S is derived from the answer sets M^+ and M^-. M^+ contains a minimal subset of H^+ for which g holds w.r.t. K. M^- contains a minimal subset of H^- where g does not hold w.r.t. K and the positive assumptions in M^+. As such, S contains the negative assumption which cause g to no longer hold when the positive assumptions which otherwise cause it to hold.

Crampton et al. [2] note that ABAC is particularly vulnerable to *attribute hiding attacks*. The definition of M^- and S results in the some interesting outcomes with respect to this. Ideally the opponent should not know which attributes are negative assumptions as they indicate which attributes can be exploited in the attack. Since the atoms in S are presented without the negation as failure operand it is not initially clear to the opponent which of these atoms are negative assumptions. Furthermore, rules which contain multiple NAF atoms in their body only one needs to hold for the rule to fail. Since M^- is a result of a minimal subset of H^- the number of negative assumptions revealed in a proposal is minimised.

Example 3 ($\alpha(K, g)$). *Consider* K_{Alice} *from example 1. For* $\alpha(K_{Alice}, allow(bob, view, \text{"}cats.jpg\text{"}))$ *when:*

$$H^+_{alice,*} = \{memberof(bob, \text{"}UoL\ Lacrosse\text{"})\}$$
$$H^-_{alice,*} = \{memberof(bob, \text{"}UoL\ Tennis\text{"})\}$$

$M^+ =$

```
{ memberof(alice, "UoL Lacrosse"), allow(bob,
   view, "cats.jpg"), allow(bob, view, "dogs.
  jpg"), allow(alice, view, "fish.jpg"),
  memberof(bob, "UoL Lacrosse") }
```

While $M^- =$

```
{ memberof(alice, "UoL Lacrosse"), allow(bob,
   view, "dogs.jpg"), allow(alice, view, "
  fish.jpg"), memberof(bob, "UoL Tennis"),
  memberof(bob, "UoL Lacrosse") }
```

As such:

$$\begin{aligned} S =& \{memberof(bob, \text{"}UoL\ Lacrosse\text{"})\} \cup \\ & \{memberof(bob, \text{"}UoL\ Tennis\text{"})\} \\ =& \{memberof(bob, \text{"}UoL\ Lacrosse\text{"}), \\ & memberof(bob, \text{"}UoL\ Tennis\text{"})\} \end{aligned}$$

Resulting in $\alpha(K_{Alice}, allow(bob, view, \text{"}cats.jpg\text{"}))$ *containing the proposal:*

```
< allow(bob, view, ``cats.jpg'' ), {
  memberof(bob, ''UoL Lacrosse"), memberof(
  bob, ``UoL Tennis") } >
```

6. RESPONDING TO PROPOSALS

When presented with a proposal $\langle g, S \rangle$ from their opponent an agent replies to it. As the proposal is effectively asking "do you hold attributes S for access to resource g?" the reply aims answer these questions. To do this we build on top of the existing definition of a proposal.

Firstly, we expand S to allow for an agent to "echo" the attribute atoms they hold. Secondly, we introduce a set of tuples called *conditional assumptions* to cater to attribute disclosure rules. Finally, we provide set of atoms $\mathcal{R}$ called *rejections* which contains the attributes the agent does not hold.

Attribute disclosure rules encode utterances such as "You want to know about this attribute? I need to know this before I can do this". In other words, an opponent can reply to a support atom buying stating it may only hold under certain conditions. These are represented as a *conditional assumption* which are defined as follows.

Definition 3 (Conditional Assumption). *Let $K_A = \langle \Pi, H^+, H^- \rangle$ be an NKB of agent A, K_B be the NKB of agent B and $Q_B = \langle g, S \rangle$ be a proposal for g by B w.r.t. K_B. For every atom $sg_i \in S$ there the are answer sets:*

- $M^+ \in Ans(\Pi \cup \{ H^+_* \leftarrow \} \cup Goal(sg_i))$
- $M^- \in Ans(\Pi \cup \{H^-_* \cup (M^+ \cap H^+) \leftarrow\} \cup Goal^-(sg_i))$.

If there exists no *M^+ or M^-, where $(M^+ \cap H^+) \cup (M^- \cap H^-) = \emptyset$, then sg_i is a* sub-goal *of Q_B w.r.t. K_A. Sub-goals form part of the tuple $\langle sg_i, SS_i \rangle$ called a* conditional assumption *where $SS_i = (M^+ \cap H^+) \cup (M^- \cap H^-)$ is a set of atoms which leads to a conclusions on sg_i called sub-support. The set of all conditional assumption to Q_B w.r.t. K_A is denoted by $\gamma(K_A, Q_B)$.*

Agent A attempts to generate a conditional assumption for every support atom in S by testing them against their attribute disclosure rules. Similar to proposals, definition 2, this is achieved by extracting values from the answer sets M^+ and M^-. However, in this case M^+ and M^- contain atoms pertaining to the support of the atom sg_i. If for some sg_i there exists a M^+ or M^- where $(M^+ \cap H^+) \cup (M^- \cap H^-) = \emptyset$ then sg_i does not have a conditional assumption.

