Game-Theoretic Deductive Verification of a Contract-Signing Protocol

MATTEO SLANINA
and
HENNY B. SIPMA
and
ZOHAR MANNA
Stanford University

We present a deductive proof of a contract-signing protocol proposed by Asokan et al. The protocol is modeled as an alternating discrete system (ADS) and properties to be verified are specified in ATL*. The proof is performed using proof rules that reduce the validity of ATL* properties over ADS’s to first-order verification conditions. The proof is concisely represented by diagrams that visualize the main structure of the proof while embedding all verification conditions.

Categories and Subject Descriptors: []:
General Terms:
Additional Key Words and Phrases:

1. INTRODUCTION

The Internet has become a major medium to conduct business. Traditional face-to-face protocols, however, are often not adequate for online transactions. For example, when signing a contract, none of the participants wants to find itself in the situation where it has provided its signature, but it has no guarantee of obtaining the other party’s signature. In the last few years, several protocols have been proposed for a variety of online transactions, including contract signing, non-repudiation, certified email. These protocols, although usually relatively small, are often intricate and surprisingly hard to verify, especially in an asynchronous environment where participants can deviate from the protocol at any time, network connections can fail, and intruders can interfere.

Verification of protocol properties requires an adequate modeling formalism and specification language. It has been found that such protocols are most naturally modeled as game structures that can represent both cooperative and adversarial behaviors. The logics ATL and ATL* [Alur et al. 2002] were proposed to specify prop-

Author’s address: Matteo Slanina, Stanford University, Computer Science Department, 353 Serra Mall, Stanford, CA 94305-9045.
e-mail: matteo@cs.stanford.edu

This is a Technical Report. Permission is granted to make digital/hard copy of all or part of this material without fee for personal or classroom use, provided that the copies are not made or distributed for profit or commercial advantage, the copyright/server notice, the title of the publication, and its date appear.
© 2006 The Authors

erties of such systems. Alur et al. [2002] showed that the verification of ATL* properties over finite-state systems is decidable, and they proposed several model checking algorithms. Model checking of ATL (a restricted form of ATL*) properties over finite-state alternating systems was implemented in MOCHA [Alur et al. 1998]. MOCHA has since been applied to the analysis of several security and contract signing protocols [Kremer and Raskin 2003] [Kremer and Raskin 2002] [Chadha et al. 2006], and mechanism design [Pauly and Wooldridge 2003].

Using model checking to verify protocols, however, has several disadvantages that can be overcome by using deductive methods. The first limitation is the restriction to finite-state systems. Although in some of the analyses mentioned above the restriction to finite-state systems was not a problem, in general this is not the case. For example, the analysis of the multi-party contract signing protocol of [Kremer and Raskin 2002] [Chadha et al. 2006], which is parameterized by the number of participating parties, was limited to small instances with three or four parties. Also when dealing with multi-session protocols the state-space becomes unbounded.

The second problem is the assumption of perfect information. This assumption is necessary for model checking, because checking ATL properties over systems with imperfect information is undecidable already for finite-state systems [Alur et al. 2002]. This shortcoming is especially severe in the modeling and verification of security and transaction protocols, because assuming perfect information when in fact not all information is accessible to all players, is unsound. Indeed a player may have a strategy to achieve a goal when it can distinguish between all states, but it may not have a strategy if it can only base its decision for the next action on part of the state information. Previous verification efforts have sidestepped this issue by carefully encoding the knowledge available to each of the participants in each state, but this is rather error-prone.

A third disadvantage is the general problem that model checking is attractive for debugging, but not very convincing when it returns no bugs. A deductive proof, on the other hand, provides evidence of the validity of the protocol that exposes explicitly the assumptions made on each of the participants and their interactions and can be checked independently.

In this paper we demonstrate the construction of a deductive proof for the contract signing protocol proposed by Asokan et al. [1998]. We model the protocol as an alternating discrete system and specify the main property of this protocol, namely fairness for an honest player with a potentially dishonest partner, as an ATL* property. To verify this property we apply the proof rules proposed in [Slanina et al. 2006] that reduce the proof to checking validity of first-order verification conditions. We represent the completed proof by diagrams that concisely visualize the main argument of the proof while embedding the detailed verification conditions that support the validity of this argument.

The rest of this paper is organized as follows. In Section 2 we describe our modeling and specification language and give a brief overview of the proof system. In Section 3 we present the contract-signing protocol and its representation in our modeling language. Section 4 specializes the proof system to the type of properties we want to prove and introduces diagrams. Section 5 contains the proof that the
protocol is fair. Section 3 concludes.

2. PRELIMINARIES

2.1 Computational Model: Alternating Discrete Systems

As computational model we use alternating discrete systems (henceforth, ADS) [Slanina et al. 2006], based on the fair discrete systems of Kesten and Pnueli [2005]. An ADS is a first-order representation of alternating structures that generalizes turn-based, synchronous and asynchronous concurrency models of [Alur et al. 2002].

