Game Analysis of Abuse-free Contract Signing

Steve Kremer Jean-François Raskin

Département d'Informatique
Faculté des Sciences
Université Libre de Bruxelles, Belgium
skremer@ulb.ac.be Jean-Francois.Raskin@ulb.ac.be

Abstract

In this paper we report on the verification of two contract signing protocols. Our verification method is based on the idea of modeling those protocols as games, and reasoning about their properties as strategies for players. We use the formal model of alternating transition systems to represent the protocols and alternating-time temporal logic to specify properties. The paper focuses on the verification of abuse-freeness, relates this property to the balance property, previously studied using two other formalisms, shows some ambugities in the definition of abuse-freeness and proposes a new, stronger definition. Formal methods are not only useful here to verify automatically the protocols but also to better understand their requirements (balance and abuse-freeness are quite complicated and subtle properties).

1 Introduction

Digital contract signing is an important part of electronic commerce. In comparison to classical, paper-based contract signing, the problem of digitally signing a contract over a network is more complicated. If, for instance, Alice and Bob, want to sign a contract, and if they don't trust each other, none of them wants to be the first one to sign the contract, as the other one could refuse to do so after having got the first one's signature. This asymmetry appears in a similar way in several related problems, such as fair exchange protocols, fair non-repudiation protocols, certified e-mail protocols, etc.

In 1980, Even and Yacobi [7] showed that no deterministic contract signing protocol exists, without the participation of a trusted third party (TTP). A rather simple solution consists in using a trusted third party as an intermediary. Both, Alice and Bob send their respective signature to the TTP, who collects the signatures and forwards them to the other entity. However, due to the communication and computation bottleneck created at the TTP, this solution has been considered as inefficient. Therefore, randomized protocols, as well as solutions based on gradual information exchange, have been proposed. Protocols based on gradual exchange generally require that all involved entities have the same or related computational power. This hypothesis is however rather unrealistic. Probabilistic protocols, on the other hand, have to increase the number of messages to decrease the probability that some entity can cheat and so are inefficient. More recently, Micali [14] and Asokan et al. [2] introduced the optimistic approach. The idea is that a trusted third party only intervenes when a problem arises, e.g. an entity is trying to cheat or a network failure occurs at a crucial moment during the protocol. In such protocols the TTP is said to be offline. Such protocols generally consist in a main protocol and one or several subprotocols. The main protocol is executed by Alice and Bob in order to exchange their signatures on the contract. The subprotocols are used to contact the TTP in order to force a successful outcome or to abort the protocol. In this paper we are interested in optimistic protocols, as they got most of the attention during the last years.

Generally, we require that a contract signing protocol respects several properties. A first property is *fairness*. A contract signing protocol is fair if at the end of the protocol either all, both Alice and Bob got the other one's signatures or none of them got any valuable information. Intuitively, this property prevents an entity from cheating the other one. A second property is *timeliness*. A protocol respects timeliness, if at any moment in the protocol, each entity can reach a point where it can stop the protocol, achieving fairness. This is needed to prevent a situation, where one entity does not know whether he can stop the protocol without loosing fairness, or whether he still has to wait for a message to arrive. Both fairness and timeliness are classical properties that are also requested in fair exchange, certified e-mail and fair non-repudiation protocols. Recently, Garay et al. introduced a new property, that is specific to contract signing: abuse-freeness. A protocol is abuse-free if neither Alice nor Bob, has the power to prove to an external party, Charlie, that he can either successfully finish or stop an engaged contract signing protocol. Suppose that a contract signing protocol is not abuse-free, and that Alice has the power to decide of the outcome of the protocol. If, for instance, Alice wants to sell a house to Charlie, she could engage a contract with Bob, having the mere goal to force Charlie to increase his offer. It is clear that a protocol that is not abuse-free gives an undesirable advantage to Alice.

