I2-D6

Automated Negotiation Mechanisms

Project number:	IST-2004-506779
Project title:	Reasoning on the Web with Rules and Semantics
Project acronym:	REWERSE
Document type:	D (deliverable)
Nature of document	R (report)
Dissemination level:	CO (confidential, only for REWERSE partners)
Document number:	IST506779/Naples/I2-D6/D/CO/a0.0
Responsible editor(s):	P. A. Bonatti
Contributing participants:	Naples, Vienna
Contributing workpackages:	I2
Contractual date of delivery:	28 February 2006
Actual date of delivery:	19 April 2006

Abstract

Before trust negotiation framework can be used in practice, researchers should give several kinds of *guarantees* including the following:

- Are negotiations going to *succeed* when the policies in principle allow it? The answer is not trivial, because in some cases the policies are protected as sensitive resources, and this may prevent peers from explaining exactly what is needed to complete the negotiation.
- Is it possible to minimize the sensitivity of disclosed information? When peers do not have a complete view of all the options, an optimal strategy might not exist.

In this report we begin to study the impact of policy protection on negotiation success by using an abstract framework, covering a wide spectrum of strategies and criteria for terminating negotiations.

Moreover, we start to study the problem of minimizing the sensitivity of the information disclosed during negotiations.

Keyword List

Strategy interoperability, Credential selection, Sensitivity minimization, Optimal strategies.

Automated Negotiation Mechanisms

P. A. Bonatti[1], **T. Eiter**[2], **M. Faella**[1]

[1] Università di Napoli Federico II
Email: {bonatti,mfaella}@na.infn.it

[2] Technische Universität Wien
Email: eiter@kr.tuwien.ac.at

19 April 2006

Abstract
Before trust negotiation framework can be used in practice, researchers should give several kinds of *guarantees* including the following:

- Are negotiations going to *succeed* when the policies in principle allow it? The answer is not trivial, because in some cases the policies are protected as sensitive resources, and this may prevent peers from explaining exactly what is needed to complete the negotiation.
- Is it possible to minimize the sensitivity of disclosed information? When peers do not have a complete view of all the options, an optimal strategy might not exist.

In this report we begin to study the impact of policy protection on negotiation success by using an abstract framework, covering a wide spectrum of strategies and criteria for terminating negotiations.

Moreover, we start to study the problem of minimizing the sensitivity of the information disclosed during negotiations.

Keyword List
Strategy interoperability, Credential selection, Sensitivity minimization, Optimal strategies.

Contents

1 Introduction

Trust negotiation is promising a high degree of interoperability and flexibility. However, it is difficult to predict the behavior of the spontaneous protocols that arise from the interaction of two agents that formulate requests and disclose information based on their own rule-based policies. Before such a framework can be used in practice, researchers should give several kinds of *guarantees* including the following:

- Is the negotiation going to *succeed* when the policies in principle allow it? The answer is not trivial, because in some cases the policies are protected as sensitive resources, and this may prevent peers from explaining exactly what is needed to complete the negotiation. The few papers devoted to this issue in the past are based on specific parametric strategies and do not really identify the general properties that allow for interoperability [Yu et al., 2001, Yu et al., 2003].
- Is it possible to minimize the sensitivity of the information disclosed during negotiations? Again, when policies are protected and peers do not have a complete view of all the options, an optimal strategy—in this respect—might not exist.

In this report we begin to study the impact of policy protection on negotiation success by using an abstract framework, covering a wide spectrum of strategies and criteria for terminating negotiations. We prove some positive results for what we call *cooperative peers* and a negative result for what we call *focused peers*.

Moreover, we start to study the problem of minimizing the sensitivity of the information disclosed during negotiations. In this report we focus on a simple special case, in which the policy of the server and all of its credentials are public. We call such interactions *public negotiations* although they are degenerate forms of negotiation with at most two steps. In these scenarios it is possible to minimize the sensitivity of disclosed information; the underlying assumptions are compatible with many real world web services, that wish to specify clearly their policies and wish to publish their certifications to attract customers. These assumptions generalize the so-called transparent and unilateral negotiations that have been proposed recently in the Semantic Web community. To our knowledge, this is the first analysis of the optimal disclosure problem.

We characterize the computational complexity of the sensitivity minimization problem for this scenario. Moreover, we show how to solve the minimization problem declaratively, by an embedding into Smodels, an answer set programming engine equipped with optimization facilities.

The report is organized as follows: First, in Section 2, we study the problem of optimal credential selection, in scenarios with public server policies and credentials. Then, in Section 3, we study the effect of policy protection on negotiation success. At the end of each section we discuss the section's results and future work.

2 Public negotiations and their complexity

Many commercial web services are likely to publish their policy entirely. Transparency may attract customers wishing to protect their privacy, and hence to minimize information disclosure—or the *sensitivity* of it. In fact, knowing at once all the possible ways of obtaining a service lets clients choose immediately the best option from their point of view. On the contrary, when

policies are disclosed incrementally, a client may release some credentials and discover later that some further condition cannot be satisfied, so the credentials have been disclosed for no purpose. In other cases, the client might eventually realize that some alternative, less sensitive set of credentials might have been used instead.

Moreover, a server may wish to publish at once all the credentials that a client might ask for, such as certifications proving nice properties of the server (e.g., quality certificates, membership to organizations such as the Better Business Bureau, etc.)

In this section we are studying the problem of minimizing the sensitivity of disclosed information when the server publishes all of its policy and all of its credentials (and hence the client has all of the information it needs to make an optimal choice).

The elements that the client is using to make an optimal choice are:

- The server's policy (a stratified Datalog program);
- The server's portfolio of credentials (a set of ground facts);
- The client's initial (service) request (a ground fact);
- The client's release policy (a stratified Datalog program with integrity constraints);
- The client's portfolio of credentials and declarations.

The client should find a subset of its portfolio that together with the server's policy entails the initial request. The credentials in this subset should be releasable according to the release policy. Moreover, the sensitivity of the set should be minimized.

Formally, this problem can be formulate as a sort of abduction problem. The initial request is the observable to be proved, and the portfolio is the set of abducibles. The program from which the observable should be proved can be the union of the server's rules/facts and of the client's policy. To check that disclosed credentials can actually be released according to the client's policy, one can introduce suitable integrity constraints such as:

$$\leftarrow \mathit{credential}(C),\ \mathit{not}\ \mathit{allow}(\mathit{release}, C).$$

Then the credential selection problem can be formalized as in the next section.

2.1 The formal credential selection problem and its complexity

The *credential selection problem* (CSEL) can be formulated as follows.