Example 4 ($\gamma(K, Q)$). *Consider the NKB K_{Bob} from example 2 and let Q_{Alice} be the proposal from example 3. $\gamma(K_{Bob}, Q_{Alice})$ contains the following conditional assumptions:*

```
1 < memberof(bob, "UoL Lacrosse"), { enrolled(
    alice, "UoL", "Mathematics") } >
2 < memberof(bob, "UoL Lacrosse"), { enrolled(
    alice, "UoL", "Computer Science") } >
3 < memberof(bob, "UoL Lacrosse"), { enrolled(
    alice, "UoL", "Mathematics"), enrolled(
    alice, "UoL", "Computer Science") } >
```

Line 1 comes about when:

$$H^+_{Bob,*} = \{enrolled(alice, "UoL", "Mathematics")\}$$
$$H^-_{Bob,*} = \emptyset$$

Resulting in $M^+ =$

```
1 { enrolled(alice, "UoL", "Mathematics"),
    allow(bob, view, "cats.jpg"), allow(bob,
    view, "dogs.jpg"), memberof(bob, "UoL
    Lacrosse"), enrolled(bob, "UoL", "Computer
    Science") }
```

Such that, the SS for line 1:

$$SS_1 = \{enrolled(alice, "UoL", "Mathematics")\} \cup \emptyset$$

Similarly, for line 2 where :

$$H^+_{Bob,*} = \{enrolled(alice, "UoL", "Computer\ Science")\}$$
$$H^-_{Bob,*} = \emptyset$$

Results in $M^+ =$

```
1 { enrolled(alice, "UoL", "Computer Science"),
    allow(bob, view, "cats.jpg"), allow(bob,
    view, "dogs.jpg"), memberof(bob, "UoL
    Lacrosse"), enrolled(bob, "UoL", "Computer
    Science") }
```

Line 3 is where:

$$H^+_{Bob,*} = \{enrolled(alice, "UoL", "Mathematics"),$$
$$enrolled(alice, "UoL", "Computer\ Science")\}$$
$$H^-_{Bob,*} = \emptyset$$

We now extend the general form $\langle g, S \rangle$ of proposals to the 4-tuple $\langle g, \mathcal{S}, \mathcal{C}, \mathcal{R} \rangle$ called a *conditional proposal*.

Definition 4 (Conditional Proposal). *Let the NKB of agent A be $K_A = \langle \Pi, H^+, H^- \rangle$ and B, $Q_B = \langle g, S \rangle$ be a proposal from agent B. There exists the answer sets:*

- $M^+ \in Ans(\Pi \cup \{ H^+_* \leftarrow \} \cup Goal(g))$
- $M^- \in Ans(\Pi \cup \{H^-_* \cup (M^+ \cap H^+) \leftarrow\} \cup Goal^-(g))$.

If $S \cap H^- \neq \emptyset$ the 4-tuple $\langle g, \mathcal{S}, \mathcal{C}, \mathcal{R} \rangle$ is a conditional proposal *for Q_B w.r.t. K_A where:*

- $SG = \{ sg_i \mid \langle sg_i, SS_i \rangle \in \gamma(K_A, Q_B) \}$
- $\mathcal{S} = (M^+ \cap H^+) \cup (M^- \cap H^-) \cup (M^+ \cap S) \backslash SG$
- $\mathcal{C} = \{ \langle sg_i, SS_i \rangle \mid sg_i \in S, SS_i \subseteq \mathcal{S}, \langle sg_i, SS_i \rangle \in \gamma(K_A, Q_A) \}$
- $\mathcal{R} = \{ l \mid l \in S, l \notin M^+, l \notin SG \}$

SG is the set of all sub-goals for Q_B w.r.t. K_A. $\mathcal{S}$ is a set of atoms called support *which lead to a conclusion on g, excluding sub-goals. $\mathcal{C}$ is the set of conditional assumptions using by A to reach a conclusion on g w.r.t. Q_B. $\mathcal{R}$ is a set of atoms called* rejections *denoting any support atoms from S which A cannot support.*

However, if $S \cap H^- = \emptyset$ then the conditional proposal for Q_B w.r.t. K_A is $\langle \perp, \emptyset, \emptyset, \emptyset \rangle$ indicating negotiation failure. The set of all conditional proposals for Q_B w.r.t. K_A is denoted by $\beta(K_A, Q_B)$.

$\mathcal{S}$ is a direct extension of S from proposals. It retains its purpose of containing the assumption the agent needs to confirm with its opponent, while the addition of $(M^+ \cap S) \backslash SG$ results in the set also containing the attribute atoms from S which our agent holds, but excluding sub-goals. For each atom $sg_i \in S$ to have an associated conditional assumption in $\mathcal{C}$ it there must be a conditional assumption in $\langle sg_i, SS_i \rangle \in \gamma(K_A, Q_A)$ and $SS_i \subseteq \mathcal{S}$ indicating it was "active" in reaching a conclusion on g. $\mathcal{R}$ is a set of atoms from S which the agent does not hold.

