Model Checking of Qualitative Sensitivity Preferences to Minimize Credential Disclosure[⋆]

Zachary J. Oster, Ganesh Ram Santhanam, Samik Basu, and Vasant Honavar

Department of Computer Science, Iowa State University, Ames, Iowa 50011, USA
{zjoster,gsanthan,sbasu,honavar}@iastate.edu

Abstract. In most client-server interactions over the Web, the server requires the client to disclose certain credentials before providing the client with the requested service (server policy). The client, on the other hand, wants to minimize the sensitivity of the set of credentials disclosed (client preference). We present a qualitative preference formalism based on conditional importance networks (CI-nets) for representing and reasoning with client preferences over the relative sensitivity of sets of credentials. The semantics of CI-net preferences is described using a preference graph over the set of credentials for which the preferences are expressed. We develop a model checking-based approach for analyzing the preference graph, efficiently verifying whether one set of credentials is more sensitive than another (dominance testing). Further, we identify the least (minimum) sensitive set of information that may be disclosed by the client to get access to the desired service. We present a technique based on iterative verification and refinement of the preference graph for computing a sequence of credential sets, ensuring that a credential set with higher sensitivity is never returned before one with lower sensitivity. We present a prototype implementation and preliminary simulation results.

1 Introduction

In online transactions, a client often must choose from multiple servers that provide some desired service. Typically, each server expects to verify a set of the client's credentials (as specified by the server's access control policy) before allowing access to the requested service. As the servers may hold different access control policies, they may demand different sets of credentials from the client; some sets of credentials may be more *sensitive* to the client than others, in the sense that they compromise the client's privacy to a greater degree.

This induces a preference on the sets of credentials that the client can disclose. Given a set of servers providing the same service, the client will prefer a server requiring the disclosure of a less sensitive set of credentials over a server requiring the disclosure of a more sensitive credential set.

For all except the smallest sets of possible credentials, it is impractical for the client to explicitly specify preferences over all possible combinations of credentials. Even if the client has only four credentials, he or she would need to

⋆ This work is supported in part by U.S. National Science Foundation grants CCF0702758 and CCF1143734.

assert preferences over $2^4 = 16$ combinations of credentials. A more practical approach would be to specify preferences over individual credentials, which can then be used to reason about preferences over sets of credentials. This reduces the decision-making burden on the client while still allowing him or her to access a desired service from the server that requires the least sensitive set of credentials.

Existing approaches to this problem, including [4,7,14], assume that the client has a priori knowledge of the access control policy of the server. However, in practice some servers may restrict the disclosure of their access control policies [1], especially when the client may be able to infer sensitive information by looking at the server's policy. For example, a server storing medical records may wish to release the records of a patient A suffering from a certain disease only to specialist doctors who are qualified to treat the illness. The server may thus require clients (in this case doctors) requesting access to patient A to present credentials certifying that they are licensed doctors who are qualified to treat A's disease. However, disclosing this requirement may allow any client to infer that A is suffering from that ailment, violating privacy laws. In such settings, the server has to protect its policy from being fully or partially disclosed to its clients.

Given the privacy preferences of a client that is totally or partially ignorant of the server's requirements, the client can access the service while minimizing disclosure of the client's credentials by providing increasingly sensitive credential disclosure sets (beginning from the least sensitive set) until the client finds one that is accepted by the server. Thus there is a need for algorithms and formal methods that compute a sequence of successively next-best (more sensitive) credential disclosure sets based on the sensitivity preferences of the client.

Driving Problem. In this paper, we address two important problems in the context of these scenarios. First, we use an intuitive formalism for representing and reasoning with the client's sensitivity preferences over the credentials. We claim that it is natural for clients to specify their preferences over credentials using an expressive preference language, namely *conditional importance networks (CI-nets)* [2]. CI-nets can represent

- *Monotonicity Preference*: disclosing less information is preferred to disclosing more information
- *Set-Based Relative Importance Preference*: disclosing one set of credentials is always preferred to disclosing another set of credentials
- *Conditional Relative Importance Preference*: given the presence or absence of certain credentials in the disclosure set, including one set of credentials in the disclosure set is preferred to including another set of credentials

Second, we introduce a model checking-based technique for finding a sequence of successively next-most-preferred (more sensitive) credential disclosure sets with respect to the CI-net preferences specified by the client. This new technique is built upon our previous work on dominance testing via model checking for CP-nets [11], which are related to but less expressive than CI-nets.

Contributions. The contributions of our work are summarized as follows:

1. We employ a formal preference language and semantics based on CI-nets to represent the client's preferences over the sensitivity of the credentials.
2. We show how model checking techniques for preferential dominance testing (finding whether one set of credentials is preferred over another) can be used to find the least sensitive (most preferred) set of credentials that the client would like to disclose and that the server will accept. This is done by repeatedly finding the next-best (next-least-sensitive) set of credentials and seeing whether the server will accept this credential set.
3. We present an implementation and preliminary experiments to show the practical feasibility of our approach.

Organization. The rest of the paper is organized as follows. Section 2 motivates the addressed problems and the proposed solution using a simple example. Section 3 describes the syntax and semantics of CI-nets. Section 4 presents our approach to finding the most preferred sets of credentials according to the sensitivity preferences of the client, as well as a model checking-based technique for ordering the sets of credentials based on their relative sensitivities. Section 5 describes our implementation and summarizes the results of our initial experiments. Section 6 discusses related techniques from the existing literature. Section 7 summarizes the paper and presents some directions for future work.

2 Illustrative Example

Consider a client who is interested in obtaining some financial quote (e.g., auto and/or home insurance, mortgage, etc.) using an online service. Suppose that there are multiple servers that provide the required service, and each server's access control policy requires a combination of several credentials from the client before granting access to the service. We consider four such credentials: the client's name, residential address, bank account number, and bank routing number.

The client has some qualitative preferences over the relative importance of his credentials based on their sensitivity. The rationale behind these preferences is that the client would like to make it impossible (or at least difficult) for a third party to perform any financial transaction maliciously posing as the client. Therefore, from the client's perspective, the objective is to choose the server that provides the desired financial service by requiring the least sensitive set of client credentials. Consider the following qualitative preferences specified by the client:

P1. If my bank account number is disclosed to the server, I would rather give my address than my bank's routing number to the server. This is because my bank account number along with the bank routing number identifies my bank account precisely, and hence it is highly sensitive information compared to my bank account number and address.

P2. If I have to disclose my address without having to disclose my name, then I would prefer giving my bank's routing number over my bank account number. However, this preference does not hold when I have to disclose my name along with my address, because the combination of my name, address, and bank routing number is not any less sensitive than my name, address, and bank account number. In both cases, a malicious party needs to guess one of the credentials – bank account number or bank routing number – to gain access to important financial information.