Contract signing and related protocols did not receive as much attention as other security protocols, e.g. authentication and key exchange protocols. History showed that security protocols often contain subtle errors. The probably most notorious example is the Needham-Schroeder public-key protocol. It was believed secure during several years, until Lowe [11] found a flaw in this protocol. Rapidly, the use of formal methods to analyse and verify security protocols has been recognized to be of crucial importance. A lot of different methods have been used. Some are based on believe logics, such as BAN [5]. Other methods are based on general purpose model-checkers, e.g. FDR [12]. Theorem provers, such as Isabelle [15], have also been used, as well as special purpose tools, designed for the analysis of security protocols, e.g. the NRL protocol analyzer [13]. Only few works trying to apply formal methods on fair exchange and related protocols have been realized until now. There has been some work on a non-repudiation protocol by Schneider, using CSP [16]. The proofs are hand generated and require in depth knowledge of CSP. Zhou and Gollmann [20] also applied the SVO believe logic to non-repudiation protocols. However, they studied the validity of the non-repudiation evidences. This topic is specific to non-repudiation protocols and less interesting to contract signing protocols. Boyd and Kearny [4] used the specification animation tool Possum to analyze a fair exchange protocol. The tool gives the possibility to step through the protocol and examine the consequences of various actions. Shmatikov and Mitchell [17, 18, 19] used the finite state tool Murφ to examine fair exchange and contract signing protocols. Chadha et al. [6] used the multiset rewriting formalism combined to inductive methods to analyse abuse-free contract signing. These two methods will be discussed in more details in the paper. Recently, Kremer and Raskin also used a temporal logic, ATL, having game semantics and the corresponding model checker MOCHA to analyse several non-repudiation and fair exchange protocols. In this paper, we show how this method can be used for analyzing contract signing protocols and model abuse-freeness in a natural way.

As shown in [10], games can be used to model accurately exchange protocols. They are particularly interesting, as we have to model the fact that the entities, participating in the protocol, may be dishonest. This is one of the major differences with other classical security protocols. We do not model an external intruder, but malicious participants. Using the notions of games and strategies, we can in a natural way model cooperative behaviors, as well as adversarial behaviors and include them in our formal model. A second important difference is the fact that optimistic exchange protocols are not ping-pong protocols, but are branching. Hence, at some moment in the protocol Alice and Bob have to choose to continue the main protocol or to launch a subprotocol by contacting the TTP. Moreover, subprotcols may be executed at a moment not foreseen by the protocol and lead to subtle flaws. Therefore, no predefined order is given to these actions when specifying the protocol.

In [10], Kremer and Raskin show for instance that fairness can be rephrased naturally in terms of strategies. We say that the protocol is fair for Alice, if "Bob in collaboration with the communication channels does not have a strategy against Alice to obtain Alice's signature on the contract, without Alice having a strategy to obtain Bob's signature". This example illustrates how cooperative behavior—modeling the fact that Bob controls the communication channels—and adversarial behavior—Alice is not going to help Bob to cheat her—are useful in this context. In this paper we will above all focus on the analysis of abuse-freeness. This property has known some difficulties to be modeled in other formalisms (cf [19]), and has however a natural representation in form of strategies. Moreover, we here present the first complete modeling of abuse-freeness for verification purposes. In [19] and [6] a variant of abuse-freeness, called balance, has been studied. We also formally show the relationship between abuse-freeness and balance. Then we note that under certain assumptions, the contract signing protocol proposed by Asokan et al. [3] is actually abuse-free, contrary to what was believed, with respect to the classical definition of abuse-freeness. However, although the protocol is abuse-free, intuitively, a problem remains in this protocol. Therefore, we slightly modify the definition of abuse-freeness, keeping the same intuition, and show that the protocol is not abuse-free anymore with the new definition, which seems to us to better reflect the initial idea of abuse-freeness.

Structure of the paper. The paper will be organized as follows. In section 2 we describe two contract signing protocols, that we consider for analysis during the remaining of the paper. In section 3, we report on Shmatikov et al.'s work using Murφ, as well as on Chadha et al.'s work in the MSR formalism. In section 4 we introduce the alternating transition systems, the alternating-time temporal logic, as well as the game-guarded command language and show, in section 5, how this formalism can be used to specify and and analyze the protocols. In section 6 we we model the different properties a contract signing protocol has to respect in terms of strategies and give a formulation in the alternating-time temporal logic ATL. In section 7, we report on the results of our analysis of the two previously described protocols, and discuss a new definition of abuse-freeness. Finally, in section 8, we draw the conclusions.