States and fairness conditions are represented as value assignments to a finite set of typed variables. Formally, an alternating discrete system (ADS) is a tuple \( S = \langle \Omega, V_S, V_\Omega, \xi, \chi, \mathcal{F} \rangle \), where:

- \( \Omega \) is a finite set of players;
- \( V_S \) is a finite set of typed system variables; a state is a typed value assignment to the variables in \( V_S \); the set of all states is denoted by \( \Sigma \).
- \( V_\Omega = \langle V_a \mid a \in \Omega \rangle \) provides each player with a finite set of typed action variables. An \( a \)-action is a typed value assignment to the variables in \( V_a \); the set of all \( a \)-actions is denoted by \( \Gamma_a \). An \( A \)-action for a set of players \( A \subseteq \Omega \) is a typed value assignment to the variables in \( V_A = \bigcup_{a \in \Omega} V_a \); the set of all \( A \)-actions is denoted by \( \Gamma_A \).
- \( \xi = \langle \xi_a \mid a \in \Omega \rangle \) associates to each player \( a \) a first-order formula over variables \( V_S \) and \( V_a \) that restricts the actions player \( a \) can choose at each state: at state \( V_S \), player \( a \) can choose only actions such that \( \xi_a(V_S, V_a) \) holds. The extension of \( \xi \) to a set of players \( A \subseteq \Omega \) is defined as \( \xi_A(V_S, V_A) = \bigwedge_{a \in A} \xi_a(V_S, V_a) \);
- \( \chi(V_S, V_\Omega, V'_{\Omega}) \) expresses that the system can move from state \( V_S \) to state \( V'_{\Omega} \) when the players’ choices are \( V_\Omega \);
- \( \mathcal{F} : \Omega \rightarrow \mathcal{B}(\omega) \) assigns to each player a fairness condition, represented as a Boolean formula over atoms of the form \( \infty p \) (read “infinitely many times \( p \)”), where \( p \) is an assertion (quantifier-free formula) over \( V_S \). For example, \( \infty(x = 2 \land y > x) \rightarrow \infty(y \geq x^2) \).

We assume that an ADS has no blocking states, i.e., states from which a player has no legal action, or from which there is no available successor state for certain choices of the players. In practice this can be verified by a simple set of first-order verification conditions.

Given an ADS \( S \), a run consists of the following game played ad infinitum: At each state \( s \in \Sigma \) assigning values to variables \( V_S \), every player \( a \in \Omega \), independently of the others, picks an action by choosing values for the local variables \( V_a \) so that \( \xi_a(V_S, V_a) \) holds. Then, the next state is nondeterministically chosen among the assignments to \( V'_{\Omega} \) such that \( \chi(V_S, V_\Omega, V'_{\Omega}) \) holds. Our assumption of non-blocking guarantees that such an assignment always exists. Formally, a sequence \( \pi \in \Sigma^\omega \) is a run of \( S \) from \( s \in \Sigma \), with choices \( \rho \in \Gamma^\omega \), if \( \pi[0] = s \) and

\[
\xi_a(\pi[n], \rho[n]_a) \quad \chi(\pi[n], \rho[n], \pi[n + 1])
\]
for all $n < \omega$ and $a \in \Omega$. A run from $X \subseteq \Sigma$ is a run from any state $s \in X$. A run $\pi$ is fair to player $a$, written $\pi \models F_a$, if $F_a$ evaluates to true under the interpretation of atoms $\infty p$ as “$p$ holds at $\pi[n]$ for infinitely many $n$”.

A player $a \in \Omega$ can make its choices $\rho$ in accordance with a strategy, a function $f_a : \Sigma^+ \rightarrow \Gamma_a$, such that $\xi_a(s, f_a(ws))$ holds for all $w \in \Sigma^+$ and $s \in \Sigma$. A run $\pi$ is compatible with strategy $f_a$ for player $a$ if its choices $\rho$ satisfy $\rho[n]_a = f_a(\pi[0\ldots n])$ for all $n < \omega$. A run is compatible with strategies $f_A$ (denoting the sequence $\langle f_a \mid a \in A \rangle$), for $A \subseteq \Omega$, if it is compatible with $f_a$ for all $a \in A$. The set of all runs compatible with $f_A$ starting at a certain state $s \in \Sigma$ is called the set of outcomes of $f_A$ from $s$ and denoted $\text{out}_s(s, f_A)$.

The fundamental operator to describe properties of discrete structures is the controllable predecessors operator $\text{cpre}_A$. Given a set of states $X \subseteq \Sigma$ and a set of players $A \subseteq \Omega$, $\text{cpre}_A(X)$ denotes the set of states from which the players in $A$ have a collaborative action with which they can ensure that the game will be in $X$ at the next state. Formally,

$$\text{cpre}_A(\varphi)(V_s) \equiv \exists V_A. \xi_A(V_s, V_A) \land \forall V_{\Omega \setminus A}. \xi_{\Omega \setminus A} \rightarrow \forall V'_{S_A}. \chi(V_s, V_{\Omega}, V'_S) \rightarrow \varphi(V'_S).$$  \hspace{1cm} (1)

For classes of ADS with special properties, the cpre transformers have simpler forms [Alur et al. 2002; Slanina 2002]. To take advantage of these simpler forms, we express our verification conditions in terms of these transformers as much as possible.

2.2 The Logic ATL*

ATL* (Alternating Temporal Logic) was proposed by Alur \textit{et al.} to allow selective quantification over runs that are the possible outcomes of games [Alur et al. 2002].

2.2.0.1 Syntax.. ATL* formulas come in two types, state formulas and path formulas, defined by mutual induction. A (state) formula is one of: (1) an assertion (first-order formula) in the underlying state language, (2) a Boolean combination of state formulas, (3) $\langle A \rangle \varphi$, $[A] \varphi$, $\langle A \rangle f \varphi$, or $[A] f \varphi$, for $A$ a set of players and $\varphi$ a path formula.

A path formula is one of: (1) a state formula, (2) a Boolean combination of path formulas, or (3) an LTL temporal operator applied to path formulas. For LTL operators we use the notation of [Manna and Pnueli 1989; Manna and Pnueli 1995]: $\square$ for always in the future, $\diamond$ for eventually in the future, etc.