P3. Because I would like to protect as many details as possible regarding my bank account, when I don't have to disclose my bank account number I would provide my name and address rather than my bank's routing number.

Based on these preferences, the client can identify successively more sensitive sets of credentials (starting from the empty set) and verify whether a set of credentials is sufficient to satisfy the access control policy of any server providing the desired service. Any server that accepts this least sensitive acceptable set of credentials may be selected to provide the service to the client.

3 Background: CI-Nets

We use *conditional importance networks* (CI-nets) [2] to capture and reason with the client's preferences over the set of credentials in terms of their sensitivity. CI-nets allow a client to clearly and precisely specify sensitivity among credentials.

3.1 Syntax

Let V denote the set of credentials over which the client expresses his/her preferences. A CI-net C is a collection of conditional importance statements of the form $S^+, S^- : S_1 \succ S_2$, where S^+, S^-, S_1, and S_2 are pairwise disjoint subsets of V and where $S_2 \neq \emptyset$. Informally, given two sets of credentials which both include the set S^+ and exclude S^-, the set that contains all of the credentials in S_1 is preferred (relatively less sensitive) to the set that contains all of the credentials in S_2.

Recall the preference P1 described in Section 2 that *"if my bank account number is disclosed to the server, I would rather give my address than my bank's routing number to the server"*. It is expressed as the following CI-net statement:

$$\{\mathit{Bank\ Account\ Number}\}, \{\} : \{\mathit{Address}\} \succ \{\mathit{Bank\ Routing\ Number}\} \qquad (1)$$

Similarly, the preference P2 that *"if I have to disclose my address without having to disclose my name, then I would prefer giving my bank's routing number over my bank account number"* is expressed in the language of CI-nets as:

$$\{\mathit{Address}\}, \{\mathit{Name}\} : \{\mathit{Bank\ Routing\ Number}\} \succ \{\mathit{Bank\ Account\ Number}\} \qquad (2)$$

3.2 Semantics

The ceteris paribus ("all else being equal") semantics [2] provides a way to use the statements in a CI-net to reason about preferences over various sets of credentials. The semantics of preferences described using a CI-net C over a set of credentials V is given in terms of a strict partial order (irreflexive and transitive) relation $\succ$ over the powerset of V such that:

1. $\succ$ is monotonic, i.e., $\gamma \subset \gamma' \Rightarrow \gamma \succ \gamma'$
2. For each CI-net statement $S^+, S^- : S_1 \succ S_2$,
   $\gamma \subseteq [V \setminus (S^+ \cup S^- \cup S_1 \cup S_2)] \Rightarrow \gamma \cup S^+ \cup S_1 \succ \gamma \cup S^+ \cup S_2$

In the original CI-net formalism [2], a set is preferred to its subset, i.e., it is always preferred to have more elements in a set. We reverse the direction of monotonicity (see item 1 above) in the semantics because in our context it is always better to disclose fewer credentials. Further, note that allowing $S_2 = \emptyset$ combined with monotonicity could permit preferences where a set is preferred to itself. In order to ensure the strict partial ordering of preferences, we do not allow $S_2 = \emptyset$ in the syntax (see Section 3.1).

Going back to the CI-net preference statement P1 in our example, by the rule in item 1 above, the set of credentials {*Name*, *Address*, *Bank Account Number*} is preferred to the set {*Name*, *Bank Account Number*, *Bank Routing Number*} according to ceteris paribus semantics. Similarly, a ceteris paribus interpretation of the preference statement P2 in our example CI-net can be used to reason that the set of credentials {*Address*, *Bank Routing Number*} is less sensitive than (therefore more preferred to) the set {*Address*, *Bank Account Number*}.

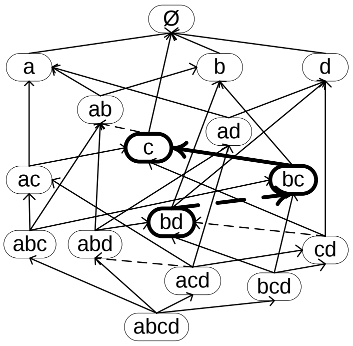
CI-Nets for Preferences over Credential Disclosure Sets. CI-nets are a natural choice for modeling client preferences over sets of credentials for the following reasons:

1. Preferences in CI-nets are monotonic. According to the semantics of CI-nets, a set of credentials is preferred to all of its proper supersets. The client would typically like to protect as many credentials as possible (ideally all) from being disclosed.
2. The CI-net semantics induces a strict partial order among the subsets of credentials with respect to the CI-net preference statements of the client. Thus, it is possible to order the subsets of credentials in a way that is consistent with the semantics of a CI-net. Such an ordering can be used to search for less sensitive sets of credentials that fulfill the server's requirement ahead of the ones that are more sensitive.

4 Finding the Most Preferred Set of Credentials

We present a method for automatically identifying a most preferred set γ of credentials that the client has to disclose in order to satisfy the server's requirement, such that there exists no other credential set γ' that (a) is preferred to (less sensitive than) γ and (b) fulfills the server's requirement. Our method consists of the following two processes:

1. *Decide:* Automatically decide the preference of a set of credentials over another, where preferences are specified using CI-nets (Section 4.1).
2. *Order:* Use the above decision process to automatically identify the preference ordering of sets of credentials, starting from the most preferred sets and ending in the least preferred ones (Section 4.4).



a = `Name`
b = `Address`
c = `Bank Routing Number`
d = `Bank Account Number`

P1. $\{d\}, \{\} \ : \ \{b\} \succ \{c\}$
P2. $\{b\}, \{a\} \ : \ \{c\} \succ \{d\}$
P3. $\{\}, \{d\} \ : \ \{a, b\} \succ \{c\}$

Fig. 1. Induced preference graph and CI-net preference statements for the preferences given in Section 2, with the improving flipping sequence from $\{$*Address*, *Bank Account Number*$\}$ to $\{$*Bank Routing Number*$\}$ shown in bold.

4.1 Dominance Testing

Given two choices (of sets of credentials), deciding the preference of one choice over the other is referred to as *dominance testing*. Dominance testing is known to be PSPACE-complete [2,6]. Recently, in [11], we have demonstrated an effective model checking [5] based approach to dominance testing for certain families of preferences, such as TCP-nets [3]. In this paper, we follow a similar approach for dominance testing between choices (of sets of credentials) where preferences are represented using CI-nets. This approach relies on alternate semantics of CI-nets given in terms of an *improving flipping sequence*, analogous to the worsening flipping sequence defined in [2].