The problem's instances are tuples $\langle P, G, IC, C, \Sigma, sen\rangle$, where

- P is a finite, stratified logic program (the rules in the release policies of the "server" and of the "client");
- G is a goal in terms of a ground atom (modeling the authorization requested by the "client");
- IC is a finite set of integrity constraints (representing forbidden combinations of credentials); this is part of the release policy;

- C is a finite set of ground facts (the portfolio of credentials and declarations of the "client"),[1] and
- $sen : 2^C \to \Sigma$ is a *sensitivity aggregation function*, where Σ is a finite set (of *sensitivity values*) partially ordered by $\preceq$.

A *solution* for a CSEL as above is a set $S \subseteq C$ such that

1. $P \cup S \models G$,
2. $P \cup S \cup IC$ is consistent, and
3. $sen(S)$ is minimal among all S which satisfy 1. and 2.

In practice, examples of possible sensitivity aggregation functions are the maximum or the sum of numeric sensitivity levels assigned to each credential and declaration. In some cases (e.g. when a combination of credentials forms a quasi-identifier) *sen* may be greater than the sum of the individual sensitivity levels. For example, if zip codes have sensitivity level 2 and birth dates have sensitivity level 3, their combination represents a quasi-identifier and therefore the corresponding sensitivity level might be 10 rather than 5.

In the abstract framework, for the purpose of estimating the inherent complexity of the problem, we only assume *sen* to be a "monotonic" function computable in polynomial time. Monotonicity is to be intended as follows:

- if $S \subseteq S'$ then $sen(S) \preceq sen(S')$.

From now on we implicitly assume *sen* to satisfy the above axiom.

We next state some elementary results regarding the computational complexity of CSEL. For the purpose of these results, we consider the size of an instance of CSEL as the number of occurrences of symbols in $P \cup \{G\} \cup IC \cup C$ and in a string BC_{sen} representing a Boolean circuit which computes, given the characteristic vector of set $S \subseteq C$ of credentials for input, the value of $sen(S)$ (in a binary encoding). Notice that by just evaluating the circuit, $sen(S)$ can be computed in polynomial time, and in fact for any polynomial-time computable function $sen(S)$ such a circuit can be constructed in polynomial time.

Our first result characterizes the complexity of CSEL in the ground (propositional) case, in terms of a suitable complexity class. We note the following lemma.

Lemma 1. *Given a CSEL in which P is ground and $S \subseteq C$, deciding whether $P \cup S \models G$ and $P \cup S \cup IC$ is consistent is feasible in polynomial time.*

Proof. Indeed, $P \cup S$ stratified, and thus its unique perfect model, M, can be computed in polynomial time (see [Dantsin et al., 2001] for background). Checking $P \cup S \models G$ then amounts to checking $M \models G$, which is feasible in polynomial time, and testing whether $P \cup S \cup IC$ is consistent to checking whether $M \models IC$, which also feasible in polynomial time. □

Theorem 1. *If P and IC are ground then CSEL is* FNP//log*-complete. The theorem holds even if P and IC are positive.*

[1] From now on we mention only credentials, for the sake of readability. Implicitely, by "credential" we mean "credential *or declaration*". In the abstract framework adopted here, they are indistiguishable.

The complexity class FNP//OptP[$O(\log n)$] (for short, FNP//log) is from [Chen and Toda, 1995]. Intuitively, FNP//log contains all problems such that a solution for an instance I can be nondeterministically computed by a transducer in polynomial time, if the result $opt(I)$ of an NP optimization problem on I is known. Here, $opt(I)$ is an integer having $O(\log |I|)$ bits, where $|I|$ denotes the size of the input I; an NP *optimization problem* is the problem of computing the maximum value of any solution for an instance I, given that deciding $opt(I) \geq k$ is in NP, and recognizing solutions is polynomial.

For example, computing the largest set S of pairwise connected nodes in a given graph G (i.e., a maximum clique) is a problem in FNP//log (observe that different maximum cliques may exist). Indeed, computing the *size* of a maximum clique in G is an NP-optimization problem with $O(\log |G|)$ output bits, since testing whether a set S is a clique is easy (just check whether G has an edge between each pair of nodes in S), and deciding whether $opt(G) \geq k$ is in NP (guess a clique of size $\geq k$). Note, however, that this problem is not known to be FNP//log-complete.

Proof. of Theorem 1. *Membership.* We first have to show membership of the problem. To this end, we have (a) to find (a) an optimization problem on instances I of CSEL, such that the optimum value $opt(I)$ has $O(\log |I|)$ bits, where deciding $opt(I) \geq k$ is in NP, and (b) to construct, given $opt(I)$, a solution of CSEL nondeterministically in polynomial time.

As for (a), we take the problem to compute the lowest rank $r(s)$ of a sensitivity value $s \in \Sigma$ with respect to $\preceq$ such that $s = sen(S)$ for some set of credentials S which satisfies the conditions 1 and 2 of a solution of CSEL, converted to a maximization problem. The rank of $s \in \Sigma$, denoted $r(s)$, is the smallest integer i such that $s = s_i$ for some maximal chain $s_0 \prec s_1 \prec s_2 \ldots \prec s_m$ in Σ. Note that $r(s)$ has $\log |\Sigma|$ many bits, and is easily computable bottom up in Σ in polynomial time. In particular, $r(s) = 0$ iff there is no $s' \in \Sigma$ such that $s' \preceq s \wedge s \not\preceq s'$. We convert this to a maximization problem by considering $g(s) = r^* - r(s)$, where $r^* = \max_{s \in \Sigma} r(s)$.

Let $opt(I)$ be then the maximum over all $g(s)$ such that $s = sen(S)$ for some set of credentials $S \subseteq C$ which satisfies 1 and 2. Deciding whether $opt(I) \geq k$ is in NP, since we can guess an $S \subseteq C$ and verify in polynomial time that S satisfies conditions 1 and 2 of a solution, and also that $f(sen(S)) \leq k$. This establishes the part (a).

As for part (b), note that any $S \subseteq C$ which satisfies 1 and 2 of a solution of CSEL and has $f(sen(S)) = opt(I)$ must be a solution of CSEL (however, that for any solution S of CSEL, $f(sen(S)) = opt(I)$ is not true in general). Thus, given I and $opt(I)$, we can simply guess a solution S of CSEL and check in polynomial time that S satisfies conditions 1 and 2, and that $f(sen(S)) = opt(I)$.