Example 5 ($\beta(K, Q)$). *Consider the NKB K_{Bob} from example 2 and let Q_{Alice} be the proposal from example 3. $\beta(K_{Bob}, Q_{Alice})$ contains the conditional proposals:*

```
1 < allow(bob, view, "cats.jpg"), { enrolled(
    alice, "UoL", "Computer Science") }, { <
    memberof(bob, "UoL Lacrosse")], { enrolled(
    alice, "UoL", "Computer Science") } > }, [
    memberof(bob, "UoL Tennis") } >
2 < allow(bob, view, "cats.jpg"), { enrolled(
    alice, "UoL", "Mathematics") } , { <
    memberof(bob, "UoL Lacrosse"), { enrolled(
    alice, "UoL", "Mathematics") } > }, {
    memberof(bob, "UoL Tennis") } > }
3 < allow(bob, view, "cats.jpg"), { enrolled(
    alice, "UoL", "Computer Science") }, { <
    memberof(bob, "UoL Lacrosse"), { enrolled(
    alice, "UoL", "Computer Science") > }, {
    memberof(bob, "UoL Tennis") } >
4 < allow(bob, view, "cats.jpg") , { enrolled(
    alice, "UoL", "Mathematics") }, { <
    memberof(bob, "UoL Lacrosse"), { enrolled(
    alice, "UoL", "Mathematics") } > }, {
    memberof(bob, "UoL Tennis") } >
```

For all of the above conditional proposals

$$\mathcal{R} = \{memberof(bob, \text{"}UoL\ Tennis\text{"})\}$$

As there exists no $H^+_{Bob,*}$ *or* $H^-_{Bob,*}$ *where*

$$memberof(bob, \text{"}UoL\ Tennis\text{"}) \in M^+$$

Also for all of the above it can be seen that $\mathcal{C}$ *is a subset of* $\gamma(K_{Bob}, Q_{Alice})$ *(example 4) where* S *is a subset of some* SS_i *where* $\langle sg_i, SS_i\rangle \in \gamma(K_{Bob}, Q_{Alice})$

As the negotiation progresses, intuitively, the number of conditional assumptions being applied grows as more disclosure rules become involved. Just as with support atoms there will be conditional assumptions which an agent cannot support. As such, they require a method for rejecting them. This creates a problem for what we have defined to far as none of the formalisms take conditional assumptions into account.

Firstly, we need to consider how conditional assumptions are applied in the computations. So far assumptions have been applied as sets of rules based on minimal subsets of H^+ and H^- using $\{H^+_* \leftarrow\}$ and $\{H^-_* \leftarrow\}$ respectively. Using similar logic we use the conditional assumptions to generate rules. However, as conditional assumptions are a reply to an agent's assumptions this relationship needs to also be taken into account. We apply all assumptions as a set of rules derived from the set of assumptions H and conditional assumptions $\mathcal{C}$:

$$\begin{aligned} Assum(H, \mathcal{C}) = (\ \{H \leftarrow\} \backslash \{\ sg_i. \mid \langle sg_i, SS_i\rangle \in \mathcal{C}\ \}\ ) \cup \\ \{\ sg_i \leftarrow \bigwedge SS_i. \mid \langle sg_i, SS_i\rangle \in \mathcal{C}\ \} \end{aligned} \tag{4}$$

This set of rules results in the sub-goals of conditional assumptions taking priority over an agent's assumptions. This is done because we view conditional assumptions as partially confirmed assumption, and as such hold more "weight". We now redefine conditional assumptions to take conditional assumptions in a conditional proposal into account.

Definition 5 (Conditional Assumption (Extended)). *Let* $K_A = \langle \Pi, H^+, H^-\rangle$ *be an NKB of agent* A*,* K_B *be the NKB of agent* B *and* $Q_B = \langle g, \mathcal{S}, \mathcal{C}, \mathcal{R}\rangle$ *be a conditional proposal for* g *by* B *w.r.t.* K_B*. For every atom* $sg_i \in S$ *there are answer sets:*

- $M^+ \in Ans(\Pi \cup Assum(H^+_*, \mathcal{C}) \cup Goal(sg_i))$
- $M^- \in Ans(\Pi \cup Assum(H^-_* \cup (M^+ \cap H^+), \mathcal{C}) \cup Goal^-(sg_i))$.

If there exists no M^+ *or* M^-*, where* $(M^+ \cap H^+) \cup (M^- \cap H^-) = \emptyset$ *then* sg_i *is a sub-goal of* Q_B *w.r.t.* K_A*. Sub-goals form part of the tuple* $\langle sg_i, SS_i\rangle$ *called a* conditional assumption*.* $SS_i = (M^+ \cap H^+) \cup (M^- \cap H^-)$*, called sub-support which denotes a set of atoms which leads to a conclusion on* sg_i*. We denote the set of all conditional assumptions to* Q_B *w.r.t.* K_A *by* $\Gamma(K_A, Q_B)$*.*

Additionally, for any conditional assumption $\langle sg_i, SS_i\rangle \in \mathcal{C}$ *if there exists no* M^+ *or* M^-*, where* $(M^+ \cap H^+) \cup (M^- \cap H^-) = \emptyset$ *then* $\langle sg_i, SS_i\rangle$ *is also a conditional assumptions to* Q_B *w.r.t.* K_A *and is also in* $\Gamma(K_A, Q_B)$*.*

We finally define a *response* as a special case of a conditional proposal which has been derived w.r.t. an agent's NKB and a conditional proposal from their opponent.