2 Two optimistic contract signing protocols

In this section we give a brief informal description of two protocols. The first one is the Asokan-Shoup-Waidner protocol, ASW for short. The second one is the Garay-Jakobson-MacKenzie protocol, GJM for short.

2.1 The ASW protocol

The protocol was initially presented in [3]. The protocol consists of a main protocol, an abort protocol and a recovery protocol. We use the following notation to describe the protocol.

- $X \rightarrow Y$: transmission from entity X to entity Y
- $h()$: a collision resistant one-way hash function
- $S_X()$: the signature function of entity X
- t: the contract text
- N_A: a nonce chosen by A
- N_B: a nonce chosen by B

2.1.1 Main protocol

The main protocol (protocol 1) consists of four messages. In the first two messages they express their intention to sign a given contract t. Therefore both Alice and Bob, commit to their respective nonces, N_A and N_B, without however revealing them. In the messages 3 and 4, they exchange the previously commited nonces. The signed commitment and the nonce itself, are considered as an evidence that the contract is signed. Now, we show how each entity has to react in case of problem, i.e. if a given message does not arrive, either because it has been lost or never been sent. If the first message does not arrive the protocol is not considered as started, and so Bob does not have to take any special action. If the second message does not arrive (this can also be due to the fact that the first message has been lost previously), Alice can launch an abort protocol. Note, that at this point Alice cannot launch the recovery protocol, as she does not have any proof that the protocol has been started with Bob. If the third or the fourth message does not arrive, Bob, respectively Alice, can launch a recovery protocol, to force the successful termination of the protocol.

Protocol 1 ASW main protocol

1. A $\rightarrow$ B: $m_1 = S_A(A, B, \mathsf{TTP}, t, h(N_A))$
2. B $\rightarrow$ A: $m_2 = S_B(h(m_1), h(N_B))$
3. A $\rightarrow$ B: $m_3 = N_A$
4. B $\rightarrow$ A: $m_4 = N_B$

2.1.2 Abort protocol

Alice starts the abort protocol (protocol 2) if the second message of the main protocol does not arrive. Therefore she sends a signed abort request to the TTP. Note that the abort protocol and the recovery protocol are mutually exclusive. If a recovery protocol, for the given protocol run, identified by the first message of the main protocol, has been launched before, the TTP will issue an affidavit, replacing the nonce, to witness of a succesfull contract signing. Otherwise, the TTP sends a signed abort token. Note, that this abort token does not mean that the protocol has not been signed. Alice could run a succesful main protocol, and hence get a signed contract with Bob, and run an abort protocol afterwords. The abort token only represents the promise of the TTP not to recover the same protocol run later.

Protocol 2 ASW abort protocol

1. A $\rightarrow$ TTP: $a_1 = S_A(abort, m_1)$

   if reloved(m_1)=true then

2. TTP $\rightarrow$ A: $S_{\mathsf{TTP}}(m_1, m_2)$

   else

   aborted(m_1):=true

3. TTP $\rightarrow$ A: $S_{\mathsf{TTP}}(aborted, a_1)$

2.1.3 Recovery protocol

The recovery protocol is generally launched if the third or the fourth message of the protocol does not arrive. As the protocol is similar wether it is launched by Alice or Bob, we denote by X, the initiator of the protocol. The recovery request, consists in a signed message containing the two first messages of the main protcol. If the given protocol run, has already been aborted, the TTP sends a copy of the abort token. Otherwise, the TTP sends an affidavit to confirm that the contract is signed.

Protocol 3 ASW recovery protocol

1. X $\rightarrow$ TTP: m_1, m_2

   if aborted(m_1)=true then

2. TTP $\rightarrow$ X: $S_{\mathsf{TTP}}(aborted, a_1)$

   else

   recovered(m_1):=true

3. TTP $\rightarrow$ X: $S_{\mathsf{TTP}}(m_1, m_2)$

2.2 The GJM protocol

In [8], Garay, Jakobsson and MacKenzie presented a new protocol, as well as the concept of abuse-free contract signing. The structure of the protocol is very similar to the ASW protocol. However, to achieve abuse-freeness, they introduced a new cryptographic primitive, known as *Private Contract Signatures*, PCS for short. We first define PCS. Then we go on describing the protocol, using the same notation as for the ASW protocol. The protocol, we describe, is not the original one. In [18], Shmatikov and Mitchell discovered a flaw, that could easily be fixed. We here describe the fixed version.