Definition 1 (Improving Flipping Sequence [2]). *A sequence of credential sets* $\gamma_1, \gamma_2, \cdots \gamma_{n-1}, \gamma_n$ *is an* improving flipping sequence *with respect to a set of CI-net statements if and only if, for* $1 \leq i < n$*, either*

1. *(Monotonicity Flip)* $\gamma_{i+1} \subset \gamma_i$*; or*
2. *(Importance Flip) there exists a conditional importance statement* $S^+, S^- : S_1 \succ S_2$ *in the CI-net for which all of the following hold:*
   (a) $\gamma_{i+1} \supseteq S^+$*,* $\gamma_i \supseteq S^+$*,* $\gamma_{i+1} \cap S^- = \gamma_i \cap S^- = \emptyset$*;*
   (b) $\gamma_{i+1} \supseteq S_1$*,* $\gamma_i \supseteq S_2$*,* $\gamma_{i+1} \cap S_2 = \gamma_i \cap S_1 = \emptyset$*;*
   (c) $\gamma = V \setminus (S^+ \cup S^- \cup S_1 \cup S_2) \Rightarrow \gamma \cap \gamma_{i+1} = \gamma \cap \gamma_i$.

In the above definition, condition (1) states that disclosing a set of credentials is always preferred to disclosing its superset. (2) states that if the set S^+ of credentials are disclosed and the set S^- of credentials are not disclosed, then disclosing the set S_1 of credentials is preferred to disclosing the set S_2 of credentials, all other disclosures being identical (which is ensured by condition (2c)). Given a CI-net C and two sets γ and γ' of credentials, we say that γ is preferred to γ', denoted by $C \models \gamma \succ \gamma'$, if and only if there is an improving flipping sequence with respect to C from γ' to γ (Proposition 1, [2]).

In our example CI-net (right side of Figure 1), we can thus say that the set $\{$*Bank Routing Number*$\}$ is preferred to the set $\{$*Address*, *Bank Account Number*$\}$. This is because the set $\{$*Address*, *Bank Account Number*$\}$ has an improving (importance) flip to the set $\{$*Address*, *Bank Routing Number*$\}$ (see preference P2 in Section 2 and its CI-net representation, Equation 2 in Section 3), which in turn has an improving (monotonicity) flip to $\{$*Bank Routing Number*$\}$.

From the above definition, one can construct a graph where each node corresponds to a set of credentials and each directed edge from one node to another denotes an "improving flip", capturing the fact that the set of credentials at the destination node is preferred to the set of credentials at the source node. This graph is referred to as the *induced preference graph* [2].

Definition 2 (Induced Preference Graph). *Given a CI-net C over a set of credentials V, the* induced preference graph $\delta(C) = (N, E)$ *is constructed as follows. The nodes N correspond to the powerset of V, and for a pair $\gamma, \gamma' \in N$, the directed edge $(\gamma, \gamma') \in E$ indicates an improving (monotonicity or importance) flip from γ to γ' as per the CI-net semantics (Definition 1) such that $\gamma' \succ \gamma$.*

Figure 1 presents the CI-net statements representing the preferences over credentials specified in Section 2, along with the corresponding induced preference graph. The solid edges between sets of credentials in this graph correspond to monotonicity flips and the dotted edges correspond to importance flips. Each path in the graph corresponds to an improving flipping sequence.

A set γ' of credentials *dominates* (i.e., is preferred to) another set γ with respect to CI-net C if and only if the node corresponding to γ' in $\delta(C)$ is reachable from γ. For example, the set $\{$*Bank Routing Number*$\}$ is preferred to the set $\{$*Address*, *Bank Account Number*$\}$ due to the existence of the path $bd \rightarrow bc \rightarrow c$, which is highlighted in Figure 1. The induced preference graph of a CI-net is *consistent* if and only if it is cycle-free.

4.2 Kripke Structure Modeling of CI-Net Semantics

We use the Cadence SMV symbolic model checker [8] to verify reachability (and therefore dominance) from one node to another in the induced preference graph. There are three primary advantages in using Cadence SMV for testing dominance. First, Cadence SMV is equipped with (symbolic or BDD-based) algorithms that allow for efficient state-space exploration of large graphs. Second, Cadence SMV can verify properties (beyond simple reachability) in expressive temporal logic (e.g., CTL and LTL), a capability that we will use in Section 4.4 to obtain a preference ordering over sets of credentials. Finally, the SMV input language allows us to directly encode the CI-net preference statements. The induced preference graph is then automatically constructed by the model checker to answer dominance (verification) queries. The model checker takes as input a Kripke structure $\langle S, S_0, T, L\rangle$, where S is the set of states, $S_0 \subseteq S$ is the set of start states, $T \subseteq S \times S$ is the set of transition relations, and L is a labeling function mapping each state in S to a set of propositions that hold at that state.

In our encoding, we represent each credential (say, x_i) as a proposition, where the value of the proposition is true when the credential is disclosed and false when the credential is not disclosed. The propositions are uninitialized, which allow the model checker to consider all possible valuations of propositions as initial states of the Kripke structure. Given a set of CI-net statements C, the Kripke structure K_C representing the induced preference graph $\delta(C)$ contains states that are labeled with the truth values of the set of credential propositions x_i and two types of helper Boolean variables: a set of Boolean variables h_i and a Boolean variable g.

SMV Input Language: Role of Helper Variables. In SMV, a Kripke structure is encoded using a set of variables, their possible initial valuations, and a set of transition relations. Each transition relation describes the valuation of the variable based on certain conditions on the current state variable-valuations. For instance, consider a Kripke structure with two Boolean variables a and b.

```
                                   next(a) := case
                                                a = b : !a;
init(a) := 0;                                   1     : a;
                                              esac;
```

This SMV specification states that the initial valuation of a is 0, while the initial valuation of b can be either 0 or 1 since it is not explicitly given. The corresponding Kripke structure has two different start states: one where a and b are equal to 0 and another where a is equal to 0 and b is equal to 1. Furthermore, the transition relation (described by the `next` operation) states that the value of a is toggled only when the valuations of a and b are equal in the current state. The absence of `next` definitions for b indicates that the valuation of b can change non-deterministically whenever a change in state occurs in the Kripke structure.