Hardness. We show the FNP//log-hardness by a polynomial-time reduction from X-MAXIMAL MODEL: Given a Boolean formula $\varphi(Y)$ on atoms $Y = \{y_1, \ldots, y_m\}$ and a subset $X \subseteq \{y_1, \ldots, y_m\}$, compute the X-part of a model M of $\varphi(Y)$ such that $M \cap X$ is maximal, i.e., no model M' of $\varphi(Y)$ exists such that $M' \cap X \supset M \cap X$, where a model M is identified with the set of atoms that are mapped to true. Completeness of this problem for FNP//log is shown in [Chen and Toda, 1995].

We will reduce X-MAXIMAL MODEL to computing some solution of a CSEL instance in polynomial time in two parts, according to [Chen and Toda, 1995]. In Part 1, we will show that, for any instance $\varphi(Y)$ of X-MAXIMAL MODEL, an instance $f(\varphi(Y)) = \langle P, G, IC, C, \Sigma, sen \rangle$ of our problem is constructible in polynomial time, where it will be guaranteed that $f(\varphi(Y))$ has some solution.

In Part 2, we will show that, from every solution S of $f(\varphi(Y))$ and $\varphi(Y)$, some X-maximal model M of $\varphi(Y)$ can be constructed in time polynomial in the size of S and $\varphi(Y)$.

Without loss of generality, we assume that $\varphi(\mathcal{Y}) = C_1 \wedge \cdots C_k$ is a satisfiable conjunction of non-tautological clauses $C_i = l_{i,1} \vee l_{i_2} \vee l_{i_3}$, i.e., a CNF.

Part 1. We construct the following positive logic program P. Let $y'_1, \ldots, y'_k$, g and $c_1, \ldots, c_m$ be fresh propositional atoms.

- for each clause C_i and each variable y_j occurring in C_i, we set up the rule

$$c_i \leftarrow y_j,$$

  if y_j occurs positively in C_i, and for each variable y_j occurring in C_i, we set up the rule

$$c_i \leftarrow y'_j,$$

  if y_j occurs negatively in C_i.

- We add the rule

$$g \leftarrow c_1, \ldots, c_k.$$

The goal G is the atom g, C consists of $y_1, y'_1, \ldots, y_m, y'_m$, and IC consists of the constraints $\leftarrow y_j, y'_j$, for $j = 1 \ldots, m$.

Intuitively, $S \subseteq C$ satisfies conditions 1 and 2 of a solution, if it corresponds to the (consistent) partial truth assignment to the variables such that y_j is set true if $y_j \in S$ and y_j is set false if $y'_j \in S$.

Now we let $\Sigma = \{0, 1, \ldots, m\}$, with $\preceq$ the usual ordering of numbers, and define $sen(S) = |S \cap \{x' \mid x \in X\}|$ (recall that $X \subseteq Y$). Intuitively, the sensitivity of S is given by the number of negative literals $\neg y_j$ (represented by y') where $y_j \in X$, that belong to S.

It is then easy to see that S is a solution of this CSEL instance $f(\varphi(Y))$ if and only if $X \setminus \{x \mid x' \in S\}$ is an X-maximal model of $\varphi(Y)$ which has largest cardinality. Thus, $f(\varphi(Y))$ has some solution. Obviously, $f(\varphi(Y))$ is constructible in polynomial time from $\varphi(Y)$.

Part 2. Given any solution S of $f(\varphi(Y))$ and $\varphi(Y)$, we can compute the X-maximal model $X \setminus \{x \mid x' \in S\}$ of $\varphi(Y)$ easily.

Thus, summing up, solving X-MAXIMAL MODEL can be polynomially reduced to solving CSEL, which proves hardness. □

Theorem 2. *If P is positive and binary, i.e., each rule has at most one atom in the body, and in IC at most a constant number of credentials occur negatively, then the ground CSEL problem is solvable in logarithmic work space.*

Proof. Let P be a program of this form, and suppose that k credentials occur negatively in IC. The following holds: If $P \cup S \models G$ and $P \cup S \cup IC$ is satisfiable, then there is some subset $S' \subseteq S$ of size $|S'| \leq k + 1$ such that $P \cup S' \models G$ and $P \cup S' \cup IC$ is satisfiable as well.

Indeed, suppose for the moment that $k = 0$. Then, $P \cup S \models G$ iff there exists some $c \in S$ such that $P \cup \{c\} \models G$. This holds since inference of G from $P \cup S$ amounts to the reachability of some sink c from G in the usual dependency graph of $P \cup G$. Reachability of c from G can be decided in nondeterministic logarithmic space (NLOG), und thus $P \cup \{c\} \models G$ tested in NLOG. Furthermore, consistency of $P \cup \{c\} \cup IC$ is decidable in NLOG, since this can be reduced to non-reachability of a node in a graph, which is on co-NLOG, and since co-NLOG = NLOG,

also in NLOG. Thus, for a given singleton $S = \{c\}$, conditions 1 and 2 of a solution can be tested in nlog space. Cycling though all $c \in C$, we thus can determine the optimal such S in nlog space as well, which by axiom 1 for *sen* is then a solution for CSEL.

Now consider the case were $k > 0$ credentials occur negatively in IC. Then, like above either empty or singleton S is sufficient for $P \cup S \models G$; however, integrity constraints in IC containing negative literals might be violated by such S. However, at most k credentials (from those occurring negatively) need to be added to S to $S' \supseteq S$ in order to satisfy such integrity constraints. The number of such S' is polynomial if k is constant, and each S' occupies logarithmic space. Thus, by cyclic through polynomially many candidate S', and exploiting axiom 1 for *sen*, we can find a solution nondeterministically in logarithmic workspace. □

We remark that if arbitrarily many credentials may occur in IC under negation, then the problem is NP-hard even if P consists just of the fact G.

In the non-ground case, the complexity of the problem is expectedly higher. Here we are concerned with the case of function-free programs, i.e., datalog programs. For such programs, it is well-known that deciding inference of a ground atom is complete for exponential time (see [Dantsin et al., 2001]). The following result is thus not difficult to derive.

Theorem 3. *The unrestricted CSEL problem is solvable in exponential time, and EXPTIME-hard. The theorem holds even if P is positive and IC is empty.*

Proof. Membership. The membership of the problem in EXPTIME can be shown via a naive algorithm, which cycles through all subsets $S \subseteq C$ of C and, for each S checks whether conditions 1 and 2 of a solution hold, and maintains a current best such S (according to $sen(S)$) which is superseded if some better S is encountered. Checking 1 and 2 is, by naive grounding and Lemma 1 feasible in single exponential time. Thus, overall the algorithm runs in single exponential $\times$ single exponential = single exponential time.