Definition 6 (Response). *Let* $K_A = \langle \Pi, H^+, H^-\rangle$ *be the NKB of agent* A *and* $Q_B = \langle g, \mathcal{S}, \mathcal{C}, \mathcal{R}\rangle$ *be a conditional proposal from another agent* B*. As such there exists a NKB* $K_A \oplus Q_B = \langle \Pi, H^+\backslash\mathcal{R}, H^-\backslash\mathcal{R}\rangle$ *denoting a NKB of agent* A *updated w.r.t. information provided by* B*. There are two answer sets:*

- $M^+ \in Ans(\Pi \cup Assum(H^+_*, (\mathcal{C} \cap \Gamma(K_A \oplus Q_B, Q_B))) \cup Goal(g))$ *and;*
- $M^- \in Ans(\Pi \cup Assum(H^-_* \cup (M^+ \cap H^+), (\mathcal{C} \cap \Gamma(K_A \oplus Q_B, Q_B))) \cup Goal^-(g))$.

A's response to Q_B *w.r.t.* $K_A \oplus Q_B$ *is the conditional proposal* $\langle \mathcal{G}, \mathcal{S}', \mathcal{C}', \mathcal{F}\rangle$ *where:*

- $SG = \{\ sg_i \mid \langle sg_i, SS_i\rangle\ \in \Gamma(K_A \oplus Q_B, Q_B)\ \}$
- $\mathcal{S}' = (M^+ \cap H^+) \cup (M^- \cap H^-) \cup (M^+ \cap S) \backslash SG$
- $\mathcal{C}' = \{\ \langle sg_i, SS_i\rangle \mid \langle sg_i, SS_i\rangle \in \Gamma(K_A \oplus Q_B, Q_B), SS_i \subseteq \mathcal{S}\ \} \cup \{\ \langle sg_i, SS_i\rangle \mid \langle sg_i, SS_i\rangle \in (\mathcal{C} \cap \Gamma(K_A \oplus Q_B, Q_B))\ \}$
- $\mathcal{F} = \{\ l \mid l \in S, l \notin M^+, l \notin SG\ \}$

However, A's response to Q_B *w.r.t.* $K_A \oplus Q_B$ *is the conditional proposal* $\langle \bot, \emptyset, \emptyset, \emptyset\rangle$ *if any of the following holds:*

- $\mathcal{S} \cap H^- \neq \emptyset$
- $SG \cap H^- \neq \emptyset$
- $H \backslash \mathcal{R} = \emptyset$

We denote the set of all responses to Q_B *w.r.t.* $K_A \oplus Q_B$ *by* $\beta(K_A \oplus Q_B, Q_B)$

The key difference between a response and a conditional proposal that a response considers information learnt from the opponent. $K_A \oplus Q_B$ is an NKB which takes into account the rejections of the opponent. By removing rejected attribute atoms from their H the agent avoids re-asking their opponent for attributes they have already rejected.

M^+ and M^+ are now computed using $\mathcal{C} \cap \Gamma(K_A \oplus Q_B, Q_B)$, a subset of $\mathcal{C}$ where agent A can agree with these conditional assumptions. SG is the set of all possible subgoals for Q_B w.r.t. $K_A \oplus Q_B$ while $\mathcal{S}'$ remains unchanged from the conditional proposal definition of S.

$\mathcal{C}'$ consist of two distinct parts; new conditional assumption and conditional assumptions which can carry over from $\mathcal{C}$. $\{\ \langle sg_i, SS_i\rangle \mid \langle sg_i, SS_i\rangle \in \Gamma(K_A \oplus Q_B, Q_B), SS_i \subseteq \mathcal{S}\ \}$

is the set of new conditional assumptions introduced to the negotiation. While, $\{ \langle sg_i, SS_i \rangle \mid \langle sg_i, SS_i \rangle \in (\mathcal{C} \cap \Gamma(K_A \oplus Q_B, Q_B)) \}$ are the conditional assumptions from $\mathcal{C}$ which the agent can accept. Finally, $\mathcal{F}$ is the of all rejected attribute atoms from $\mathcal{S}$.

If $\mathcal{S}$ contains any elements from H^-, there is a $\langle sg_i, SS_i \rangle \in \mathcal{C}$ where $sg_i \in H^-$ or $H \backslash \mathcal{R} = \emptyset$ the agent's response is the special conditional proposal $\langle \bot, \emptyset, \emptyset, \emptyset \rangle$ denoting an unsuccessful negotiation. This is because these three cases denote scenarios where a successful negotiation is no longer possible. If either $\mathcal{S} \cap H^- \neq \emptyset$ or $\{ \langle sg_i, SS_i \rangle \mid \langle sg_i, SS_i \rangle \in \mathcal{C}, sg_i \in H^- \} \neq \emptyset$ is true then the opponent has confirmed they hold a negative assumption. While $H \backslash \mathcal{R} = \emptyset$ indcates a scenario where the agent has run out of assumptions, so no new response can be generated.