In the encoding of $\delta(C)$ as a Kripke structure K_C, attributes over which the CI-net statements are specified are encoded as Boolean variables in K_C. Each state in K_C corresponds to a node in $\delta(C)$: if $x_3 \wedge x_4$ holds (evaluates to true) in a state in K_C, that state corresponds to the node annotated with x_3 and x_4 in $\delta(C)$. Next, note that the existence of a given edge in $\delta(C)$ depends on the contents of the source and destination nodes (improving flip, see Definition 1). Direct encoding of such edges in SMV requires encoding of transitions in K_C where the `next` operation on each variable (describing the enabling condition of the transitions) includes conditions that depend on the variables' values in the next states. Encoding such conditions in SMV may lead to circular dependencies between `next` operations for two or more variables. For instance,

```
next(a) :=  case                      next(b) :=  case
              next(b) : !a;                         next(a) : !b;
              1       : a;                          1       : b;
            esac;                                 esac;
```

From the above encoding, it is not clear what valuation a and b should have in the next state when the current state valuations of the variables are equal to 1.

Role of h_i. In order to correctly encode the edges of $\delta(C)$ as transitions in K_C, we have used one auxiliary variable h_i for each proposition x_i. Each h_i is encoded such that if h_i is 0 (false) in the current state, then in the next state the valuation of x_i cannot change; otherwise, the valuation of x_i may change in the next state if a condition corresponding to a CI-net preference statement is satisfied. The h_i variables are all initialized to 0 and the model checker performs updates to the h_is non-deterministically. For instance, the semantics of the CI-net statement $\{d\}, \{\} : \{b\} \succ \{c\}$ (preference P1 from Section 2, resulting in edges $cd \rightarrow bd$ and $acd \rightarrow abd$ in $\delta(C)$) can be encoded in the SMV language as

```
next(b) := case
                h_a = 0           -- a does not change in next state
            &   b = 0 & h_b = 1   -- b can change in the next state
            &   c = 1 & h_c = 1   -- c can change in the next state
            &   d = 1 & h_d = 0   -- d does not change in next state
                      : 1
            ...
           esac;
next(c) := case
                h_a = 0           -- a does not change in next state
            &   b = 0 & h_b = 1   -- b can change in the next state
            &   c = 1 & h_c = 1   -- c can change in the next state
            &   d = 1 & h_d = 0   -- d does not change in next state
                      : 0
            ...
           esac;
```

The enabling conditions are identical in both cases to ensure that the valuations of b and c are updated under identical conditions as specified by the CI-net, i.e., when $d = 1$ in the current and next states (ensured by $h_d = 0$ in the current state) and when the valuation of a is unaltered in the current and next states (ensured by $h_a = 0$ in the current state). Further, $c = 1$ and $h_c = 1$ in the current state, which allows the value of c to change in the next state; similarly, $b = 0$ and $h_b = 1$ in the current state, which allows for the toggling of b in the next state.

In this way, the semantics of CI-nets as given in Definition 1 is directly encoded as SMV specifications. This encoding eliminates the need to manually construct the induced preference graph $\delta(C)$; instead, the model checker automatically constructs and explores the Kripke model representing $\delta(C)$.

Role of g. Within the above encoding, the different valuations of each h_i for the same valuation of each x_i correspond to states in K_C that allow different ways in which the valuation of that x_i can be changed. Consequently, K_C contains multiple states where an identical set of x_i's hold true; all of these states correspond to one node in $\delta(C)$. Transitions between these states do not change the valuation of any x_i and, therefore, do not correspond to any edge in $\delta(C)$.

The variable g is set to 1 (true) whenever a transition traversed in K_C results in a change in the valuation of at least one of the x_i's (i.e., when a transition in K_C corresponds to an edge in $\delta(C)$). Conversely, if a transition in K_C does not indicate a change in any of the x_i variables, the variable g is set to 0 (false). Consider the following SMV code, which updates g based on the CI-net statements that encode the preferences expressed in Section 2:

```
next(g) := case
        -- Guards corresponding to P1, where g will be set to 1 :
                     h_a = 0 -- a does not change in next state
            & b = 0 & h_b = 1 -- b can change in the next state
            & c = 1 & h_c = 1 -- c can change in the next state
            & d = 1 & h_d = 0 -- d does not change in next state
                : 1 -- g is set to 1 indicating that this transition
                    -- corresponds to a change in "b" or "c"
        ...
        -- Guards corresponding to P2, where g will be set to 1 :
        ...
        -- Guards corresponding to P3, where g will be set to 1 :
        ...

        1: 0 -- default case : if no variables change, then g is 0
    esac;
```

Note that these are precisely the same conditions under which b changes to 1 (true) and c changes to 0 (false), as defined in the previous SMV code excerpt. The code in this excerpt sets g to 1 whenever the conditions for changing the value of b and c are satisfied. The full `next(g)` block contains conditions for setting g to 1 when any monotonicity or importance flip causes one or more variables to change; we have omitted the remaining conditions to save space. The `1` condition at the end of the block sets g to 0 if no other condition is met, i.e., if no variables change during the specified transition. In Section 4.4, we show how the variable g can be used directly to compute the ordering of preferred solutions.

Figure 2 shows how the data variables x_i, the helper variables h_i, and the change variable g interact within the Kripke structure K_C for a node in the induced preference graph $\delta(C)$ containing variables a and b. The most preferred node in $\delta(C)$ is the empty set, while the least preferred node is the set of all elements; nodes containing a and b are intermediate nodes. Each node in $\delta(C)$ is modeled by a set of interconnected states in K_C. In Figure 2, we have expanded and shown the set of states in K_C that corresponds to one node (where $a = 1$ and $b = 0$) in $\delta(C)$. The expanded node is divided into two subsets of states: the left subset a_g represents the set of states where $g = 1$, while the right subset $a_{\neg g}$ represents the set of states where $g = 0$. There are four states in both subsets, one for each possible valuation of the two Boolean variables h_a and h_b. Any state in a_g can be reached from some state in K_C that represents the node where $a = 1$ and $b = 1$ in $\delta(C)$. States where $h_a = 1$ and $h_b = 0$ move to states in K_C where $a = 0$ and $b = 0$, regardless of g's value. All other states in a_g can move to some state in $a_{\neg g}$ by a transition in K_C; however, since $g = 0$ in all states in $a_{\neg g}$, any transition to or between the states in $a_{\neg g}$ does not correspond to any edge in $\delta(C)$. Note that, as in a_g, the state