The EXPTIME-hardness can be shown by a simple reduction from deciding whether $P \models A$ for some positive datalog programs P and an atom A. If we set in CSEL $\langle P, G, IC, C, \Sigma, sen\rangle$ the goal G to A, $IC = \emptyset$ and let $C = \{\}$, then some solution exists iff $P \models A$ holds. □

An important aspect of the EXPTIME hardness in the preceding result is that the arities of the predicates in P can become arbitrarily large. If the predicate aritities are bounded by a constant, then the complexity of the problem drops to classes within the polynomial hierarchy. Indeed, inference $P \models A$ of a ground atom A from a stratified datalog program is decidable in polynomial time with an NP oracle in this setting. The problem is thus solvable in polynomial time with an oracle for Σ_2^p: following a similar line as in the proof of Theorem 2, we can first compute the sensitivity value $s^* = sen(S^*)$ of some solution S^* using the Σ_2^p oracle, asking whether there exists some $S \subseteq C$ which satisfies 1 and 2 of a solution and has sensitivity $\preceq s$; given s^*, we can compute an optimal S^* step by step asking the oracle, for each credential $c_i \in C = \{c_1, \ldots, c_l\}$, whether there exists some solution S which does not contain c_i (if not, then we include c_i).

We also note that the complexity of the problem in the datalog case falls back to the one of the ground (propositional) case, if the number of variables in each rule is bounded by a constant, since then the grounding has polynomial size. Note also that this case is not straight subsumed by the case of bounded predicate arities.

2.2 Credential selection using Smodels

When the set of sensitivity levels Σ is a set of reals and *sen* is a member of a small set of simple functions, it is possible to embed the CSEL problem into the ASP system SMODELS.

SMODELS features an optimization facility to minimize the "weight" of stable models. Such weights are the sum of weights associated to individual ground atoms. This means that objective functions different from a sum must be encoded with some additional programming effort (currently we know how to encode only some objective functions). The objective function is expressed with a directive such as

$$\texttt{minimize}\ [A_1 = w_1, \ldots, A_n = w_n]$$

where $w_1, \ldots, w_n$ are the weights associated to atoms $A_1, \ldots, A_n$, respectively. For more details the reader is referred to [Simons et al., 2002].

Concerning the abduction problem, there are well-known encodings [Satoh and Iwayama, 1991, Bonatti, 2004] where the search space is obtained via cyclic recursion through negation.

The embedding of a CSEL instance $\langle P, G, IC, C, sen\rangle$ into SMODELS—illustrated in Figure 1—requires a new propositional symbol $\bar{c}$ for each atom $c \in C$. This is needed to create the search space: for each $S \subseteq C$ there exists a stable model M of this program such that $c \in M$ iff $c \in S$ and $\bar{c} \in M$ iff $c \in C \setminus S$.

Different aggregation functions *sen* require different sets of auxiliary rules A_{sen} and different optimization statements O_{sen}. In the following let $C = \{c_1, \ldots, c_n\}$.

- For $sen = \text{sum}$, let $A_{sen} = \emptyset$ and $O_{sen} = [c_1 = sen(\{c_1\}), \ldots, c_n = sen(\{c_n\})]$.
- For $sen = \max$, let A_{sen} be the following set of rules, for $i = 1, \ldots, n$, where the predicates *lev* and *maxlev* are new predicates:

$$\begin{aligned} lev(i) &\leftarrow c \qquad \text{where } c \in C \text{ and } sen(\{c\}) = i \\ lev(i) &\leftarrow lev(i+1) \\ maxlev(i) &\leftarrow lev(i),\ \texttt{not}\ lev(i+1) \end{aligned}$$

  and $O_{sen} = [maxlev(1) = 1, \ldots, maxlev(m) = m]$, where $m = \max\{sen(\{c\}) \mid c \in C\}$.

  Intuitively, for all sensitivity levels i below the maximal one, $lev(i)$ holds, and hence $maxlev(j)$ holds only for the maximal level j. The above rules can be easily generalized to partially ordered (possibly not numeric) sensitivity levels.

- If *sen* increases the sum of individual sensitivity levels when particular combinations of credentials (e.g., quasi-identifiers) are selected, then for each such combination $C_j = \{c'_1, \ldots, c'_k\}$ requiring increment d, let A_{sen} contain the rule

$$inc(j, d) \leftarrow c'_1, \ldots, c'_k$$

  and let $O_{sen} = [c_1 = sen(\{c_1\}), \ldots, c_n = sen(\{c_n\}), inc(j_1, d_1) = d_1, \ldots, inc(j_m, d_m) = d_m]$, where $inc(j_1, d_1), \ldots, inc(j_m, d_m)$ are the heads of the rules in A_{sen}.

From the results of [Simons et al., 2002, Satoh and Iwayama, 1991, Bonatti, 2004] it follows easily that:

Theorem 4. *For each instance I of CSEL, the least cost stable models of the embedding of Fig. 1 are in one-to-one correspondence with the solutions of I.*

$P \cup IC \cup \{c \leftarrow \texttt{not}\ \bar{c} \mid c \in C\} \cup \{\bar{c} \leftarrow \texttt{not}\ c \mid c \in C\} \cup \{\leftarrow \texttt{not}\ G\} \cup A_{sen}$
minimize O_{sen}

Figure 1: Embedding CSEL into SMODELS

2.3 Conclusions and future work

The problem of minimizing the sensitivity of disclosed information is hard even if all the information is available to the peer. However, we do not expect policies and portfolios to be large enough to create real problems; the optimization problem is to be solved on the clients, so it does not increase the computational load on the server. We have shown how to exploit a state-of-the-art anser set solver (SMODELS) to solve the problem. We are planning to build a random policy generator to carry out an experimental evaluation of this technique. We are also planning to exploit other engines, such as DLVHEX, to compare their efficiency with SMODEL's, and achieve greater flexibility in the definition of the objective function via the external function calls supported by DLVHEX.

We have not yet tackled the problem of minimizing information disclosure in more general scenarios, where the policy is disclosed incrementally. This will be the subject of further work.

Finally, it may be interesting to study approximate algorithms for the credential selection problem.

3 Negotiation strategies

In this section we abstract the negotiation framework of PROTUNE, by removing all details that are irrelevant to the study of the negotiations between two peers (in this report we do not address negotiations between 3 or more peers). The strategies for deciding which rules and credentials are to be disclosed at each step, are classified according to their abstract properties, for example:

- *Truthfulness*: the peer does not "invent" any information.
- *Monotonicity*: the more information is acquired from the other peer, the more is released to it.
- *Focussed*: the peer discloses only credentials and declarations that are explicitly asked for (no spontaneous disclosures).
- *Cooperative* (w.r.t. some class of scenarios): no other strategy in the same class of scenarios performs better (i.e. it leads to success in a larger number of cases).