The operators $\langle A \rangle$, $[A]$, $\langle A \rangle f$, $[A] f$ are called alternating quantifiers. The most basic one is $\langle A \rangle$, stating that $A$ have a strategy to make a path formula true in all runs starting in the current state. The dual operator $[A]$ is defined as $[A] \varphi \equiv \neg \langle A \rangle \neg \varphi$: we usually say that $A$ cannot avoid $\varphi$ from happening. The fair alternating quantifiers $\langle A \rangle f$ and $[A] f$ are similar, but interpreted over all fair runs instead of all runs.

2.2.0.2 Semantics.. Let $S$ be an ADS. The truth relations $S, s \models \varphi$ and $S, \pi \models \psi$ for a state formula $\varphi$ at a state $s$ and for a path formula $\psi$ over a path $\pi$, are defined by mutual induction on the structure of the formula.

$\neg S, s \models p$, for $p$ an assertion, if $s \models p$ in the assertion language;
Game-Theoretic Deductive Verification of a Contract-Signing Protocol

$-S, s \models \langle A \rangle \psi$ if there exist strategies $f_A$ such that, for all $\pi \in out(s, f_A)$, we have $S, \pi \models \psi$;

$-S, s \models [A] \psi$ if, for all strategies $f_A$, there exists an outcome $\pi \in out(s, f_A)$ such that $S, \pi \models \psi$ holds;

$-S, s \models \langle A \rangle_f \psi$ if there exist strategies $f_A$ such that, for all outcomes $\pi \in out(s, f_A)$ such that $\pi \models F_{\Omega \setminus A}$, we have also $\pi \models F_A$ and $S, \pi \models \psi$;

$-S, s \models [A]_f \psi$ if, for all strategies $f_A$, there is at least an outcome $\pi \in out(s, f_A)$ such that $\pi \models F_{\Omega \setminus A}$ and, if $\pi \models F_A$, then also $S, \pi \models \psi$.

Boolean operators distribute over $\models$ in the natural way, both for state and path formulas; LTL operators are evaluated over path formulas in the usual way. We say that $S \models p \Rightarrow \varphi$ when $S, s \models \varphi$ for all states $s \in \Sigma$ satisfying $p$.

2.3 Proof System

Our proof system operates on statements of the form $S \models p \Rightarrow \langle A \rangle \varphi$, where $S$ is an ADS, $p$ is an assertion, $\langle A \rangle$ is an alternating quantifier, and $\varphi$ is a path formula in positive normal form. (Every ATL* formula can be put in positive normal form, that is, all negations have been pushed to the assertion level, in the same way used for propositional logic and LTL, and rewriting $\neg \langle A \rangle \varphi$ to $\langle A \rangle \neg \varphi$ etc.) The rules of the proof system, shown in Figure 1, can be classified into four groups:

1. A basic state rule, which reduces all statements to the form $p \Rightarrow \langle A \rangle \varphi(p)$, where $p$ is an LTL formula and $\varphi$ a state formula appearing with positive polarity. The rule requires an auxiliary assertion $q$ that approximates the set of states on which $\psi$ holds.

2. A basic path rule, which reduces the LTL formula $\varphi$ to an assertion while extending the system $S$ by synchronously composing it with an automaton for $\varphi$. The role of the automaton $A_\varphi$ is to act as a temporal tester, that is, to observe the evolution of $\varphi$ on the ADS [Kesten and Pnueli 2005].

3. A history rule, which augments the system with an extra player $h$ and extra history variables such that, in the new system, $A$ can win $\varphi$ with memoryless strategies from its winning set.

4. Assertion rules, which reduce the validity of statements of the form $p \Rightarrow \langle A \rangle q$, where $p$ and $q$ are assertions, and $\langle A \rangle$ is one of the four alternating quantifiers $\langle A \rangle$, $\langle A \rangle_f$, $\langle A \rangle_t$, $\langle A \rangle_{tt}$, to first-order verification conditions. To apply the rule we first make the fairness conditions explicit, which produces a statement of the form

$$p \Rightarrow \langle A \rangle \bigwedge_{i} (\bigwedge_{i} J_{i}^1 \land \ldots \land J_{i}^k \rightarrow \infty q_i) .$$

which is dealt with directly by the rule POS-ASSERTION for the quantifier $\langle A \rangle$ and a corresponding rule NEG-ASSERTION for $[A]$. The intuition behind this rule is similar to that for the analogous rules for LTL [Manna and Pnueli 1989]. The ranking functions enforce progress towards realizing the $q_i$, and the $r_i^j$ denote regions inside which the ranking functions are constant. The verification conditions assure that, assuming fairness of the adversaries, the players $A$ can eventually force the game out of these regions, and thus decrease the ranking.
M. Slanina, H.B. Sipma, Z. Manna

\[
\begin{align*}
\text{basic-state:} & \quad p \Rightarrow \varphi(q) \\
& \quad q \Rightarrow \psi \\
& \quad p \Rightarrow \varphi(\psi)
\end{align*}
\]

\[
\begin{align*}
\text{basic-path:} & \quad S, S_p = p \land q = q_0 \Rightarrow \langle A, a \varphi \rangle T \\
& \quad S \models p \Rightarrow \langle A \rangle \varphi \\
& \quad S \models p \Rightarrow \langle A \rangle \varphi
\end{align*}
\]

\[
\begin{align*}
\text{history:} & \quad S[h, V_h] = p \Rightarrow \langle A, h \rangle \varphi \\
& \quad S \models p \Rightarrow \langle A \rangle \varphi
\end{align*}
\]