THEOREM 2. *For any NKB $K = \langle \Pi, H^+, H^- \rangle$ which contains a non-trivial policy and any conditional proposal $Q = \langle g, \mathcal{S}, \mathcal{C}, \mathcal{R} \rangle$. If $\mathcal{R} = \emptyset$ then $K \oplus Q = K$.*

PROOF. Suppose we have the NKB $K = \langle \Pi, H^+, H^- \rangle$ and the conditional proposal $Q = \langle g, \mathcal{S}, \mathcal{C}, \mathcal{R} \rangle$. If $\mathcal{R} = \emptyset$ then:

$$\begin{aligned} K \oplus Q &= \langle \Pi, H^+ \backslash \emptyset, H^- \backslash \emptyset \rangle \\ &= \langle \Pi, H^+, H^- \rangle \\ &= K \end{aligned}$$

Therefore, $K \oplus Q = K$ when $\mathcal{R} = \emptyset$ $\square$

The above illustrates an interesting situation. Since the conditional proposal results in no change to the agent's NKB it is possible response resulting from this NKB does not contribute towards progress towards a conclusion on some goal. As such, we formal define response which do.

PROPERTY 3 (PRODUCTIVE RESPONSES). *Let the NKB of agent A be $K_A = \langle \Pi, H^+, H^- \rangle$ and $Q_B = \langle g, \mathcal{S}, \mathcal{C}, \mathcal{R} \rangle$ be a conditional proposal from agent B to A. We say that A's response to Q_B, $Q_A \in \beta(K_A, Q_B)$ is* productive *if either:*

1. $K_A \oplus Q_B \neq K_A$
2. $\mathcal{C} \subseteq \Gamma(K_A \oplus Q_B, Q_B)$
3. $Q_A = Q_B$
4. $\langle \bot, \emptyset, \emptyset, \emptyset \rangle \in \beta(K_A, Q_B)$

Productive responses are responses which "make progress" towards a conclusion on the goal. This progress is characterised by any one of the listed conditions. Condition 1) has it so Q_A is the product of an updated NKB, while 2) has the response accept all of the currently active conditional assumptions. Conditions 3) and 4) relate to negotiation success and failure, as we show later (section 7. Condition 3) denotes one condition of negotiation success, while 4) denote an unsuccessful negotiation.

EXAMPLE 6 ($\beta(K \oplus Q, Q)$). *Again considering our running example. Let Q_{Bob} be line 1 from example 5. $\beta(K_{Alice} \oplus Q_{Bob}, Q_{Bob})$ contains:*

```
1  <{allow(bob, view, "cats.jpg")}, {memberof(
     bob, "UoL Robotics"), enrolled(bob, "UoL
     ", "Computer Science")}, {<{memberof(bob,
      "UoL Lacrosse")}, {enrolled(alice, "UoL
     ", "Computer Science")}>, <{enrolled(
     alice, "UoL", "Computer Science")}, {
     memberof(bob, "UoL Robotics"), enrolled(
     bob, "UoL", "Computer Science")}>}, {}>
```

$$\begin{aligned} &K_{Alice} \oplus Q_{Bob} = \\ &\quad \langle \Pi_{Alice}, H^+_{Alice} \backslash \{memberof(bob, "UoL\ Tennis")\}, \\ &\qquad H^-_{Alice} \backslash \{memberof(bob, "UoL\ Tennis")\} \rangle \end{aligned}$$

Furthermore, Alice has been able to accept all of the conditional assumptions Bob introduced into the negotiation, including them in her response. She also introduces a new conditional assumption relating to her enrollment in the UoL Computer Science program.

7. NEGOTIATION

Finally, we define a negotiation. The basis of a negotiation in this framework is a "I go, you go" scenario where two agents exchange responses to reach a conclusion on some initial query.

DEFINITION 7 (NEGOTIATION). *Let agents A and B have the NKBs K_A and K_B, respectively. A negotiation over a request for g by B to A, starting with B, is a infinite sequence of conditional proposals $w_1, \cdots, w_n$ where:*

- $w_1 = \langle g, \emptyset, \emptyset, \emptyset \rangle$ *where* $\langle g, \emptyset, \emptyset, \emptyset \rangle \in \alpha(K_B, g)$
- $w_i = \langle g, \mathcal{S}_i, \mathcal{C}_i, \mathcal{F}_i \rangle$
- $w_{i+1} \in \beta(K_{i+1}, w_i)$ *where for every* $i > 1$*:*
  - $K_0 = K_A$ *and* $K_{2k+2} = K_{2k} \oplus w_{2k+1}$ *for* $k \geq 0$*; and*
  - $K_1 = K_B$ *and* $K_{2k+1} = K_{2k-1} \oplus w_2$ *for* $k > 0$.

We say that a negotiation has concluded unsuccessfully *when $w_i = \langle \bot, \emptyset, \emptyset, \emptyset \rangle$. However, the negotiation has concluded* successfully *when $w_i \in \beta(K_i, w_i)$ and $w_i \in \beta(K_{i+1}, w_i)$.*

A negotiation can either conclude successfully or unsuccessfully. Successful terminates occur when both agents can accept the same conditional proposal. While the negotiation is unsuccessful of $w_i = \langle \bot, \emptyset, \emptyset, \emptyset \rangle$, the special response denoting the agent cannot continue this negotiation.