We are interested in *interoperability*, i.e., the property of successfully completing all negotiations that would be successful if the entire policies were disclosed. This is needed to study the impact that policy protection has on negotiation success.

3.1 Definitions

We consider the following fixed sets of propositional items:

- a set of names Names;

- a set of resources $\mathsf{Resources}$;

- a set of abbreviations Abbr.

We assume that the above sets are finite or denumerable. We assume that $\mathsf{Resources}$ contains a special element Res. We set $\mathsf{Prop} = \mathsf{Names} \cup \mathsf{Resources} \cup \mathsf{Abbr}$.

A *rule* is a pair (A, B), where A (the *head* of the rule) is an element of Prop, while B (called the *body* of the rule) is a subset of $\mathsf{Resources} \cup \mathsf{Abbr}$. We denote the rule $(A, \{B_1, \ldots, B_k\})$ as $A \leftarrow B_1, \ldots, B_k$. A rule whose body is the empty set is called a *fact*. A fact $(A, \emptyset)$ is denoted by $A \leftarrow$ or simply by A, when this does not lead to ambiguities. Rules that are not facts are called *proper rules*. We denote Rules the set of all rules.

A *program* is a finite nonempty set of rules. In this context, the rule $A \leftarrow B_1, \ldots, B_k$ can be interpreted as follows: if $B_1, \ldots, B_k$ are all true, then:

$$\begin{cases} \text{the rule called } A \text{ can be released} & \text{if } A \in \mathsf{Names} \\ \text{resource } A \text{ is owned} & \text{if } A \in \mathsf{Resources} \\ \text{property } A \text{ is true} & \text{if } A \in \mathsf{Abbr} \end{cases}$$

A *message* is a set of rules. We denote Msgs the set of all messages. A *release strategy* is a function $S : \mathsf{Msgs}^* \to \mathsf{Msgs}$. Given a sequence of messages, a strategy prescribes the next "move" of the peer.

A *peer* is a triple $\mathcal{P} = (P, \lambda, S)$, where P is a program, $\lambda : P \to \mathsf{Names}$ is an injective function, assigning names to the rules in P, and S is a release strategy. The program P must satisfy the following *consistency rule*:

> If $A \leftarrow B_1, \ldots, B_n \in P$ and $A \in \mathsf{Names}$, then there is $r \in P$ such that $\lambda(r) = A$.

We assume that we are given two peers $\mathcal{P}_1$ and $\mathcal{P}_2$, where $\mathcal{P}_i = (P_i, \lambda_i, S_i)$. We assume that peer $\mathcal{P}_2$ contains rule $\mathsf{Res} \leftarrow$. The objective for peer $\mathcal{P}_1$ is to obtain the release of such rule. We assume that peers do not overload names. Formally, for all $\alpha \in \mathsf{Names}$ there is at most one $i \in \{1, 2\}$ such that $\lambda_i(r) = \alpha$ for some $r \in P_i$. Moreover, we assume that the following *independency condition* holds: if program P_i contains proper rule $A \leftarrow B_1, \ldots, B_n$, then A does not appear in the body of any rule in P_{3-i}. As stated in the forthcoming Lemma 2, this assumption prevents proper rules released from a peer from causing the unlocking of new information in the other peer.

A *negotiation* is a finite sequence of messages $m_0 m_1 \ldots m_k$, such that for all $i \geq 0$, $m_i = S_{1+(i \mod 2)}(\sigma_{i-1})$, where σ_{i-1} denotes the prefix of σ up to the $i-1$-th message.

Given a negotiation σ and a peer index $i \in \{1, 2\}$, we denote $\mathsf{received}(\sigma, i)$ the set of rules received by peer $\mathcal{P}_i$ during σ. Formally, if $\sigma = m_0 m_1 \ldots m_k$, we have $\mathsf{received}(\sigma, 1) = m_1 \cup m_3 \cup \ldots m_{k-1+(k \mod 2)}$ and $\mathsf{received}(\sigma, 2) = m_0 \cup m_2 \cup \ldots m_{k-(k \mod 2)}$.

We define the following classes of peers:

- *Truthful.* For all negotiations σ, we have $S_i(\sigma) \subseteq P_i$. Intuitively, a peer only transmits information that is known to it.

- *Monotonic.* Given two negotiations σ and σ', if $\mathsf{received}(\sigma, i) \subseteq \mathsf{received}(\sigma', i)$, then $S_i(\sigma) \subseteq S_i(\sigma')$.

Termination. A *termination criterion* is a set of negotiations F. If a negotiation belongs to F, then it is considered terminated with failure. A negotiation σ is *successful* if the fact Res belongs to $\mathsf{received}(\sigma, 1)$. Clearly, a termination criterion F cannot contain any successful negotiation.

We are only interested in termination criteria that bound the length of negotiations. Formally, there is no infinite negotiation such that all of its prefixes are neither terminated nor successful.

Security. Given a negotiation σ and a peer $\mathcal{P}_i = (P_i, \lambda_i, S_i)$, we say that rule $r \in P_i$ is *unlocked* for $\mathcal{P}_i$ at σ if $\lambda(r)$ is provable using the rules in P_i plus the rules in the current negotiation σ. We denote $\mathsf{unlocked}(\sigma, i)$ the set of all unlocked rules for $\mathcal{P}_i$ at σ.

We say that peer $\mathcal{P}_i$ is *secure* if it only reveals unlocked rules. Formally, for all negotiations σ, $S_i(\sigma) \subseteq \mathsf{unlocked}(\sigma, i)$.

In the following, we restrict to truthful and secure peers.

Focus. Let σ be a negotiation and i a peer index. The set $\mathsf{relevant}(\sigma, i)$ is the smallest subset of P_i satisfying the following condition: For all $(A, B) \in P_i$, if A occurs in the body of a rule in $\mathsf{received}(\sigma, i) \cup \mathsf{relevant}(\sigma, i)$, then $(A, B) \in \mathsf{relevant}(\sigma, i)$.

Given a set $V \subseteq P_i$, we say that the peer $\mathcal{P}_i$ is *focused* w.r.t. V if, for all negotiations σ, we have that $S_i(\sigma) \subseteq \mathsf{relevant}(\sigma, i) \cup V$. Intuitively, a focused peer only reveals information that is either relevant or voluntary.