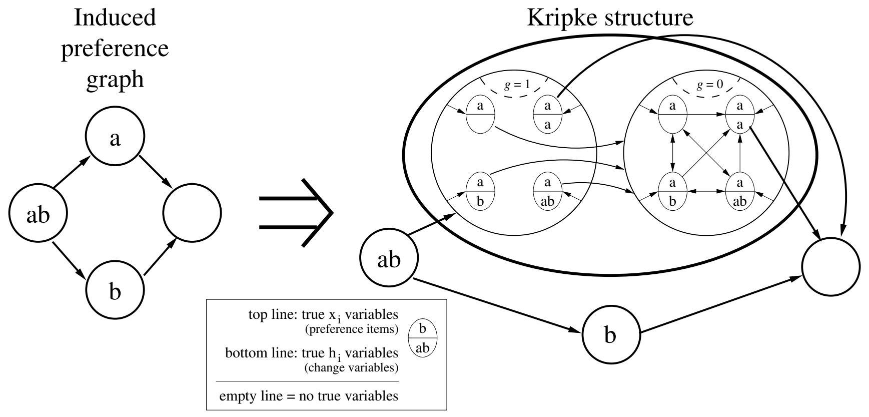

Induced preference graph
a
ab
b
Kripke structure
g = 1
g = 0
a
a
a
a
b
a
ab
ab
b
top line: true x_i variables (preference items)
bottom line: true h_i variables (change variables)
empty line = no true variables

Fig. 2. Diagram of a Kripke-structure encoding of part of an induced preference graph

in $a_{\neg g}$ where $h_a = 1$ and $h_b = 0$ has transitions to states where $a = 0$ and $b = 0$. The rest of the Kripke structure K_C is constructed similarly: each node in $\delta(C)$ corresponds to a set of states in K_C, where the number of states in the set is exponential in the number of variables (credentials) in $\delta(C)$. Further details of the SMV encoding process may be found on this paper's website at `http://fmg.cs.iastate.edu/project-pages/credentials.html`.

Theorem 1. *Given a CI-net C, a Kripke structure K_C constructed as described in this subsection preserves the semantics of the induced preference graph $\delta(C)$ of the CI-net.*

Proof. Consider the induced preference graph $\delta(C)$ for CI-net C as defined in Definition 2. Each state in K_C maps onto exactly one node in $\delta(C)$. Furthermore, given two nodes $\gamma, \gamma' \in \delta(C)$ and two states $s, s' \in K_C$ where s maps to γ and s' maps to γ', there exists a directed edge $(\gamma, \gamma') \in \delta(C)$ if and only if both (1) there exists a transition $s \rightarrow s' \in K_C$ and (2) $g = 1$ in state s'. This transition $s \rightarrow s'$ models the improving flip (γ, γ') in the induced preference graph. $\square$

4.3 Model Checking for Verifying Consistency and Dominance

Given a CI-net C, we use the method in Section 4.2 to specify the corresponding Kripke model K_C for input to the Cadence SMV model checker. We begin by verifying that the induced preference graph $\delta(C)$ modeled by K_C is consistent (i.e., cycle-free). This is done by checking K_C against the LTL formula `F G`$(g = 0)$, which is satisfied if and only if every path from the initial state in K_C eventually reaches a point where no x_i variable ever changes (i.e., g is always 0) in any future state.[1] If a cycle exists in the induced preference graph, then every state in the cycle always has at least one outgoing transition from that state where $g = 1$, indicating that a variable is changing; this violates the consistency property.

[1] Details of LTL syntax and semantics can be obtained in [10].

After the model K_C is verified to be consistent, it can be used for preference reasoning. For any two sets of credentials γ and γ', we use the following CTL formula to check whether γ' is preferred to γ: $X \Rightarrow \texttt{EF}(X')$, where X (resp. X') is the propositional formula indicating the presence or absence of credentials in γ (resp. γ'). This property is satisfied by any state in K_C where X holds true and where there is a path leading to a state where X' holds true.[2] If the property is satisfied, we conclude that γ' is preferred to γ. An improving flipping sequence from γ to γ' can be obtained by querying the model checker with the negation of the formula $X \Rightarrow \texttt{EF}(X')$. The counter-example to this formula returned by the model checker is a path in the Kripke structure that proves dominance, which can be used to construct the improving flipping sequence. On the other hand, if the property $X \Rightarrow \texttt{EF}(X')$ is not satisfied, then there exists no improving flipping sequence from γ to γ', i.e., γ' is *not* preferred to γ. In the CI-net used in our example (see Section 2), the model checker returns true when queried with the formula $(acd \Rightarrow \texttt{EF}(bd))$, which verifies that bd is preferred to or dominates acd. When we query the model checker with the CTL formula $\neg(acd \Rightarrow \texttt{EF}(bd))$, it yields a counter-example corresponding to either the path $acd \rightarrow cd \rightarrow bd$ or the path $acd \rightarrow abd \rightarrow bd$. Either path gives a proof of the dominance of bd over acd.

We find the most preferred set of credentials by verifying the CTL property $\texttt{EF}(g = 1)$ for all states in K_C. This property is satisfied at a state s in K_C if and only if s can reach any state (including itself) where g evaluates to 1 (true). The property is not satisfied at states in K_C that correspond to the *top-most node* (containing the most preferred set of credentials) of the induced preference graph (see, for instance, Figure 1). This is because the top-most node in $\delta(C)$ does not contain any outgoing edges. Any one of the states in K_C that corresponds to the top-most node in $\delta(C)$ is identified by Cadence SMV as a counterexample, proving the unsatisfiability of the property $\texttt{EF}(g = 1)$. In our running example, this query returns the state where variables a, b, c, and d are false, which corresponds to the empty set of credentials. This reflects the fact that not disclosing any credentials at all is the most preferred option.

4.4 Preference Ordering over Credential-Sets

Once the induced preference graph $\delta(C)$ is modeled as a Kripke structure K_C, our next objective is to order the sets of credentials from most to least preferred. Note that $\delta(C)$ specifies a strict partial order between sets of credentials. The ordering we obtain is a total order consistent with this strict partial order. We achieve this by performing model checking on the model K_C and its modifications against CTL properties. The steps in our approach are as follows.

1. We verify all states in K_C against the CTL property $\texttt{EF}(g = 1)$, which returns the most preferred set of credentials (say γ_i) from the top of $\delta(C)$. Since $\delta(C)$ is a strict partial order, it may have multiple elements at the top.

[2] Details of CTL syntax and semantics can be obtained in [5].

Any state that corresponds to any one of the top elements will be returned as a counterexample (proving the unsatisfiability of the CTL property).

2. Let $\gamma_1, \gamma_2, \ldots, \gamma_n$ be the sequence of sets of credentials that has been obtained so far (as the total order consistent with the partial order presented in $\delta(C)$). We define the following formula

$$I = \bigvee_{i=1}^{n} \bigwedge_{j} (x_{ij}) \tag{3}$$

where x_{ij} is the proposition representing the presence or absence of the jth credential in the set γ_i. We then query the model checker to verify whether the modified CTL property $\texttt{EF}(g = 1) \ \vee \ I$ holds true in all states of K_C. The property is satisfied by a state s in K_C if and only if (a) s can reach some state (including itself) where $g = 1$ or (b) s corresponds to nodes $\gamma_1, \gamma_2, \ldots, \gamma_n$ in $\delta(C)$. If the property is not satisfied by s, then s cannot reach a state where g is set to true and s does not correspond to nodes $\gamma_1, \gamma_2, \ldots, \gamma_n$.