PROPERTY 4 (PRODUCTIVE NEGOTIATIONS). *A negotiation consisting of the sequence $w_1, \cdots, w_n$ of responses is said to be a* productive negotiation *if every w_i, $1 \leq i \leq n$ is a productive response.*

THEOREM 3. *If a negotiation is productive, then it is finite.*

PROOF. For a negotiation to be finite it must conclude. As per definition 7 a negotiation can either conclude successfully or unsuccessfully. Consider the conditions under which a response can be production in definition 3. A successful negotiation where each w_i is productive under condition 1), such that the NKB:

$$\begin{aligned} K_{2k+2} &= K_{2k} \oplus w_{2k+1} \\ &= \langle \Pi_{2k}, H^+_{2k}, H^-_{2k} \rangle \oplus \langle g, \mathcal{S}_{2k+1}, \mathcal{C}_{2k+1}, \mathcal{R}_{2k+1} \rangle \quad (5) \\ &= \langle \Pi_{2k}, H^+_{2k} \backslash \mathcal{R}_{2k+1}, H^-_{2k} \backslash \mathcal{R}_{2k+1} \rangle \end{aligned}$$

Would eventually result in $H = \emptyset$, which as per definition 6, results in $\langle \bot, \emptyset, \emptyset, \emptyset \rangle \in \beta(K_{2k+2}, w_{2k+1})$, triggering negotiation failure. Therefore, a negotiation consisting of w_i's productive under condition 1) will eventually result in negotiation failure, making it finite.

Conversely, a productive negotiation consisting only of w_i's productive under condition 2) must conclude in negotiation success since all conditional assumptions are acceptable under both agents, hence no rejections.

If all the w_i's in a negotiation are productive under conditions 3) and 4), as per definition 3, then the negotiation must be finite as the conditions correspond to a successful or unsuccessful negotiation. □

8. IMPLEMENTATION

We have developed a prototype implementation written in Java. This prototype utilises the set of Java interface classes for the popular ASP solver DLV; DLVWrapper v4.2. Since our framework makes heavy use of set operations we take advantage of the standard set classes in Java. Algorithm 1 illustrates an implementation of definition 7. It begins by first collecting both the NKB of the agent and the opponent, in this case the opponent in the one making the query. Note that definition 7 specifies that a negotiation is infinite. However, for practicality the algorithm is limited to a finite number of rounds as denoted by the *RoundLimit* variable.

Algorithm 1: Negotiation Algorithm

Input : NKBs of two agents, a query and round limit
Output: Decision over the query

1 Let K_{Ag} be the NKB of the resource holder.
2 Let K_{Op} be the NKB of the resource requester.
3 Let Q be a conditional proposal denote a request for access to some resource.
4 Let *RoundLim* be the number of rounds this negotiation is restricted to.
5 $Q_{Op} = Q$
6 $OpAccept = false$
7 $round = 2$
8 **while** $round \neq RoundLim$ **do**
9 **if** $round$ **mod** $2 = 0$ **then**
10 K_{Ag} = OPlus(K_{Ag}, Q_{Op})
11 Q_{Ag} = Response(K_{Ag}, Q_{Op})
12 **if** $Q_{Ag} = Q_{Op}$ **and** $OpAccept = true$ **then**
13 Successful Negotiation
14 **else**
15 $OpAccept = false$
16 **if** $round$ **mod** $2 \neq 0$ **then**
17 K_{Op} = OPlus(K_{Op}, Q_{Ag})
18 Q_{Op} = Response(K_{Op}, Q_{Ag})
19 **if** $Q_{Op} = Q_{Ag}$ **then**
20 $OpAccept = true$
21 **else**
22 $OpAccept = false$
23 **if** $Q_{Ag} = \langle \bperp, \emptyset, \emptyset, \emptyset \rangle$ **or** $Q_{Op} = \langle \bperp, \emptyset, \emptyset, \emptyset \rangle$ **then**
24 Unsuccessful Negotiation
25 $round = round + 1$
26 Unsuccessful Negotiation

Negotiation follows an "I go, you go" approach where each principal updates their NKB using the OPlus() function (lines 10 and 17). The updated NKB is then passed to function Response( ) (lines 11 and 18), which serves as both an implementation of definition 6 and selects a response from the set of possible responses. We leave the details of a response selection algorithm as a implementation specific issue since the selection is dependent on goal of the implementation. For instance, our prototype selects the first response which can be produced by Response( ), which results in rounds that take less time to compute, but potentially more rounds.

Negotiation conclude successfully when during the agent's turn $Q_{Ag} = Q_{Op}$ and in the previous round the opponent sets their *OpAccept* flag. Otherwise, *OpAccept* is reset to false. Unsuccessful negotiations can be triggered any round if $Q_{Ag} = \langle \bperp, \emptyset, \emptyset, \emptyset \rangle$ or $Q_{Op} = \langle \bperp, \emptyset, \emptyset, \emptyset \rangle$ (line 23) or the round limit is reached.

8.1 Experiments

To the best of our knowledge there is no standard set of access control policies to evaluate our prototype's performance. As such we develop our own set of experiments based on the Alice and Bob examples used throughout this paper and on the examples presented by Son and Sakama [8].

All of these experiment have been run on computer of the following specifications; Intel Core i7 2.9GHz, 8GB RAM MacBook Pro running OSX 10.9.2, Java RE 1.6.0_37 and DLV release (2012-7-12). In these experiments we measure the average CPU time taken for a negotiation to reach a conclusion on some query w.r.t. the NKBs of the agent and opponent.