Interoperability. We define the *most liberal release strategy* to be the strategy that always transmits all the proper rules in the program, plus those facts that are currently releasable (e.g., whose name is provable in the current negotiation). We say that peers $\mathcal{P}_1$ and $\mathcal{P}_2$ are *interoperable* if there is a successful negotiation or there is no successful negotiation even when peers employ the most liberal release strategy.

Cooperation. Next, we want to formalize the assumption that, other things being equal, peers are interested in achieving a successful negotiation. To this purpose, we employ the game-theoretic notion of *domination*.

Consider a termination criterion F. Given two peers $\mathcal{Q}_1$ and $\mathcal{Q}_2$, let $val(\mathcal{Q}_1, \mathcal{Q}_2) = 1$ if they achieve a successfull negotiation and 0 otherwise. Given two peers $\mathcal{Q}_1 = (P, \lambda, S_1)$ and $\mathcal{Q}_2 = (P, \lambda, S_2)$ having the same program and name labeling, we say that $\mathcal{Q}_1$ is *dominated* by $\mathcal{Q}_2$ (w.r.t. F) if for all peers $\mathcal{P}$, $val(\mathcal{Q}_1, \mathcal{P}) \leq val(\mathcal{Q}_2, \mathcal{P})$, and there exists a peer $\mathcal{P}^*$ such that $val(\mathcal{Q}_1, \mathcal{P}^*) < val(\mathcal{Q}_2, \mathcal{P}^*)$. Intuitively $\mathcal{Q}_1$ is dominated by $\mathcal{Q}_2$ if $\mathcal{Q}_2$ is at least as good as $\mathcal{Q}_1$ in achieving successful negotiations, and it is definitely better in at least one case. Domination defines a strict (i.e. irreflexive) partial order on peers sharing the same program and name labeling.

We say that a peer is *undominated* (w.r.t. F) if it is not dominated by any peer in the same category. For instance, a focused peer is undominated if it is not dominated by any focused peer. We say that the peer $\mathcal{P}_i$ is *cooperative* (w.r.t. F) if it is undominated.

3.2 Termination Criteria

Consider the following family of termination criteria. For all $k > 0$, let

$$F_k = \left\{ \sigma = \sigma_0\sigma_1 \dots \sigma_n \mid \sigma_{n-k+1} \cup \sigma_{n-k+2} \cup \dots \cup \sigma_n \subseteq \bigcup_{i=0}^{n-k} \sigma_i \right\}.$$

Intuitively, F_k stipulates that a negotiation is failed as soon as the peers exchange k messages that bear no new information. A message that bears no new information is called a *vacuous* mesasge. We proceed to show some properties of peers that are cooperative w.r.t. a termination criterion in this family. For simplicity, we develop the arguments for $\mathcal{P}_1$.

3.2.1 Cooperation

For all $k > 0$, in order to be cooperative w.r.t. F_k, peers do not send vacuous messages that lead to immediate termination, unless they have no other choice. This property is expressed by the following theorem. For a negotiation $\sigma = m_0 m_1 \dots m_n$ and an integer $0 \leq j \leq n+1$, let $\mathsf{tail}(\sigma, j)$ the set of the last j messages in σ. Formally, $\mathsf{tail}(\sigma, j) = \{m_{n-j+1}, \dots, m_n\}$, where it is understood that $\mathsf{tail}(\sigma, 0) = \emptyset$.

Theorem 5. *Assume that $\mathcal{P}_1$ is cooperative w.r.t. F_k, for some $k > 0$. For all negotiations σ, if $S_1(\sigma) \subseteq \mathsf{received}(\sigma, 2)$ and all messages in $\mathsf{tail}(\sigma, k-1)$ are vacuous, then $\mathsf{unlocked}(\sigma, 1) \subseteq \mathsf{received}(\sigma, 2)$.*

Proof. Assume by contradiction that there is a negotiation σ such that $S_1(\sigma) \subseteq \mathsf{received}(\sigma, 2)$, but there is a rule $r \in \mathsf{unlocked}(\sigma, 1) \setminus \mathsf{received}(\sigma, 2)$. We show that $\mathcal{P}_1$ is dominated by another peer $\mathcal{P}_1' = (P_1, \lambda_1, S_1')$. The strategy S_1' behaves exactly like S_1, except that $S_1'(\sigma) = S_1(\sigma) \cup \{r\}$.

First, we have to check that, for all peers $\mathcal{P}_2$, $val(\mathcal{P}_1, \mathcal{P}_2) \leq val(\mathcal{P}_1', \mathcal{P}_2)$. Consider any peer $\mathcal{P}_2$; if $\mathcal{P}_1$ and $\mathcal{P}_2$ do not give rise to the negotiation σ, then $\mathcal{P}_1'$ behaves exactly like $\mathcal{P}_1$ and so, if $\mathcal{P}_1$ achieves a successful negotiation, so does $\mathcal{P}_1'$. Next, assume that $\mathcal{P}_1$ and $\mathcal{P}_2$ do give rise to the negotiation σ. Then, $val(\mathcal{P}_1, \mathcal{P}_2) = 0$ and certainly $\mathcal{P}_1'$ cannot do any worse than this.

Finally, we have to exhibit a peer $\mathcal{P}_2$ that prefers $\mathcal{P}_1'$ over $\mathcal{P}_1$. We define $\mathcal{P}_2$ so that, together with $\mathcal{P}_1$, it gives rise to σ. Moreover, $\mathcal{P}_2$ releases Res as soon as it receives r. It follows from the construction that $val(\mathcal{P}_1, \mathcal{P}_2) = 0$ and $val(\mathcal{P}_1', \mathcal{P}_2) = 1$. □

Among the consequences of Theorem 5 is the fact that peers that are cooperative w.r.t. F_1 do not send vacuous messages unless they have no other choice. The following examples show that Theorem 5 is somewhat tight. In the following, we write $r : A \leftarrow B$ to denote the rule $A \leftarrow B$ and to signify that r is its name, i.e. $r = \lambda(A \leftarrow B)$.

First, peers that are cooperative w.r.t. F_1 are not forced to always disclose *all* unlocked information.

Example 1. Consider peer $\mathcal{P}_1$, whose program is $P_1 = \{r_1^1 : C_1;\ r_1^2 : C_2;\ r_1^1;\ r_1^2\}$ (where the identifier before each colon is the name of the rule on the right of the colon). In the first message, $\mathcal{P}_1$ releases only C_1, then it also releases C_2. Notice that $\mathcal{P}_1$ is not behaving in a monotonic way. By inspection, the only peer that has any chance to dominate $\mathcal{P}_1$ is the peer $\mathcal{P}_1'$ that immediately releases both C_1 and C_2. However, there is a peer $\mathcal{P}_2$ that prefers $\mathcal{P}_1$ over $\mathcal{P}_1'$. Namely, it is the (non-monotonic) peer that releases Res only after two messages. We conclude that $\mathcal{P}_1$ is cooperative w.r.t. F_1. □

Moreover, for $k > 1$, cooperative peers can even emit vacuous messages when not forced to do so.