2.2.1 Private Contract Signatures

The aim of PCS is to combine designated verifier signatures and convertible signatures. If Alice uses a PCS to sign a message, destined to Bob, this means that only Bob and a given TTP can verify the authenticity of the signature. Moreover Bob can verify that the TTP can convert the PCS into a conventional signature. In [8], PCS are formally defined. Here we only give an intuitive decription. A $PCS_A(m, B, T)$ is A's signature on message m destined to B, with respect to a third party T. Moreover PCS provides the designated verifier property, i.e. only B can be sure of the authenticity of the PCS. A $PCS_A(m, B, T)$ can be converted to a universally verifiable signature by T.

In [8], two versions of this new primitive are presented. The first one, provides TTP-accountability, i.e. the univerally verifiable signatures produced by A and T are distinguishable. The second one provides TTP-invisibility, i.e. the univerally verifiable signatures produced by A and T are undistinguishable. To simplify notations we only consider the second case, and denote the universally verifiable signature as $S_A(m)$.

2.2.2 Main protocol

In the main protocol (protocol 4), Alice and Bob, first exchange, in messages 1 and 2, their respective PCS on the contract. Then they go on, in messages 3 and 4, to exchange their universally verifiable signatures on the contract. The general structure of the protocol is similar to the ASW protocol. If the second message does not arrive, Alice launches an abort protocol. If the third or the fourth message does not arrive, Bob, respectively Alice, execute the recovery protcol.

Protocol 4 GJM main protocol

1. A → B: $m_1 = PCS_A(t, B, \mathsf{TTP})$
2. B → A: $m_2 = PCS_B(t, A, \mathsf{TTP})$
3. A → B: $S_A(t)$
4. B → A: $S_B(t)$

2.2.3 Abort protocol

The abort protocol (protocol 5 is executed by Alice if the second message of the main protocol does not arrive. Alice therefore sends a signed abort request to the TTP. As in the ASW protocol, the abort and the recovery protocols are mutually exclusive. If the current protocol run has already been recovered, the TTP sends Bob's universally verifiable signature to Alice.

Protocol 5 GJM abort protocol

1. A → TTP: $a_1 = S_A(t, A, B, abort)$

   if recovered=true then

2. TTP → A: $a_2 = S_B(t)$

   else

   aborted:=true

3. TTP → A: $a_3 = S_{\mathsf{TTP}}(a_1)$

2.2.4 Recovery protocol

The recovery protocol (protocol 6) is foreseen to be executed if either the third or the fourth message does not arrive. As the protocol is symmetric for Alice and Bob, we denote by X the initiator of the protocol and by Y the other entity. To start the protocol the initiator sends both the PCS of Alice and Bob to the TTP. If the protocol has been aborted before, the TTP sends a copy of the abort token. Otherwise the TTP converts both PCS, saves the universally verifiable signatures, and sends Y's universally verifiable signature to X.

Protocol 6 GJM recovery protocol

1. X → TTP: $PCS_A(t, B, \mathsf{TTP}), PCS_B(t, A, \mathsf{TTP})$

   if aborted=true then

2. TTP → X: $S_{\mathsf{TTP}}(a_1)$

   else

   recovered:=true

3. TTP → X: $S_Y(t)$

3 Previous analysis of contract signing

In this section, we discuss two previous attempts to analyse digital contract signing protocols. The first method we describe is the one used by Shmatikov and Mitchell [18, 19]. The second one has recently been presented by Chadha, Kanovich and Scedrov [6].