We measure the time taken of each negotiation by calling the Java Main.getCpuTime( ) before and after a negotiation, taking the difference. An average time is taken by repeating the same query and NKB combination 1000 times. We finally, convert the time reported by Main.getCpuTime( ) from nanoseconds to milliseconds for readability. To gauge the complexity of a experiment set take we use the ASP grounder *gringo* [3] to report the number of grounded rules in the program $\Pi \cup \{H^+ \leftarrow\}$ where Π and H^+ are the policy and positive assumptions of some principals' NKB.

8.2 Experiment Set: AnB

This set of experiments utilise the NKBs of Alice and Bob which we introduced in example 1 and 2. We apply 4 different queries to these NKBs, as shown in table 1. Each query was applied already knowing whether or not the negotiation would conclude successfully or unsuccessfully, as denoted by the S or U in the Outcomes column.

Query "allow(bob, view "cats.jpg")" is *from Bob to Alice*, concludes successfully and involves one conditional assumption. While query "enrolled(bob, "UoL", "Computer Science")" is *from Alice to Bob*, concludes successfully and involves no conditional assumptions. Query "allow(bob, view, "dogs.jpg")" is *from Bob to Alice*, concludes unsuccessfully because Bob holds one of Alice's negative assumptions. Query "allow(alice, view, "fish.jpg")" is *Alice to Bob* and concludes unsuccessfully because Bob holds no such rule. By grounding $\Pi_{Alice} \cup \{H^+_{Alice} \leftarrow\}$ we find the program contains 6 grounded rules, while $\Pi_{Bob} \cup \{H^+_{Bob} \leftarrow\}$ contains 8, giving a total of 14 grounded rules. As we can see in table 1 the queries which involved conditional assumptions took longer, in terms of time and rounds, than ones that did not. This is to be expected as per definition 6 for each response a set of conditional assumptions is also computed. However, in all cases the query was resolved within an acceptable time

Query	Outcome	Time (ms)	No. Rounds
allow(bob, view "cats.jpg")	S	263	8
enrolled(bob, "UoL", "Computer Science")	S	48	5
allow(bob, view, "dogs.jpg")	U	89	2
allow(alice, view, "fish.jpg")	U	6	2

Table 1: Alice and Bob Experiment Results.

frame.

8.3 Experiment Set: Son and Sakama Example

For this experiment set we derive our NKBs from the buyer/seller example presented by Son and Sakama [8]. However, since our policy language does not support rule preference we have taken liberties with the source material, but still capture the spirit of the example.

Example 7 (Seller NKB $\langle \Pi_{Seller}, H^{+}_{Seller}, H^{-}_{Seller} \rangle$). *Below we have the policy of the seller agent. Lines 1 to 3 have the agent grouping certain attributes to form type of customer. Lines 4 to 8 has the seller assigning various price levels to different types of customer. While in lines 9 to 11 these rules extract particular properties of different products. The remaining lines of this policy represent the seller product database using disjunctive rule heads to emulate preference. Though the use of disjunctive heads is at odds with our policy language definitions, definition 1, we make the concession in this specific case to as a technique to encode preference. Since* Disjunctive Logic Programs *(DLPs) are at least as complex as NLPs [5] this change in policy language could only result in less favourable results for our prototype.*

```
1  whole_sale_customer :- registered.
2  student_customer :- student.
3  senior_customer :- age(A), A >= 65.
4  high_pr.
5  low_pr :- senior_customer.
6  low_pr :- student_customer, good_credit.
7  low_pr :- student_customer, pay_cash.
8  lowest_pr :- whole_sale_customer, quantity(A)
     , A >= 100.
9  make( A ) :- product( A, _, _).
10 madeIn( A ) :- product( _, A, _ ).
11 colour( A ) :- product( _, _, A ).
12 product("Top Lacrosse", "France", yellow) ∨
     product("Lacrosse Tech", "Austria", blue)
     ∨ product("Ball-o-Rama", "New Zealand",
     yellow) :- high_pr, not low_pr.
13 product("Lacrosse Tech", "Austria", blue) :-
     low_pr, not lowest_pr.
14 product("Ball-o-Rama", "New Zealand", yellow)
     :- lowest_pr.
```

As with our previous examples assumptions are provided manually. However, this examples illustrates this approaches shortcommings. Firstly, this technique fails at capturing rules bodies such as "age(A), A >= 65" as it wold either require an infinite number of entires or, ideally, be represented using some aggregate operand. Similarly, with "quantity".

```
1  registered, student, age(65), good_credit,
     pay_cash, quantity(100)
```

Example 8 (Buyer NKB $\langle \Pi_{Buyer}, H^{+}_{Buyer}, H^{-}_{Buyer} \rangle$). *Here we have the Buyers NKB.* $\Pi_{Buyer} =$

```
1  age(25). student. pay_cash. quantity(1).
2  sale :- make( "Top Lacrosse" ), madeIn( "
     France" ), not color( blue ), high_pr.
3  sale :- make( "Lacrosse Tech" ), not madeIn(
     "Australia" ), low_pr.
4  sale :- make( "Ball-o-Rama" ), color(
     tangerine ), low_pr.
5  sale :- make( "Econocrosse" ), lowest_pr.
```

Line 1 show the buyer's attributes. This buyer encodes their purchase preferences with the rules on lines 2 to 5, where they are willing to accept a "sale" if a certain combination of product attributes and price are met.