Example 2. Consider the termination condition F_3. Consider peer $\mathcal{P}_1$ with the same program as in Example 1. This time, its release strategy is to first emit the empty message, then release C_1 and then release C_2. $\mathcal{P}_1$ is not monotonic, but it is cooperative, essentially because it does not terminate the negotiation by its own fault. For instance, the peer $\mathcal{P}'_1$ that has the same program as $\mathcal{P}_1$, but immediately releases both C_1 and C_2 does *not* dominate $\mathcal{P}_1$. The peer $\mathcal{P}_2$ that unconditionally releases Res after the third message (and sends empty messages otherwise) prefers $\mathcal{P}_1$ over $\mathcal{P}'_1$. A similar example can be built for all $k > 1$. □

3.2.2 Cooperation against Monotonic Peers

Peers that are cooperative w.r.t. F_1 against monotonic peers always release all unlocked information.

Theorem 6. *If $\mathcal{P}_1$ is cooperative w.r.t. F_1 against monotonic peers, then for all negotiations σ, we have $S_1(\sigma) = \mathsf{unlocked}(\sigma, 1)$.*

Proof. Assume by contradiction that there is a negotiation σ such that $\mathcal{P}_1$ retains rule r after σ. Formally, $r \in \mathsf{unlocked}(\sigma, 1)$, $r \notin \mathsf{received}(\sigma)$, and $r \notin S_1(\sigma)$. We can check that $\mathcal{P}_1$ is dominated by another peer $\mathcal{P}'_1 = (P_1, \lambda_1, S'_1)$, where $S'_1(\rho) = \mathsf{unlocked}(\rho, 1)$ for all negotiations ρ.

First, for all monotonic peers $\mathcal{P}_2$, if $\mathcal{P}_1$ and $\mathcal{P}_2$ lead to a successful negotiation, so do $\mathcal{P}'_1$ and $\mathcal{P}_2$. In particular, suppose that $\mathcal{P}_1$ and $\mathcal{P}_2$ lead to the successful negotiation ρ, while $\mathcal{P}'_1$ and $\mathcal{P}_2$ lead to the negotiation ρ'. Using the fact that $\mathcal{P}_2$ is monotonic, we can prove by induction that ρ is step-wise contained in ρ'. It remains to be checked that no vacuous message occurs in ρ' before Res is released. Since both $\mathcal{P}'_1$ and $\mathcal{P}_2$ are monotonic, they do not recover from vacuous messages. In other words, a vacuous message in ρ' would imply that all subsequent messages are also vacuous.

Finally, we show that there is a peer $\mathcal{P}_2$ that "prefers" $\mathcal{P}'_1$ over $\mathcal{P}_1$. We define $\mathcal{P}_2$ as follows: at the beginning, $\mathcal{P}_2$ behaves in such a way as to obtain negotiation σ. Then, $\mathcal{P}_2$ waits for rule r in order to release Res. Finally, $\mathcal{P}_2$ issues a vacuous message in reply to $\sigma \cdot S_1(\sigma)$. One can check that $\mathcal{P}_2$ can be built in such a way as to be monotonic. We obtain the contradiction that $\mathcal{P}_1$ is dominated by $\mathcal{P}'_1$. □

For all $k > 0$, in order to be cooperative w.r.t. F_k against monotonic peers, peers do not send vacuous messages after a series of $k-2$ vacuous messages, unless they have no other choice.

Theorem 7. *Assume that $\mathcal{P}_1$ is cooperative w.r.t. F_k, for some $k > 0$, against monotonic peers. For all negotiations σ, if $S_1(\sigma) \subseteq \mathsf{received}(\sigma, 2)$ and all messages in $\mathsf{tail}(\sigma, k-2)$ are vacuous, then $\mathsf{unlocked}(\sigma, 1) \subseteq \mathsf{received}(\sigma, 2)$.*

Proof. Assume by contradiction that there is a negotiation σ such that $\mathcal{P}_1$ sends a vacuous message after σ, although it could send some new rule r. Formally, there is $r \in \mathsf{unlocked}(\sigma, 1)$, $r \notin \mathsf{received}(\sigma)$, while $S_1(\sigma) \subseteq \mathsf{received}(\sigma)$. We can check that $\mathcal{P}_1$ is dominated by another peer $\mathcal{P}'_1 = (P_1, \lambda_1, S'_1)$, where S'_1 coincides with S_1 except for $(\sigma) = S_1(\sigma) \cup \{r\}$. The proof proceeds similarly to the one of Theorem 6. □

However, being cooperative w.r.t. F_k, for $k > 1$, against monotonic peers, does not imply revealing *all* unlocked information at once.

Example 3. Suppose we are using termination criterion F_2, and consider peer $\mathcal{P}_1$ from Example 1. As before, the only peer that has any chance to dominate $\mathcal{P}_1$ is the peer $\mathcal{P}_1'$ that immediately releases both C_1 and C_2. However, it is easy to check that $\mathcal{P}_1$ and $\mathcal{P}_1'$ are in fact equivalent w.r.t. all monotonic peers $\mathcal{P}_2$. So, $\mathcal{P}_1$ is cooperative against monotonic peers. □

Moreover, cooperative peers can also send vacuous messages, provided they do not lead to immediate or 1-step delayed termination.

3.3 Interoperability

We remind that all peers are assumed to be truthful and secure. The following example shows that two peers that are cooperative w.r.t. F_1 need not be interoperable.

Example 4. Consider $\mathcal{P}_1$ from Example 1. As for $\mathcal{P}_2$, we set $P_2 = \{r_2^1 : \mathsf{Res};\ r_2^1 \leftarrow C_2\}$. As release strategy, we set $S_2(\sigma) = \mathsf{unlocked}(\sigma, 2)$ for all negotiations σ. It is easy to check that $\mathcal{P}_2$ is truthful, secure and cooperative w.r.t. F_1. $\mathcal{P}_1$ and $\mathcal{P}_2$ have a successful negotiation under the most liberal release strategy, namely the negotiation $\{C_1;\ C_2\}\{\mathsf{Res};\ r_2^1 \leftarrow C_2\}$. However, they do not achieve a successful negotiation under their own strategies. □

Before we can prove the main theorem of this section, we need to state the following lemma. It essentially affirms that the set of unlocked rules depends only on the facts released so far, and not on the proper rules. It is a direct consequence of the independency condition that we put on the definition of peer.