3. If the model checker returns false, then it identifies (as a counterexample) a state corresponding to a set of credentials γ_{n+1}, which is at least as preferred as one of the previously identified sets of credentials $\gamma_1, \gamma_2, \ldots, \gamma_n$. In this case, we iterate Step 2 using the new sequence $\gamma_1, \gamma_2, \ldots, \gamma_{n+1}$. Otherwise, the property is satisfied by all states in K_C, meaning there exists no set of credentials that is at least as preferred as one of the elements in $\gamma_1, \gamma_2, \ldots, \gamma_n$. If this occurs, we remove from the Kripke structure K_C all states corresponding to the credential sets $\gamma_1, \gamma_2, \ldots, \gamma_n$ (obtained by iterating Step 2 so far) by adding $\neg I$ (see Equation 3) to the Kripke structure as an invariant (the model checker only considers the states where the invariant holds). Thus, the reduced model corresponds to the induced preference graph where the nodes corresponding to $\gamma_1, \gamma_2, \ldots, \gamma_n$ are not considered. We then iterate starting from Step 1 until the invariant results in a model where no states are considered by the model checker.

Note that in Step 2, the states in the model corresponding to $\gamma_1 \ldots \gamma_n$ are *not considered* as counterexamples by the model checker, as I is added as a disjunction to the property $\texttt{EF}(g = 1)$. This enables us to obtain the top-most nodes one by one in sequence without altering the model. However, when all the top-most nodes are obtained, we *remove* the states corresponding to $\gamma_1 \ldots \gamma_n$ from the model in Step 3 (by adding the $\neg I$ as an invariant to the current model). This modification of the model makes it possible for us to obtain the next set of top-most nodes in the subsequent iteration. We explain the above steps using the example $\delta(C)$ presented in Figure 1.

Iteration 1: The Kripke structure K_C encoding of $\delta(C)$ is first model-checked with the property $\texttt{EF}(g = 1)$ following Step 1 above. The result (counterexample) obtained is the top-most element $\gamma_{11} = \emptyset$. In Step 2, model checking is performed again with the property $\texttt{EF}(g = 1) \vee I$, where $I = (\neg a \wedge \neg b \wedge \neg c \wedge \neg d)$ corresponds to the absence of any credentials ($\gamma_{11} = \emptyset$). The property is satisfied because all

states except the one corresponding to $\gamma_{11} = \emptyset$ can reach a state where $g = 1$ (true). As per Step 3, we remove from K_C the states corresponding to the node $\gamma_{11} = \emptyset$ by adding $\neg I = (a \vee b \vee c \vee d)$ as an invariant to K_C. As a result, we have forced the model checker to consider only the states where the invariant holds (the invariant does not hold at states corresponding to γ_{11}). This can be viewed as an updated K_C, which encodes a $\delta(C)$ where the nodes $((a), (b), (d))$ are at the top (as Figure 1, but with the $\emptyset$ node and its incoming edges removed).
Iteration 2: Step 1 is performed again with the updated model, and the model checker returns as a counterexample one of the states that corresponds to either (a), (b), or (d). Note that such a state is identified non-deterministically by the model checking algorithm. Suppose that the state corresponding to (a) is obtained as a counterexample. So far, we have $\gamma_{11} = \emptyset$ (from the previous iteration) followed by $\gamma_{21} = (a)$ in our total ordering of sets of credentials. Proceeding to Step 2, we have a new $I = (a \wedge \neg b \wedge \neg c \wedge \neg d)$. When model checking is performed again, one of the states corresponding to either (b) or (d) is obtained as a counterexample. Suppose that a state corresponding to $\gamma_{22} = (b)$ is returned.

We proceed to perform Step 2 again with $I = (a \wedge \neg b \wedge \neg c \wedge \neg d) \vee (\neg a \wedge b \wedge \neg c \wedge \neg d)$. The model checker returns a counterexample state corresponding to the node $\gamma_{23} = (d)$. Proceeding further, Step 2 is again performed using $I = (a \wedge \neg b \wedge \neg c \wedge \neg d) \vee (\neg a \wedge b \wedge \neg c \wedge \neg d) \vee (\neg a \wedge \neg b \wedge \neg c \wedge d)$. At this point, the model checker fails to find any counterexamples for the property $\mathtt{EF}(g = 1) \vee I$. In Step 3, we remove all the states corresponding to the nodes (a), (b), and (d) by adding to K_C the invariant $\neg I = (\neg a \vee b \vee c \vee d) \wedge (a \vee \neg b \vee c \vee d) \wedge (a \vee b \vee c \vee \neg d)$ to the model (in conjunction with the invariant $(a \vee b \vee c \vee d)$ used to remove the node $\emptyset$ in iteration 1), and we start a new iteration from Step 1. So far, we have obtained an ordering of sets of credentials $\gamma_{11} = \emptyset, \gamma_{21} = (a), \gamma_{22} = (b), \gamma_{23} = (d)$.

The iterative process (starting from Step 1) is illustrated in Table 1. The iteration is continued until including an invariant in K_C results in an empty model (i.e., a model with no states). The number of such iterations is equal to the height of the partial order in $\delta(C)$. In the example (Figure 1), it is equal to 9. Each such iteration obtains a sequence of sets of credentials that are equally preferred (or indistinguishable as per the given preferences). For instance, in iteration 3, we obtain the equally preferred sets (ab) and (ad). Such elements are obtained by iterating Step 2 multiple times. The maximum number of iterations starting at Step 2 is equal to the width of the partial order in $\delta(C)$. In the example (Figure 1), it is equal to 3.

The main advantage of using this method is that a total ordering of sets of credentials is obtained without performing all possible pairwise comparisons. Instead, systematic updates to the model corresponding to the induced preference graph and repeated model checking using a CTL property are used to automatically and effectively find the total order over the sets of credentials.