3.1 The Murφ-approach

In [18, 19], Shmatikov and Mitchell use a finite state model checker, Murφ to analyse fair exchange and contract signing protocols. Murφ has previously been used to verify different classes of security protocols. For this analysis they use an external intruder, based on the Dolev-Yao model. This intruder collaborates with Alice or Bob to model that one of them may be dishonest. As we will see later this differs from our approach: we do not model an external intruder, but rather include the fact that Alice or Bob may be willing to cheat directly in our model. Shmatikov and Mitchell applied their method to both protocols that have been presented in the previous section. To verify the different properties they use invariant checking. They notice, that most properties are monotonic, i.e. once they have been violated, they will remain violated until the end of the protocol. This makes it possible for them to only verify final protocol states. However, abuse-freeness is not monotonic, and they have to introduce the concept of a challenger. The idea is that a challenger can flip a coin and request to Alice, respectively Bob, to either force a successful end of the protocol or to end without the possibility to obtain a signed contract, depending on the outcome of the coin flip. Note that Shmatikov and Mitchell do not totally model abuse-freeness. They do not model the concept of proving to an external entity, the power of choosing between successfully ending or not. They only verify the ability of choosing the outcome and not the fact that it can be proven. In this paper we are going to include the concept of proving to an external party into the modeling of abuse-freeness.

When analyzing the ASW protocol they detect a rather strange flaw. It is possible, by the mean of a replay attack to get a different version of a same contract. This is due to the fact that in the original paper a contract is defined as being the tuple (m_1, m_2, N_A, N_B). Indeed, Shmatikov and Mitchell succeed in showing that an external intruder can create a different contract (m_1, m_2, N_A, N'_B), that refers to the same contract text, the same persons Alice and Bob, but is validated by a different nonce. As also noticed in [19], it is not clear whether the authors of the protocol, deliberately permit this attack or not and whether it should be considered as a flaw. Shmatikov and Mitchell propose to fix this flaw, by redefining the two last messages of the protocol as $m_3 = N_A, h(N_B)$ and $m_4 = N_B, h(N_A)$. An alternative way to prevent this "attack" is to change the definition of what is a contract. We propose to make a distinction between the contract defined by the contract text and a *validator* for a given contract, corresponding to the tuple (m_1, m_2, N_A, N_B). Then, the possibility to generate different validators for a same contract should not be considered as a problem.

During the analysis of the GJM protocol, they discovered a more serious flaw. In fact, in the original protocol, when Bob wants to launch a recovery, he has to send Alice's PCS and his own verifiable signature. So if Alice first aborts the protocol, she can eavesdrop Bob's universally verifiable signature, when he tries to recover the protocol. As the recovery is not accepted by the TTP—the protocol has already been aborted—Alice receives a contract signed by Bob, while Bob does not receive Alice's signature on the contract. The flaw can easily be repaired by sending both PCS in the recovery request. In the previous section, we directly presented the fixed version of the protocol. Note that an alternative way of fixing the problem is to assume that the channels between the TTP and both Alice and Bob are confidential.

3.2 The inductive approach

In [6], Chadha, Kanovich and Scedrov use a different approach to study the GJM protocol. They model the protocol using the multiset rewriting formalism, and use inductive methods to prove properties, by reasoning about strategies in reachable configurations. They also show that the strategies may be extracted from linear logic proofs. Using their formalism they prove that the protocol is both fair and *balanced*. Balance is a version of abuse-freeness, that discards the aspect of proving to an external party. In fact, balance is similar to the version of abuse-freeness studied by Shamtikov and Mitchell. Hence, a protocol is said to be balanced for Alice (respectively Bob), if Bob (respectively Alice) does not have both the power to complete the protocol and to abort it. We will give a formal definition of balance further in the paper and we will formally show the relation with abuse-freeness.

4 A model of games to analyze contract signing protocols

For a more complete description of the underlying formal model, and the way it is used to verify exchange protocols we refer the reader to [10]. Here we only give a short overview. The complete definitions can also be found in appendix A.

Alternating Transition Systems. The formalism that we use to model contract signing protocols as games is the model of alternating transition systems, ATS for short. The formal definition of this model was first given in [1], here we only give an intuitive introduction. ATS are a game variant of usual Kripke structures. An ATS is composed of a set of states Q that represents the possible game configurations, a finite set of propositions P, a labelling function $L : Q \to 2^P$ that labels states with propositions, a set of players Σ and a game transition function δ. The game transition function defines for every player a and state q the set of choices $\delta(q,a) = \{Q_1, Q_2, \ldots, Q_n\}$, with $Q_i \subseteq Q$, that the player a can make in q. A choice is a set of possible next states. One step of the game at a state q is played as follows : each player $a \in \Sigma$ makes his choice and the next state of the game q' is the intersection (that is required to be a singleton) of the choices made by all the players of Σ, i.e. $q' = \cap_{a\in\Sigma}\delta(q,a)$. A computation is an infinite sequence $\lambda = q_0 q_1 \ldots q_n \ldots$ of states.