Below are the buyer's assumptions. Similar the Alice in the earlier experiment we distinguish negative assumptions by proceeding the atom with "not".

```
1  make("Top Lacrosse"), make("Lacrosse Tech"),
     make("Ball-o-Rama"),
2  make("Econocrosse"), madeIn("France"),
3  high_pr, low_pr, lowest_pr, color(tangerine),
4  not madeIn("Australia"), not color(blue)
```

Query "sale" is *Seller to the Buyer* and establishes which combinations a product attributes the Buyer will accept at certain price points and the conditions under which the Seller will offer these prices. Query "student" is also *Seller to Buyer* with the Seller attempting to establish if the Buyer is a student. Query "good_credit" has the Seller asking the Buyer if they have good credit, which they do not. While query "whole_sale_customer" has the Buyer attempting to work out if they are considered a whole sale customer, which they are not. By grounding $\Pi_{BuyerPB01} \cup \{H^{+}_{BuyerPB01} \leftarrow\}$ we find the program contains 14 grounded rules, while $\Pi_{SellerPB01} \cup \{H^{+}_{SellerPB01} \leftarrow\}$ contains 20, giving a total of 34 grounded rules.

We can see in table 2 what these results are consistent with those in table 1 where queries which involve conditional assumptions take longer than those that do not. While the time taken to resolve the query "sale" is significantly higher than any other query in these experiments, despite the number of rounds being comparable to the slowest query in the Alice and Bob experiment. Given that the agent and opponent in this example have conflicting rules relating to product price this higher execution time is to be expected as multiple conditional assumptions are involved with some of them being rejected by negotiating parties.

9. CONCLUSIONS AND FUTURE WORK

In conclusion, we have presented a framework for ABAC policy evaluation based on negotiation and formalised in ASP. Son and Sakama note that negotiations require reasoning with incomplete knowledge [8], a know strength of logic programming. With their work focused on the buyer/seller

Query	Outcome	Time (ms)	No. Rounds
sale	S	681	10
student	S	63	4
good_credit	U	10	2
whole_sale_customer	U	41	4

Table 2: Son and Sakama Example Experiment Results. 34 Rules

negotiation scenario their framework has feature we which are unsuitable in access control. For instance, through the use of weighted preferences Son and Sakama allow the goal of a negotiation to change [8]. This is logical in a price negotiation where the make of, for example, a washing machine can be supplemented by another, cheaper model. However, in access control a request for access to a specific resource, such as a picture of a cat is not easily replaced with access to dog pictures. As such, we do not allow for goals to change during negotiations. We do however, provide support for attribute disclosure rules.

Core to the work of Li et al. [6] is that during *Automated Trust Negotiation* some information about each principal may be considered sensitive, requiring attribute disclosure rules to define when they are revealed. While they [6] use these rules to construct *trust-target graphs* which are exchanged via a negotiation mediator we take the approach of partial rule sharing with our *conditional assumptions*.

We opt for partial rule sharing through conditional assumptions over the principals simply sharing their policy rules directly. This is to address concerns raised by Crampton et al. [2]. Crampton et al. note that ABAC systems are particularly vulnerable to attribute hiding attacks. As such, the policy rules themselves become important to protect as to not reveal which attributes an opponent should withhold to gain an advantage during negotiation.

Continued work on this framework will consider the development of an automated assumptions generation methodology along with integration of the framework with various ABAC language.

10. REFERENCES

[1] C. Baral. *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press, 1st edition, 2010.

[2] M. C. Crampton, J. PTaCL: A Language for Attribute-Based Access Control in Open Systems. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7215 LNCS:390–409, 2012.

[3] M. Gebser, R. Kaminski, B. Faufmann, M. Ostrowski, T. Schab, and S. Thiele. *A User's Guide to gringo, clasp, clingo and iclingo*. University of Potsdam, 2008.

[4] V. C. Hu, D. Ferraiolo, R. Kuhn, A. R. Friedman, A. J. Lang, M. M. Cogdell, A. Schnitzer, K. Sandlin, R. Miller, K. Scarfone, et al. Guide to attribute based access control (abac) definition and considerations (draft). *NIST Special Publication*, 800:162, 2013.

[5] N. Leone, G. Pfeifer, W. Faber, T. Eiter, G. Gottlob, S. Perri, and F. Scarcello. The dlv system for knowledge representation and reasoning. *ACM Trans. Comput. Logic*, 7(3):499–562, July 2006.

[6] J. Li, N. Li, and W. H. Winsborough. Automated trust negotiation using cryptographic credentials. *ACM Trans. Inf. Syst. Secur.*, 13(1):2:1–2:35, Nov. 2009.

[7] R. Sandhu. The Authorization Leap From Rights to Attributes: Maturation or Chaos? In *Proceedings of the 17th ACM symposium on Access Control Models and Technologies*, SACMAT '12, pages 69–70, New York, NY, USA, 2012. ACM.

[8] T. C. Son and C. Sakama. Negotiation using logic programming with consistency restoring rules. In *IJCAI*, pages 930–935, 2009.

[9] D. s.r.l. DLV System. http://www.dlvsystem.com, 2012.

[10] E. Yuan and J. Tong. Attributed Based Access Control (ABAC) for Web Services. In *Web Services, 2005. ICWS 2005. Proceedings. 2005 IEEE International Conference on*, pages 2 vol. (xxxiii+856), July 2005.