\[
\begin{align*}
\text{pos-assertion:} & \quad p \Rightarrow \wedge_{i=1}^{n} \neg r^i \\
& \quad \forall i \in \{1, \ldots, n\}: \Rightarrow \bigvee_{j=1}^{n} \neg r^j_i \\
& \quad \forall \{j_i \mid i \in \{1, \ldots, n\}\}: \\
& \quad \Lambda_{i=1}^{n} (r^i_j \land \delta^i_j = a_i) \Rightarrow \text{cpre} \Lambda_{i=1}^{n} \left[ \left( r^i_j \wedge q_i \right) \vee \bigvee_{k=1}^{k} (r^i_j \wedge \delta^i_j < a_i) \vee \left( r^i_j \wedge \delta^i_j \leq a_i \wedge a_i = 0 \right) \right] \\
& \quad p \Rightarrow \langle A \rangle \Lambda_{i=1}^{n} (\infty J^i_j \land \ldots \land \infty J^i_k \rightarrow \infty q_i)
\end{align*}
\]

Fig. 1. The four proof rules of the proof system

The application of these rules results in a set of verification conditions to be proved in the underlying theory of the system plus the cpre predicate transformers. Completeness of the proof system is relative to validities in the first-order logic, with fixpoints and cpre, of the underlying theory—the same as required for relative completeness for LTL proof systems [Manna and Pnueli 1989]. A more detailed description of the proof rules can be found in [Slanina et al. 2006].

3. CASE STUDY: ASW PROTOCOL

We demonstrate the application of our formalism and proof system on negotiating protocols by presenting a deductive proof of fairness of the contract-signing protocol first proposed by Asokan et al. [1998] (henceforth called the ASW protocol). This protocol was earlier analyzed by Kremer and Raskin using model checking [Kremer and Raskin 2002].

3.1 Protocol and Modeling Assumptions

Figure 2 shows the three main components of the protocol in the notation common for describing protocols. In this protocol A and B are the contract signers, with A the originator, and B the recipient; T is the trusted third party; t is the contract text; \( N_X \) is a secret nonce generated by agent X; \( \text{Sig}_X(m) \), where \( X \in \{A, B, T\} \) is a signature of message \( m \) with X’s public key; and \( h \) is a one-way collision-resistant hash function.

We model this protocol, similarly to [Kremer and Raskin 2002], as the following ADS: The agents are A, B, and T as above, and C which represents the communication channels. The agents A and B are potentially dishonest. The corresponding honest agents, which always act according to the protocol, are denoted \( A_h \) and \( B_h \). For each agent \( X \in \{A, B, T\} \) and every message \( m \in \{m_1, m_2, \ldots\} \), \( K(X, m) \) stands for \( X \, \text{knows} \, m \) if \( X \in \{A, B, T\} \), and \( C \, \text{contains} \, m \), that is, \( m \, \text{is in transit} \).
Main protocol:

\[
\begin{array}{cccc}
&M1 & A & \frac{m_1 = \text{Sig}_A(A;B;T;h(N_A))}{\text{give_up?quit}} & B \\
&M2 & M_1 & \frac{\text{give_up?abort}}{\text{give_up?quit}} & M_1 \\
&M3 & M_2 & \frac{m_2 = \text{Sig}_B(h(m_1);h(N_B))}{\text{give_up?resolve}} & M_2 \\
&M4 & M_3 & \frac{m_3 = N_A}{\text{give_up?resolve}} & M_3 \\
&M5 & M_4 & \frac{m_4 = N_B}{\text{give_up?resolve}} & M_4 \\
\end{array}
\]

Abort protocol:

\[
\begin{array}{cccc}
&A & \frac{a_1 = \text{Sig}_A(\text{abort}, m_1)}{\text{if resolved}(m_2)} & T \\
&A & \frac{\text{if resolved}(m_2)}{\text{then} a_2 := \text{Sig}_T(m_1, m_2)} & T \\
&A & \frac{\text{else} \ \text{aborted}(m_1) := T}{a_2 := \text{Sig}_T(\text{aborted}, a_1)} & T \\
&A & \frac{a_2}{\text{if resolved}(m_2)} & T \\
&A & \frac{\text{if resolved}(m_2)}{\text{then} r_2 := \text{Sig}_T(\text{aborted}, a_1)} & T \\
&A & \frac{\text{else} \ \text{resolved}(m_1) := T}{r_2 := \text{Sig}_T(m_1, m_2)} & T \\
\end{array}
\]

Resolve protocol:

\[
\begin{array}{cccc}
&R & \frac{r_1 = (m_1, m_2)}{\text{if aborted}(m_1)} & T \\
&R & \frac{\text{if aborted}(m_1)}{\text{then} r_2 := \text{Sig}_T(\text{aborted}, a_1)} & T \\
&R & \frac{\text{else} \ \text{resolved}(m_1) := T}{r_2 := \text{Sig}_T(m_1, m_2)} & T \\
&R & \frac{r_2}{\text{if resolved}(m_1)} & T \\
&R & \frac{\text{if resolved}(m_1)}{\text{then} r_2 := \text{Sig}_T(\text{aborted}, a_1)} & T \\
&R & \frac{\text{else} \ \text{resolved}(m_1) := T}{r_2 := \text{Sig}_T(m_1, m_2)} & T \\
\end{array}
\]

Fig. 2. ASW protocol

if \(X = C\).