Alternating-time Temporal Logic. We now introduce the alternating-time temporal logic [1], ATL for short. For a set of players $A \subseteq \Sigma$, a set of computations Λ, and a state q, consider the following game between a protagonist and an antagonist. The game starts at state q. At each step, to determine the next state, the protagonist chooses the choices controlled by the players in the set A, while the antagonist chooses the remaining choices. If the resulting infinite computation belongs to the set Λ, then the protagonist wins. If the protagonist has a winning strategy, we say that the ATL formula $\langle\langle A\rangle\rangle\Lambda$ is satisfied in state q. Here, $\langle\langle A\rangle\rangle$ is a path quantifier, parameterized by the set A of players, which ranges over all computations that the players in A can force the game into, irrespective of how the players in $\Sigma \setminus A$ proceed. The set Λ is defined using temporal logic formulas. The usual temporal operators are used: $\Diamond$ meaning *eventually*, $\Box$ meaning *always*, $\bigcirc$ meaning *next*, $\mathcal{U}$ meaning *until*. If the reader is familiar with the branching time temporal logics, he may see that the parameterized path quantifier $\langle\langle A\rangle\rangle$ is a generalization of the path quantifiers of CTL: the existential path quantifier $\exists$ corresponds to $\langle\langle\Sigma\rangle\rangle$, and the universal path quantifier $\forall$ corresponds to $\langle\langle\emptyset\rangle\rangle$. We now illustrate the expressive power of ATL. Consider the set of players $\Sigma = \{a,b,c\}$ and the following formulas with their verbal reading:

- $\langle\langle a\rangle\rangle\Diamond p$, player a has a strategy against players b and c to eventually reach a state where the proposition p is true;
- $\neg\langle\langle b,c\rangle\rangle\Box p$, the coalition of players b and c does not have a strategy against a to reach a point where the proposition p will be true for ever;
- $\langle\langle a,b\rangle\rangle\bigcirc(p \wedge \neg\langle\langle c\rangle\rangle\Box p)$, a and b can cooperate so that the next state satisfies p and from there c does not have a strategy to impose p for ever.

Those three formulas are a good illustration of the great expressive power of ATL to express cooperative as well as adversarial behaviors between players.

Fairness. [1] As in the usual temporal logic setting, fairness can be used to rule out computations. For example, to evaluate a formula of the form $\langle\langle a\rangle\rangle\Diamond\phi$, we may want to only consider computations where the antagonists, that is the agents in $\Sigma \setminus \{a\}$, respect their fairness constraints. A *fairness constraint* is a function $\gamma : Q \times \Sigma \rightarrow 2^{2^Q}$ such that for each $q \in Q$ and $a \in \Sigma$, we have $\gamma(q,a) \subseteq \delta(q,a)$. Given a computation $\lambda = q_0 q_1 \ldots q_n \ldots$, we say that γ is *a-enabled* at position $i \geq 0$ if $\gamma(q_i, a) \neq \emptyset$. We say that γ is *a-taken* at position $i \geq 0$ if there exists a set $Q' \in \gamma(q_i, a)$ such that $q_{i+1} \in Q'$. Finally given a set $A \subseteq \Sigma$, we say that λ is *weakly $\langle\gamma, A\rangle$-fair* [2] if for each agent $a \in A$, either there are infinitely many positions of λ at which γ is not a-enabled, or there are infinitely many positions of λ at which γ is a-taken. A *fairness condition* Γ is a set of fairness constraints.

[1] Unfortunately, fairness is a term used in verification and must not be confused with the property that must ensure contract signing protocols.

[2] Note that there exists also, as in the usual temporal logic setting a notion of strong fairness, the interested reader is referred to [1] for the definition. We will not need the notion of strong fairness in this paper.

```
A:
[] true → p':=true
[] true →

B:
[] p → q':=true
[] true →

C:
[] q → r':=true
[] true →
```

Figure 1: A simple program in the game guarded command language.