Finding Preferred Sets of Credentials with Sensitivity Thresholds. We have presented a technique for using sensitivity preferences to generate a sequence or ordering of sets of credentials such that less (or equally) sensitive sets of credentials are obtained prior to more sensitive sets of credentials based on

Table 1. Steps in finding the ordering of sets of credentials for example in Section 2

#	*Iteration*	*Query*	*Result*	*Action*
1.	Iteration 1	$\mathtt{EF}(g = 1)$	[ ]	$I = (\bar{a}\bar{b}\bar{c}\bar{d})$
2.		$\mathtt{EF}(g = 1) \vee I$	–	Revise model by adding $\neg I$ as invariant
3.	Iteration 2	$\mathtt{EF}(g = 1)$	$[a]$	$I = (a\bar{b}\bar{c}\bar{d})$
4.		$\mathtt{EF}(g = 1) \vee I$	$[b]$	$I = (a\bar{b}\bar{c}\bar{d}) \ \vee \ (\bar{a}b\bar{c}\bar{d})$
5.		$\mathtt{EF}(g = 1) \vee I$	$[d]$	$I = (a\bar{b}\bar{c}\bar{d}) \ \vee \ (\bar{a}b\bar{c}\bar{d}) \ \vee \ (\bar{a}\bar{b}\bar{c}d)$
6.		$\mathtt{EF}(g = 1) \vee I$	–	Revise model by adding $\neg I$ as invariant
7.	Iteration 3	$\mathtt{EF}(g = 1)$	$[ab]$	$I = (ab\bar{c}\bar{d})$
8.		$\mathtt{EF}(g = 1) \vee I$	$[ad]$	$I = (ab\bar{c}\bar{d}) \ \vee \ (a\bar{b}\bar{c}d)$
9.		$\mathtt{EF}(g = 1) \vee I$	–	Revise model by adding $\neg I$ as invariant
10.	Iteration 4	$\mathtt{EF}(g = 1)$	$[c]$	$I = (\bar{a}\bar{b}c\bar{d})$
11.		$\mathtt{EF}(g = 1) \vee I$	–	Revise model by adding $\neg I$ as invariant
12.	Iteration 5	$\mathtt{EF}(g = 1)$	$[ac]$	$I = (a\bar{b}c\bar{d})$
13.		$\mathtt{EF}(g = 1) \vee I$	$[bc]$	$I = (a\bar{b}c\bar{d}) \ \vee \ (\bar{a}bc\bar{d})$
14.		$\mathtt{EF}(g = 1) \vee I$	–	Revise model by adding $\neg I$ as invariant
15.	Iteration 6	$\mathtt{EF}(g = 1)$	$[abc]$	$I = (abc\bar{d})$
16.		$\mathtt{EF}(g = 1) \vee I$	$[bd]$	$I = (abc\bar{d}) \ \vee \ (\bar{a}b\bar{c}d)$
17.		$\mathtt{EF}(g = 1) \vee I$	–	Revise model by adding $\neg I$ as invariant
18.	Iteration 7	$\mathtt{EF}(g = 1)$	$[abd]$	$I = (ab\bar{c}d)$
19.		$\mathtt{EF}(g = 1) \vee I$	$[cd]$	$I = (ab\bar{c}d) \ \vee \ (\bar{a}\bar{b}cd)$
20.		$\mathtt{EF}(g = 1) \vee I$	–	Revise model by adding $\neg I$ as invariant
21.	Iteration 8	$\mathtt{EF}(g = 1)$	$[acd]$	$I = (a\bar{b}cd)$
22.		$\mathtt{EF}(g = 1) \vee I$	$[bcd]$	$I = (a\bar{b}cd) \ \vee \ (\bar{a}bcd)$
23.		$\mathtt{EF}(g = 1) \vee I$	–	Revise model by adding $\neg I$ as invariant
24.	Iteration 9	$\mathtt{EF}(g = 1)$	$[abcd]$	$I = (abcd)$
25.		$\mathtt{EF}(g = 1) \vee I$	–	Revise model by adding $\neg I$ as invariant
26.		$\mathtt{EF}(g = 1)$	–	No more states to explore. Terminate.

the preferences of the client. As we have shown, such an ordering can be used to select a server from many that provide similar services. However, this ordering by itself does not allow clients to prevent highly sensitive sets of credentials from being disclosed. In many settings, clients may want to add additional constraints to prevent such unacceptable disclosures. One way to express these constraints is to specify, in addition to the preferences, one or more "threshold" sets of credentials which indicate the *maximum* sensitivity of information that the client would like to disclose. In other words, the client will consider disclosing sets of credentials in order of their sensitivity, as long as they are not more sensitive than the threshold(s). Our model-based technique can seamlessly incorporate such thresholds by extending the property to $\mathtt{EF}(g = \mathit{true}) \vee \mathtt{EX}\ \mathtt{EF}(\bigvee_i t_i)$, where t_is denote the credential sets describing the thresholds.

5 Implementation and Experiments

5.1 Overview of Framework

We have implemented our approach to finding the most preferred set of credentials with respect to the client's sensitivity preferences in a Java-based framework. Our framework consists of two primary modules:

1. A *pre-processor* module that uses two sub-modules to produce input to the model checker, namely:
   (a) *Parser:* Reads CI-net statements specified in a text input file.
   (b) *Translator:* Automatically translates CI-net statements to generate the SMV input model.
2. A *reasoning driver* module that coordinates preference reasoning. Its two sub-modules invoke the Cadence SMV model checker [8] to do different tasks:
   (a) *Consistency Checker:* Checks the consistency of CI-nets, returning true if and only if the CI-net is consistent.
   (b) *Rank Order Generator:* Takes the model generated by the pre-processor, generates appropriate temporal properties, and invokes the Cadence SMV model checker [8]. After the first run of the model checker, it reads the output of the model checker, appropriately updates the property or refines the model (by including invariants), and repeatedly invokes the model checker until all ordered results are obtained.

5.2 Experimental Setup

For our experiments, we generated random CI-nets with between 5 and 20 variables (denoting the disclosure of credentials) and either 5 or 10 CI-net statements. We tested the consistency of each sample CI-net generated according to these combinations of variables and statements; consistency is necessary to ensure that the induced preference graph does not contain any loops. We collected 20 consistent samples for each combination of variables and statements being considered, then applied the algorithm described in Section 4 to find the top 25 (next-)most preferred sets of credentials for each randomly generated sample. Our experiments were performed and results were recorded on a Dell Latitude E5420 with an Intel Core i5-2410M 2.30 GHz dual-core CPU and 4 GB of RAM, running a 64-bit Windows 7 operating system.