In the construction of the model we make the following assumptions: (1) The channel \(C\) is resilient: every message is eventually delivered. This is the only assumption on the communication channel. (2) A dishonest agent has control over \(C\). (3) \(C\) does not necessarily deliver messages in order. Furthermore, we assume perfect public key cryptography. We encode this assumption in the transition model as follows: An agent can only send certain messages after he has received (knows) all the information necessary to compose it. This can be encoded either as a propositional theory on the \(K(X, m)\) atoms or as a first-order theory of \(K, \text{Sig}_T, N_f, \text{etc}, \) that covers the propositional theory. For this case study we use the former. Our formalism and proof system, however, also allows the latter.

3.2 Modeling Language

To describe the protocol as an ADS we use a somewhat simpler modeling language than the representation introduced in Section 2 to ease readability. This modeling language, similar to the one used in [Alur et al. 1998], uses \textit{guarded transitions} to represent the actions of the agents. Formally, a \textit{guarded transition system} is a tuple \((\Omega, V, T, \mathcal{F})\), where:

\(- \Omega \) is a finite set of agents;
\( V \) is a finite set of typed variables.

\( \mathcal{T} = \{ T_a \mid a \in \Omega \} \) is a finite set of transitions, partitioned among agents (transitions in \( T_a \) are controlled by ("belong to") agent \( a \)). A transition \( \tau \in \mathcal{T} \) has the form \( \tau : G \rightarrow \bar{x} := \bar{t} \), where \( G \) is the guard, a boolean expression over \( V \), and \( \bar{x} := \bar{t} \) is an assignment of a list of expressions \( \bar{t} \) over \( V \) to a list of variables \( \bar{x} \) from \( V \).

\( \mathcal{J} \subseteq \mathcal{T} \) is the set of just transitions.

An extra player, the scheduler, chooses, at every step, which agent executes. That agent then chooses one of its enabled transitions and executes it by simultaneously evaluating the expressions in \( \bar{t} \) and assigning them to \( \bar{x} \), leaving all other variables unchanged.

To simplify the translation of guarded transition systems into an ADS, we make the following two assumptions on the model, which hold of our model of the ASW protocol: (1) For every agent \( a \), the set \( T_a \) contains an idling transition, that is, a transition that is always enabled and leaves all variables unchanged; (2) The set \( \mathcal{J} \) of just transitions only contains transitions \( \tau : G \rightarrow \bar{x} := \bar{t} \) such that \( G \rightarrow \neg G[\bar{x}/\bar{t}] \) is valid, that is, all just transitions disable their own guard when taken. Under these assumptions we associate with a guarded transition system \( \langle \Omega, V, \mathcal{T}, \mathcal{J} \rangle \) the ADS \( \langle \Omega \cup \{ z \}, V_S, V_{\Omega}, \xi, \chi, \mathcal{F} \rangle \), where:

- \( z \) is a fresh agent called the scheduler;
- \( V_S = V \cup \{ \lambda \} \), where \( \lambda \) is a fresh variable of type \( \Omega \) denoting the last scheduled agent;
- \( V_a = T_a \) for \( a \in \Omega \), i.e., the choices of a regular player are on which transition to execute;
- \( V_z = \Omega \), i.e., the choice of the scheduler \( z \) is on which agent to schedule;
- \( \xi_a(s, \tau) \) if and only if \( s \models G_\tau \); \( \xi_z(s, a) \equiv T \);
- \( \chi(s, \tau_{a_1}, \ldots, \tau_{a_n}, a_i, s') \) iff \( \tau_{a_i} \) is \( G : \bar{x} := \bar{t} \) and \( s' = s[\bar{x} \mapsto \bar{t}(s), \lambda \mapsto a_i] \);
- \( \mathcal{F}_a = \bigwedge_{\tau \in \mathcal{J}_a} \neg G_\tau \) for \( a \in \Omega \), where \( \mathcal{J}_a = \mathcal{J} \cap T_a \); \( \mathcal{F}_z = \bigwedge_{a \in \Omega} \neg \omega(\lambda = a) \).

The condition \( \xi_z(s, a) \equiv T \) reflects that the scheduler is free to choose any agent to execute next. This is possible because we assumed that every player has an idling transition that is always enabled, so the scheduler does not have to check whether an agent has available moves before it schedules it. Because of our assumption of self-disabling transitions the simple fairness condition \( \mathcal{F}_a \) suffices to capture the justice requirement of the guarded transitions. The fairness condition \( \mathcal{F}_z \) states that every agent must be scheduled infinitely many times.

As mentioned before, the cpre (controllable predecessors) predicate can often be simplified when dealing with an ADS with a specific structure. In the case of an
Transitions “owned by” $A_h$:

- $m_1$.send : $M_1 \land K(A, m_1) \rightarrow M_2 \land K(C, m_1)$
- $m_2$.receive : $M_2 \land K(A, m_2) \rightarrow M_3$
- $m_3$.send : $M_3 \land K(A, m_2) \rightarrow M_4 \land K(C, m_3)$
- $m_4$.receive : $M_4 \land K(A, m_4) \rightarrow F$
- $a_{\text{abort}} : M_2 \land K(A, a_1) \rightarrow A_1$
- $a_{\text{resolve}} : A_1 \rightarrow A_2 \land K(C, a_1)$
- $a_2$.receive : $A_2 \land (K(A, T.A.\text{aborted}) \lor K(A, T.A.\text{resolved})) \rightarrow F$
- $A_{\text{resolve}} : M_4 \rightarrow R_1$
- $A.r_1$.send : $R_1 \land K(A, m_2) \rightarrow R_2 \land K(C, A.r_1)$
- $A.r_2$.send : $R_2 \land (K(A, T.A.\text{aborted}) \lor K(A, T.A.\text{resolved})) \rightarrow F$

Transitions “owned by” $B$:

- $\text{give}_\uparrow : \rightarrow F$
- $m_2$.send : $K(B, m_1) \rightarrow K(C, m_2)$
- $m_4$.send : $\rightarrow K(C, m_4)$
- $B.r_1$.send : $K(B, m_1) \rightarrow K(C, B.r_1)$

Fig. 3. Guarded transitions of honest agent $A_h$ and potentially dishonest agent $B$.