So, for a given set $A \subseteq \Sigma$, we say that λ is *weakly* $\langle\Gamma, A\rangle$*-fair*, if λ is weakly $\langle\gamma, A\rangle$-fair for all fairness constraints $\gamma \in \Gamma$. When ATL formulas are evaluated in states an ATS with fairness constraints, only fair computations are considered, see appendix A for details.

Game Guarded Command Language. Instead of modeling protocols directly with ATS we will use a more user-oriented notation: a guarded command language a la Dijkstra. The details about the syntax and semantics of this language (given in terms of ATS) can be found in [9]. Here follows an intuitive presentation. Each player $a \in \Sigma$ disposes of a set of guarded commands of the form:

$$\xi \equiv guard_\xi \rightarrow update_\xi \tag{1}$$

A computation-step is defined as follows: each player $a \in \Sigma$ chooses one of his commands whose guard evaluates to true, and the next state is obtained by taking the conjunction of the effects of each update part of the commands selected by the players. Fairness constraints can be expressed at the level of guarded commands: an execution is fair to the guarded command ξ of a player a iff it is not the case that *guard* evaluates constantly to true and the command is never taken.

Illustration. In the example of figure 1, we use the following syntactic assymptions. First, in the update part of the command, if a variable x does not appear in the right-hand side of the command, it is considered as unchanged by the command, so implicitely, we have $x = x'$ for each such variable. Second, the command *true* $\rightarrow$ means that the player owning this command can choose to leave unchanged its controlled variables, this is a kind of "idle" action. Note that as the guard is true, that action can be taken at any step of the game if no fairness constraints are imposed on the player.

We are now equipped to consider the example. A state of the system, is a valuation for the propositions p, q, r. Let us consider the state v such that $v_p = \mathit{false}$, $v_q = \mathit{false}$, and $v_r = \mathit{false}$. From that state, players A, B, and C can cooperate so that the system reach a state where r is true. Formally, this is expressed by the following statement:

$$v \models \langle\langle A, B, C\rangle\rangle \Diamond r \tag{2}$$

But as player C can always choose to perform the "idle" action, then A and B alone do not have a strategy to reach a state where r is true, so

$$v \models \neg\langle\langle A, B\rangle\rangle\Diamond r \tag{3}$$

In fact, player C has a spoiling strategy to prevent A and B to reach a r-state. Furthermore, he has a winning strategy to impose $\neg r$ for ever[3]:

$$v \models \langle\langle C\rangle\rangle\Box\neg r \tag{4}$$

The winning strategy simply consists in taking the idle action at every step of the game. If we want to exclude this behavior (because it is not a realistic one for example), we can use fairness constraints. By setting a weak fairness constraint on the action of C that set r to true, we exclude the strategies of C that consists in taking the idle action for ever. This fairness constraint excludes spoiling strategies of C for formula 3. So, we have :

$$v \models_F \langle\langle A, B\rangle\rangle\Diamond r \tag{5}$$

5 Modeling of the protocols

The notation, classically used in literature to describe cryptographic protocols ($A \rightarrow B : m$, to denote that Alice sends a message m to Bob), has several drawbacks. The protocols are presented as a linear sequence of message exchanges, with a predefined order. In the case of optimistic exchange protocols, like contract signing protocols, often subprotocols can be invoked at different moments, e.g. abort or recovery protocols. Unfortunately, running a subprotocol at a time not foreseen by the designer, may have unexpected side-effects, threatening the security of a protocol if one of the participating entities tries to cheat the other entities. Therefore, we use the modeling language described above to model the exchange protocols. To execute a given action ξ, $guard_\xi$ must evaluate to true. The guard $guard_\xi$ is used to represent the elements (such as keys, messages, ...) necessary for a participating entity, that is a player in our modelization, to execute the action ξ. The variables of a player represent his current knowledge and thus determine which actions he/she can execute.

Trusted Third Party. The TTP is a special player and has to be modeled in a particular way. The TTP must be *impartial*, it may not help one or the other player. To make sure that the TTP does not have a strategy to help one of the players to cheat, we model the TTP in a deterministic way: at each stage of the execution of the protocol, the TTP executes the action requested by the protocol.