To examine the practical feasibility of our approach, we collected time and memory usage data from the Cadence SMV model checker for each sample tested in the experiment. We observed that consistency checking was much more resource-intensive than identifying next-most-preferred credential sets, especially with 16 or more variables; however, with 15 or fewer variables, consistency checking generally used less than one second and 7 MB of memory (for 5 statements) or a few seconds and 15 MB of memory (for 10 statements). To identify each next-most-preferred set of credentials given up to 18 variables and 5 statements (or up to 16 variables and 10 statements), the model checker generally used less than 300 ms of time and less than 7 MB of memory; however, resource usage can increase significantly once these bounds are passed. The amounts of resources required to identify the next-most-preferred set in each case remain relatively stable regardless of whether the overall most-preferred credential set or the 25th-most-preferred set is being obtained. These results show that our approach is feasible for use in practical applications.

Data from our experiments and a prototype version of our tool are available at `http://fmg.cs.iastate.edu/project-pages/credentials.html`.

6 Related Work

In the past, cost-based approaches [4] for minimizing credential disclosure have been proposed. These approaches assign higher cost to more sensitive credentials of the client; the objective is to minimize the cost associated with disclosing a set of credentials while satisfying the server's requirements. Similarly, the point-based approach in [14] assigns points to each credential based on the trustworthiness of the client, and the client values its credentials with a private score. The approaches in [4] and [14] use quantitative valuations to model preferences; in our view, qualitative valuations are better for representing the naturally qualitative preferences in this setting. Kärger et al. developed an expressive logic-based preference formalism [7] for specifying qualitative privacy preferences over the user's credentials, which can be used to minimize the sensitivity of the disclosed credentials. In contrast to all of these approaches, which require the client to have a priori knowledge of the server's access control policy, our method is able to minimize disclosure of the client's credentials even when all or part of the access control policy is unavailable to the client.

A similar problem arises in online trust negotiation [12,13,15], where a client iteratively negotiates with a server in order to determine the least sensitive set of credentials that is acceptable to the server. Our approach can be applied within such automatic trust negotiation frameworks, even when negotiating with servers that have partially or fully protected access control policies.

Our earlier work in [11] introduced a new technique for using model checking to compute dominance between two outcomes when preferences are expressed in CP-nets. This paper builds on the ideas in [11] to solve two different problems using model checking: in addition to computing dominance when preferences are expressed in CI-nets, we also compute the sequence of next-most-preferred outcomes. Our work in [9] addresses a related problem in the domain of goal-oriented requirements engineering, while this paper focuses on the details of the modeling strategy and the method of computing next-most-preferred sets in order to minimize the sensitivity of credentials disclosed by a client to the server.

7 Summary and Discussion

In this paper, we introduced a new approach based on the CI-net [2] formalism for representing and reasoning with a client's sensitivity preferences over credentials. We have developed a model checking-based technique for finding a sequence of successively next-preferred (more sensitive) credential disclosure sets with respect to CI-net preferences specified by the client. Our approach involves encoding the semantics of a CI-net as a model in the input language of the Cadence SMV model checker, then querying the model checker with temporal logic formulas to check the consistency of the CI-net preferences and to obtain the top-k ranked sets of credentials such that less sensitive credentials are returned before more sensitive ones. We have presented an implementation and performed experiments that show the practical feasibility of our approach for computing consistency and finding the top 25 sets of credentials when given CI-nets of varying sizes.

Our approach can be used in client-server negotiation settings such as choosing the most preferred server to provide a service (such that least-sensitive credentials are disclosed), as well as in online trust negotiation where clients incrementally disclose sensitive credentials while negotiating with servers that have partially or fully protected access control policies. We are now seeking "real-world" industrial applications where we can compare the performance of our approach against existing solutions to these problems. Our future plans also involve developing techniques that take into account server preferences for obtaining some client credentials over others along with the client's sensitivity preferences.

References

1. Ardagna, C.A., De Capitani di Vimercati, S., Foresti, S., Neven, G., Paraboschi, S., Preiss, F.-S., Samarati, P., Verdicchio, M.: Fine-Grained Disclosure of Access Policies. In: Soriano, M., Qing, S., López, J. (eds.) ICICS 2010. LNCS, vol. 6476, pp. 16–30. Springer, Heidelberg (2010)
2. Bouveret, S., Endriss, U., Lang, J.: Conditional importance networks: A graphical language for representing ordinal, monotonic preferences over sets of goods. In: Boutilier, C. (ed.) IJCAI, pp. 67–72 (2009)
3. Brafman, R.I., Domshlak, C., Shimony, S.E.: On graphical modeling of preference and importance. J. Artif. Intell. Res. (JAIR) 25, 389–424 (2006)
4. Chen, W., Clarke, L., Kurose, J., Towsley, D.: Optimizing cost-sensitive trust-negotiation protocols. In: INFOCOM, pp. 1431–1442 (2005)
5. Clarke, E., Grumberg, O., Peled, D.: Model Checking. MIT Press (January 2000)
6. Goldsmith, J., Lang, J., Truszczynski, M., Wilson, N.: The computational complexity of dominance and consistency in CP-nets. JAIR 33, 403–432 (2008)
7. Kärger, P., Olmedilla, D., Balke, W.-T.: Exploiting Preferences for Minimal Credential Disclosure in Policy-Driven Trust Negotiations. In: Jonker, W., Petković, M. (eds.) SDM 2008. LNCS, vol. 5159, pp. 99–118. Springer, Heidelberg (2008)
8. McMillan, K.L.: Cadence SMV (software). Release 10-11-02p1 (2002), `http://www.kenmcmil.com/smv.html`
9. Oster, Z.J., Santhanam, G.R., Basu, S.: Automating analysis of qualitative preferences in goal-oriented requirements engineering. In: Alexander, P., Pasareanu, C.S., Hosking, J.G. (eds.) ASE, pp. 448–451. IEEE (2011)
10. Pnueli, A.: The temporal logic of programs. In: FOCS, pp. 46–57. IEEE Computer Society (1977)

11. Santhanam, G.R., Basu, S., Honavar, V.: Dominance testing via model checking. In: AAAI, pp. 357–362. AAAI Press (2010)
12. Winsborough, W., Seamons, K., Jones, V.: Automated trust negotiation. In: Proceedings DARPA Information Survivability Conference and Exposition, DISCEX 2000, vol. 1, pp. 88–102. IEEE (2000)
13. Winsborough, W.H., Li, N.: Safety in automated trust negotiation. In: IEEE Symposium on Security and Privacy, pp. 147–160. IEEE Computer Society (2004)
14. Yao, D., Frikken, K.B., Atallah, M.J., Tamassia, R.: Private information: To reveal or not to reveal. ACM Trans. Inf. Syst. Secur. 12, 6:1–6:27 (2008)
15. Yu, T., Winslett, M., Seamons, K.E.: Interoperable strategies in automated trust negotiation. In: Reiter, M.K., Samarati, P. (eds.) ACM Conference on Computer and Communications Security, pp. 146–155. ACM (2001)