ADS derived from a guarded transition system, the cpre can be reduced to

$$
cpre_A(\varphi) \equiv \begin{cases} 
\bigwedge_{a \in A} \bigvee_{\tau \in T_a} (G_\tau \land \varphi[\vec{x}_\tau/\vec{t}_\tau, \lambda/a]) & \text{if } z \notin A \\
\bigwedge_{a \in \Omega \backslash A} \bigwedge_{\tau \in T_a} (G_\tau \rightarrow \varphi[\vec{x}_\tau/\vec{t}_\tau, \lambda/a]) & \text{if } z \in A
\end{cases}
$$

3.3 Model of the ASW Protocol

We now represent the ASW protocol as a guarded transition system. The set of agents is $\Omega = \{A, B, T, C\}$. The set of variables $V$ includes one boolean variable for each $K(X, m)$ proposition, as discussed above, and three state variables $A.\text{state}$, $B.\text{state}$, and $T.\text{state}$. The variables $A.\text{state}$ and $B.\text{state}$ range over the values $\{M_1, M_2, M_3, M_4, A_1, A_2, R_1, R_2, F\}$, where $F$ stands for finished and denotes termination, while $T.\text{state}$ ranges over $\{\text{clean, resolved, aborted}\}$. The transitions of the four agents are shown in Figures 3 and 4. For conciseness, we use the following abbreviated notation for checking and setting of state variables: for agent $a$ and state value $x, x (\neg x)$ appearing in a guard stands for $a.\text{state} = x (a.\text{state} \neq x)$, and $x$ appearing in the update stands for $a.\text{state} := x$. Similarly for propositions $p$ appearing in the update, $p$ stands for $p := T$ and $\neg p$ stands for $p := F$. All transitions owned by $C$ and $T$ are just, except for the idling transitions. None of the transitions owned by $A_h$ and $B$ is just. We use simpler, but equivalent, fairness conditions for $T$ than those following directly from the translation in the previous
The proof system described in Section 2 is sound and relatively complete, as proved in [Slanina et al. 2006]. In practice, however, it is often convenient to augment the original proof system.

4.1 Auxiliary Proof Rules

The proof system described in Section 4 is sound and relatively complete, as proved in [Slanina et al. 2006]. In practice, however, it is often convenient to augment the original proof system.

In the proof of fairness of the ASW protocol, presented in the next section, we...
use two additional rules. The first one,

\[
\text{LEFT-\lor}:
\begin{align*}
p_1 & \Rightarrow \varphi \\
p_2 & \Rightarrow \varphi \\
p_1 \lor p_2 & \Rightarrow \varphi
\end{align*}
\]

is a simple splitting rule similar to those in Gentzen-style proof systems that allows decomposition into multiple subgoals. The second rule,

\[
\text{JUST-TERM}:
\begin{align*}
p & \Rightarrow q \lor r \\
r & \Rightarrow \bigvee_i r_i \\
\text{For all } i: \\
\ r_i \land \delta_i = a \Rightarrow \text{cpre}_A(q \lor \bigvee_j (r_j \land \delta_j \triangleleft a) \lor (r_i \land \delta_i \preceq a \land \neg J_i)) \\
p & \Rightarrow \langle A \rangle (\bigwedge_i \infty J_i \rightarrow q)
\end{align*}
\]

is a simplification of rule \text{POS-ASSERTION}, specialized for properties of the form \( p \Rightarrow \langle A \rangle (\bigwedge_i \infty J_i \rightarrow q) \), based on rule \text{s-RESP-CONTROL} from [Slanina 2002].

The rule requires an intermediate assertion \( r \) that holds at \( p \)-states and from which \( A \) can force the system to either remain within \( r \)-states or reach the goal \( q \). Within the \( r \)-states progress towards the goal is enforced by the requirement that at each step a ranking function be decreased or a specific fairness requirement be falsified. Both conditions preclude that the system can stay forever in an \( r \)-state.

### 4.2 Representing Rule Applications with Diagrams

The application of proof rules generally requires several intermediate assertions and ranking functions and produces a large number of first-order verification conditions. Verification diagrams [Manna et al. 1998] provide a precise and concise representation to visualize such proofs. They have been used successfully to represent proofs of temporal properties of reactive systems. Here we introduce a new type of verification diagram that can represent proofs of properties of the form \( p \Rightarrow \langle A \rangle (\bigwedge_i \infty J_i \rightarrow q) \) over ADS’s.

**Definition 1 Diagram.** Given an ADS \( S \), a set of agents \( A \subseteq \Omega \), an ordered list of fairness conditions \( J_1, \ldots, J_k \), and a well-founded order \( \langle \mathcal{A}, \triangleleft \rangle \), a diagram is a digraph with nodes \( N \), edges \( E \), terminal nodes \( N_T \subseteq N \), and three labeling functions on the nodes – for every node \( n \in N \):

1. \( \mu(n) \), an assertion in the language of \( S \);
2. \( \delta(n) : \Sigma \rightarrow \mathcal{A} \), a ranking function defined on the states satisfying \( \mu(n) \);
3. \( j(n) \in \{1, \ldots, k\} \), denoting an associated fairness condition, with \( j(n) \) minimal for \( n \in N_T \).