Lemma 2. *Given a peer $\mathcal{P}_i$ and two negotiations σ_1 and σ_2, if the set of facts in σ_1 is contained in the set of facts in σ_2, then* $\mathsf{unlocked}(\sigma_1, i) \subseteq \mathsf{unlocked}(\sigma_2, i)$*, for all $i = 1, 2$.*

Theorem 8. *For all $k > 0$, if both peers are cooperative w.r.t. F_k against monotonic peers, then they are interoperable.*

Proof. Assume that $\mathcal{P}_1$ and $\mathcal{P}_2$ lead to the successful negotiation σ under the most general release strategy. Assume also that $\mathcal{P}_1$ and $\mathcal{P}_2$ give rise to the negotiation σ' when using their original release strategies.

If $k = 1$, using Theorem 6 and Lemma 2, we can prove that the set of facts that are released at each step in σ and σ' is the same. This leads immediately to the conclusion that σ' is successful. In particular, if a vacuous message occured in σ', it should *a fortiori* occur in σ, since in σ all proper rules are released in the first two steps, and all facts are released at the same time as in σ'.

If $k > 1$, assume by contradiction that σ' is terminated, and consider the last two vacuous messages. By Theorem 7, peers had no other choice than emitting these vacuous messages. Let C be the set of facts released during σ'. Clearly, we have that $\mathsf{Res} \notin C$. Now, find the greatest index i such that the set of facts in σ_i is contained in C. By Lemma 2, since the facts in σ' are not sufficient for either $\mathcal{P}_1$ or $\mathcal{P}_2$ to release any further fact, the same applies to σ_i. We obtain the contradiction that, even under the most liberal release strategy, $\mathcal{P}_1$ and $\mathcal{P}_2$ cannot release any new fact after σ_i.

□

Finally, the following result states that focused peers may not be interoperable, even when cooperative against cooperative peers.

Theorem 9. *For all $k > 0$, there are two peers that are focused and cooperative w.r.t. F_k, against cooperative peers, but are not interoperable.*

Proof. Consider the peers having the following programs:

$$P_1 = \{r_1^1 : C_1;\ r_1^2 : r_1^1\} \qquad\qquad P_2 = \{r_2^1 : \mathsf{Res};\ r_2^2 : r_2^1 \leftarrow C_1\}.$$

First, we prove that there is a successful negotiation under the most general release strategy. Such negotiation is $\{C_1\}\{\mathsf{Res};\ r_2^1 \leftarrow C_1\}$.

On the other hand, assume that peers employ the most liberal release strategy that is focused and cooperative against cooperative peers. Such strategy simply prescribes to reveal all information that is unlocked and relevant. Since rule r_2^2 cannot be unlocked, peer $\mathcal{P}_1$ will not transmit C_1, as it is not relevant. Thus, the negotiation will fail according to F_k, for all $k > 0$. □

3.4 Related work

There are only a few previous works on strategy interoperability [Yu et al., 2001, Yu et al., 2003]. They address specific families of strategies, defined on the basis of structures that resemble proof trees, and on sequences of transformations over sets of those trees. We aimed at a more general setting, based on abstract properties of the strategies and not on their internal structure. As a side effect we obtain—we believe—simpler definitions and a clearer picture of which are the real properties that enable interoperability. Currently, the results of [Yu et al., 2001, Yu et al., 2003] are stronger over their family of strategies, but we are currently extending our results by means of local properties of individual policies (see the next section).

3.5 Conclusions and future work

The impact of policy protection on negotiation success is negligible for all cooperative peers against monotonic peers. These are the "most successful" agents among those designed to interact with peers that release more information when they get more (a reasonable assumption in normal situations; we are not interested in guaranteeing interoperability with agents that only try to steal information). It turns out that these peers can also interact successfully with each other (if their policies permit), even if they are not monotonic themselves. We are still investigating the meaning of this somewhat surprising result; it might be a symptom that our requirement on strategies is too strong and can be relaxed.

On the contrary *focused* peers (i.e. those that never release a piece of information unless it is explicitly requested) do not enjoy this property. Unofrtunately, focussed peers are very common, so we are studying suitable *local conditions* that ensure interoperability. They will be the subject of a future work.

Our current results show also that for some cooperative peers, the number of "empty" messages that sanctions the end of a negotiation is not so important from the point of view of the result of the negotiation. Peers are forced to react only during the last few steps before termination, so a termination criterion based on a high number of empty messages may only have the effect of slowing down the negotiation. Simons, Winslett and Yu obtained compatible results in an unpublished work (personal communication); they prove that the number of empty messages can be restricted to $[0, 4]$ for their family of strategies.

Several other interesting aspects remain to be addressed. For example, a peer might adopt different termination criteria, such as a fixed bound on negotiation steps. The study of alternative termination protocols are an interesting subject for further research. Last but not least, multi-party negotiations still need to be addressed.

Acknowledgements

We are grateful to Silvie Spreeuwenberg and Daniel Olmedilla for their precious comments and suggestions.

References

[Bonatti, 2004] Bonatti, P. A. (2004). Abduction over unbounded domains via ASP. In de Mántaras, R. L. and Saitta, L., editors, *Proc. of ECAI'04*, pages 288–292. IOS Press.

[Chen and Toda, 1995] Chen, Z.-Z. and Toda, S. (1995). The Complexity of Selecting Maximal Solutions. *Information and Computation*, 119:231–239.

[Dantsin et al., 2001] Dantsin, E., Eiter, T., Gottlob, G., and Voronkov, A. (2001). Complexity and Expressive Power of Logic Programming. *ACM Computing Surveys*, 33(3):374–425.

[Satoh and Iwayama, 1991] Satoh, K. and Iwayama, N. (1991). Computing abduction by using the TMS. In *ICLP*, pages 505–518.

[Simons et al., 2002] Simons, P., Niemelä, I., and Soininen, T. (2002). Extending and implementing the stable model semantics. *Artif. Intell.*, 138(1-2):181–234.

[Yu et al., 2001] Yu, T., Winslett, M., and Seamons, K. (2001). Interoperable strategies in automated trust negotiation. In *CCS '01: Proceedings of the 8th ACM conference on Computer and Communications Security*, pages 146–155. ACM Press.

[Yu et al., 2003] Yu, T., Winslett, M., and Seamons, K. (2003). Supporting Structured Credentials and Sensitive Policies through Interoperable Strategies in Automated Trust Negotiation. *ACM Transactions on Information and System Security*, 6(1).