Associated with each nonterminal node \( n \in N \setminus N_T \) is the verification condition

\[
\mu(n) \land \delta(n) = a \Rightarrow \text{cpre} \left\{ \bigvee_{m \in \text{succ}(n)} (\mu(m) \land \delta(m) \triangleleft a) \lor \bigvee_{m \in \text{succ}(n)} (\mu(m) \land \delta(m) = a \land \neg J_j(m)) \right\}
\]
where \( \text{succ}(n) = \{ m \in N \mid \langle n, m \rangle \in E \} \) and \( \text{succ}_L(n) = \{ m \in \text{succ}(n) \mid j(m) = j(n) \} \), called the “level” successors.

A diagram is called valid if all its associated verification conditions are valid.

**Proposition 2.** A valid diagram \( (N, E, N_T, \mu, \delta, j) \) over an ADS \( S \) and agents \( A \) is a proof of \( S \vdash p \Rightarrow \langle A \rangle_{k \in 1} \infty (J_i \rightarrow \Diamond q) \) if \( j(n) \in \{1, \ldots, k\} \) for all \( n \in N \) and

\[
p \rightarrow \bigvee_{n \in N} \mu(n) \quad \text{and} \quad \bigwedge_{n \in N_T} (\mu(n) \rightarrow q)
\]

In the proof in the next section we label edges with transition names indicating what actions \( A \) can take to satisfy the verification conditions. These names are not officially part of the diagram; they can be discovered by a theorem prover or decision procedure checking the verification condition.

5. PROVING FAIRNESS FOR ALICE

Fairness for agent \( A \) is expressed by the formula

\[
A_h \parallel B \parallel T \parallel C \models \Theta \Rightarrow [B, C, z]_f (\text{contract}_B \rightarrow \langle A_h \rangle_f \Diamond \text{contract}_A)
\]

where \( \Theta \) is the initial condition and

\[
\text{contract}_A \equiv (K(A, m_2) \land K(A, m_4)) \lor K(A, T.A.\text{resolved})
\]

\[
\text{contract}_B \equiv (K(B, m_1) \land K(B, m_3)) \lor K(B, T.B.\text{resolved})
\]

Before we apply the proof rules we first establish some global invariants of the protocol, that is, properties of the form \( S \vdash p \Rightarrow [\Omega] \Box q \), where \( p \) and \( q \) are assertions (also written \( p \Rightarrow \Box q \), or \( \Box q \) when \( p \) is clear from the context). Global invariants form the basis of any deductive proof. Methods for generation and verification of invariants have been studied extensively (e.g., [Manna and Pnueli 1995; Björner et al. 1997]) and these methods apply directly to ADS’s, so we will not elaborate on them here. In the remainder of the proof we will assume that the following invariants hold and use them in the discharge of the verification conditions:

\[
\Box (K(A, m_2) \rightarrow A.\text{state} \neq M1) \quad (3)
\]

\[
\Box \left( A.\text{state} = R2 \land T.\text{state} \neq \text{aborted} \rightarrow K(C, A.r_1) \lor K(T, A.r_1) \lor K(C, T.A.\text{resolved}) \right) \quad (4)
\]

\[
\Box K(A, m_1) \quad (5)
\]

\[
\Box K(A, a_1) \quad (6)
\]

\[
\Box (T.\text{state} = \text{resolved} \rightarrow A.\text{state} \neq M1) \quad (7)
\]

\[
\Box (\text{A.state} \in \{M3, M4\} \rightarrow K(A, m_2)) \quad (8)
\]

The first step of the proof is to apply rule basic-state with intermediate assertion

\[
\text{poss_contract}_A \equiv \left( K(B, m_1) \land K(B, m_3) \land K(A, m_1) \land K(A, m_2) \land \text{recoverable} \land K(B, T.B.\text{resolved}) \land T.\text{state} = \text{resolved} \land \text{recoverable} \right)
\]
where

\[
\text{recoverable} \equiv \left( \neg K(C, a_1) \land T.\text{state} \neq \text{aborted} \land A.\text{state} \notin \{A1, A2, F\} \right)
\]

which produces the two subgoals

\[
\Theta \Rightarrow [B, C, z] \square (\text{contract}_B \rightarrow \text{poss}\_\text{contract}_A) \quad (9)
\]

\[
\text{poss}\_\text{contract}_A \Rightarrow \langle A_h \rangle_f \diamond \text{contract}_A \quad (10)
\]

Discovering intermediate assertions is the creative and, in general, hard part of the proof that requires insight into the system. In this case, \text{contract}_B was strengthened into an assertion that characterizes states from which it is possible for agent \(A_h\) to guarantee \text{contract}_A eventually.

Goal (9) follows immediately from the global invariant

\[
\square (\text{contract}_B \rightarrow \text{poss}\_\text{contract}_A)
\]

Goal (10) can be split using rule \text{left-}^\top (\text{Sect. 4}) into the two new goals

\[
\text{poss}\_\text{contract}_A \Rightarrow \langle A_h \rangle_f \diamond \text{contract}_A \quad (11)
\]

\[
\text{poss}\_\text{contract}_A \Rightarrow \langle A_h \rangle_f \diamond \text{contract}_A \quad (12)
\]

where

\[
\text{poss}\_\text{contract}_A,1 \equiv K(B, m_1) \land K(B, m_3) \land K(A, m_1) \land K(A, m_2) \land \text{recoverable}
\]

\[
\text{poss}\_\text{contract}_A,2 \equiv K(B, TB.\text{resolved}) \land T.\text{state} = \text{resolved} \land \text{recoverable}
\]

To prove goal (11) we first make the fairness conditions explicit,

\[
\text{poss}\_\text{contract}_A,1 \Rightarrow \langle A_h \rangle_f \left( F_T \land F_C \land F_z \rightarrow \diamond \text{contract}_A \right)
\]

and then apply rule \text{just-term} (\text{Sect. 4}) with auxiliary assertions and corresponding fairness conditions

\[
\begin{align*}
r_1 &\equiv \varphi_R \land A.\text{state} \in \{M2, M3, M4, R1\} \quad J_1 \equiv \lambda = A_h \\
r_2 &\equiv \varphi_R \land K(C, A.r_1) \quad J_2 \equiv \neg K(C, A.r_1) \\
r_3 &\equiv \varphi_R \land K(T, A.r_1) \quad J_3 \equiv \neg K(T, A.r_1) \\
r_4 &\equiv \varphi_R \land K(C, TA.\text{resolved}) \quad J_4 \equiv \neg K(C, TA.\text{resolved})
\end{align*}
\]

with \(\varphi_R \equiv K(A, m_2) \land \text{recoverable}\). We can choose \(r_i \equiv F\) for all other \(J_i\)'s. The ranking functions, with range \(\{0,\ldots,11\}\), are defined as follows:

\[
\begin{align*}
\delta_1 &= \begin{cases} 
11 & \text{if } A.\text{state} = M2 \land \lambda = A_h \\
9 & \text{if } A.\text{state} = M3 \land \lambda = A_h \\
7 & \text{if } A.\text{state} = M4 \land \lambda = A_h \\
5 & \text{if } A.\text{state} = R1 \land \lambda = A_h \\
10 & \text{if } A.\text{state} = M2 \land \lambda \neq A_h \\
8 & \text{if } A.\text{state} = M3 \land \lambda \neq A_h \\
6 & \text{if } A.\text{state} = M4 \land \lambda \neq A_h \\
4 & \text{if } A.\text{state} = R1 \land \lambda \neq A_h
\end{cases} \\
\delta_2 &= 3 \\
\delta_3 &= 2 \\
\delta_4 &= 1
\end{align*}
\]

With these assertions, fairness conditions, and ranking functions the resulting verification conditions are valid relative to the global invariants established before,
Fig. 5. A diagram for goal (13). Each node \( n \) is labeled with the assertion \( \mu(n) \). The expression \( M_2 \) is an abbreviation for \( A\_state = M_2 \), and so on for the other values. The expression in the outer box is common and considered conjoined to all the inner ones. The ranking function is defined by \( \delta(n_i) = i \). The fairness index \( j(n_i) \) is 1 for \( n_4, \ldots, n_{11} \), 2 for \( n_3 \), 3 for \( n_2 \), and 4 for \( n_1 \). Node \( n_0 \) is the only terminal node.

which completes the proof of (11). The diagram corresponding to this proof is shown in Figure 5. The verification conditions associated with the diagram are the same as those produced by rule JUST-TERM.

The proof of (12) proceeds along the same lines. After making the fairness conditions explicit, we apply rule JUST-TERM with intermediate assertions \( r_1, \ldots, r_4 \) and fairness conditions \( J_1, \ldots, J_4 \) as in the proof of (11) and the additional ones

\[
\begin{align*}
r_5 &\equiv \varphi_T \land K(C, a_1) \quad J_5 \equiv \neg K(C, a_1) \\
r_6 &\equiv \varphi_T \land K(T, a_1) \quad J_6 \equiv \neg K(T, a_1)
\end{align*}
\]

where \( \varphi_T \equiv T\_state = \text{resolved} \land \text{recoverable} \), and we can choose \( r_i \equiv \top \) for all other \( J_i \)'s. Again, all verification conditions hold relative to the invariants established...
before. A diagram representation of the proof is shown in Figure 6.

![Diagram](image)

**Fig. 6.** A diagram for the proof of (12). The conventions are the same as those of Fig. 5. The fairness index $j(n_1)$ is 1 for $n_4, \ldots, n_9$ and $n_{12}, \ldots, n_{15}$, $j(n_3) = 2$, $j(n_2) = 3$, $j(n_1) = 4$, $j(n_{11}) = 5$, and $j(n_{10}) = 6$. Node $n_0$ is the only terminal node.

6. **CONCLUSIONS**

We have presented an example of a deductive proof of the ASW contract-signing protocol using the game-theoretic models and logics of [Slanina et al. 2006]. We believe that this approach is of interest to the security community for several reasons.
(1) A game-theoretic framework is the only existing framework that faithfully models the mixture of competition and cooperation among agents that is an essential ingredient in many security systems. (2) Deductive verification allows the analysis of infinite-state and parameterized systems, which cannot be verified using model checking. (3) A deductive proof provides a certificate, that is, independently checkable evidence that also provides insight into the modeling assumptions and the reasons for the system’s correctness. (4) Deductive verification is the only option when using models with imperfect information, even for finite-state systems. In many cases such models greatly facilitate a faithful modeling of security protocols, as explicitly encoding each agent’s knowledge can be avoided.

We are currently developing a framework, including a computational model and a proof system, for the analysis of systems with imperfect information. We are also planning to perform more case studies, including the analysis of parameterized systems such as those studied in [Chadha et al. 2006].

REFERENCES


REACT-TR-2006-03, August 2006
CHANGE LOG