Communications Channels. The communication channels can also be modeled as players. Each transmission is modeled as a guarded command. We consider three different kinds of communication channels: unreliable, resilient, and operational. No assumptions have to be made about unreliable channels: data may be lost. A resilient channel (also called asynchronous network) delivers correct data after a finite, but unknown amount of time. Data may be delayed, but will eventually arrive. When using an operational channel (also called synchronous network) correct data arrive after a known, constant amount of time. Operational channels are however rather unrealistic in heterogeneous networks. To specify *unreliable channels* we add to the guarded command

[3]The carefull reader has noted that statement $v \models \neg\langle\langle A, B\rangle\rangle\Diamond r$ is not equivalent to $\langle\langle C\rangle\rangle\Box\neg r$. We have that $\langle\langle C\rangle\rangle\Box\neg r$ implies $v \models \neg\langle\langle A, B\rangle\rangle\Diamond r$ but not the other way round.

that models the delivery of a message, an action that the channel can always take and has no effect, i.e. this is simply an idle action. As a consequence, a message sent on an unreliable channel may never be delivered. A *resilient channel* can be specified in a similar way, but we have to add fairness constraints on the computations in order to force the transmission before a finite amount of time. More precisely, assume that a resilient channel c has to deliver a message m_1 when the proposition $\mathtt{SendM_1}$ is set to true and the delivery of the message is modeled by setting $\mathtt{M_1}$ to true, modeled by the guarded command $[\!] \ \mathtt{SendM1} \rightarrow \mathtt{M1} := \mathtt{true}$. To rule out computations where a resilient channel is asked to deliver M_1 and never does it, we impose a fairness constraint on that guarded command. The fairness constraint rules out trajectories where c can deliver a message and never delivers it. *Operational channels* do not have any additional action and thus immediately transmit the messages that have been sent on it. By immediately transmitting messages we alter the channel definition, which requires messages to be transmitted before a known constant time. This is however acceptable as the sender can always wait before sending the message to obtain the desired delay.

Alice and Bob. As we explained earlier, we want to verify properties of contract signing protocols where Alice and Bob are either honest, i.e. following the protocol, or malicious. We believe that the model of an external intruder used to verify other security protocols, especially authentication protocols, is not well suited here. We want to model explicitly that Alice and Bob can construct attacks and so diverge from the honest execution of the protocol. As a consequence, we consider two theories for Alice and Bob: A_h, respectively B_h, for the honest behavior and A, respectively B, for the malicious behavior.

Each player $\mathsf{A}_h, \mathsf{A}, \mathsf{B}_h, \mathsf{B}$ is defined by a set of guarded commands. Intuitively, the descriptions of A_h and B_h contain guarded commands describing their behavior as described by the protocol. The player A, respectively B, is obtained from A_h, respectively B_h, by (i) relaxing guards and (ii) adding new actions.

We now explain more in detail how the guarded commands of A_h and B_h are constructed from an informal arrow based description of the protocols and after how this guarded commands are relaxed and extended to obtain the players A and B.

When starting to model an informally described protocol we apply the following method. First, we determine the initial knowledge of each player. Initial knowledge may for instance include nonces, public and shared keys as well as initially known messages, such as the contract text. All these items are represented by boolean variables initialized to true. All other variables are set to false, indicating that their value is not known yet. Then we model all cryptographic operations, that are useful to build the messages of the protocol. To represent the sending of a message $\mathtt{X} \rightarrow \mathtt{Y} : \mathtt{M_i} = \mathtt{m_{i1}}, \mathtt{m_{i2}}, \ldots, \mathtt{m_{in}}$ of the protocol, a guarded command $\mathtt{m_{i1}} \wedge \cdots \wedge \mathtt{m_{in}} \rightarrow \mathtt{SENDm_i} := \mathtt{true}$ is added to player $\mathtt{X}$'s description. The guard contains the conjunction of all the elements of the message. Moreover we force the order of the sent messages. The update relation expresses the intention to send the message. The transmission of the message is represented by a guarded command added to the description of the communication channel between $\mathtt{X}$ and $\mathtt{Y}$. The command will be of the form $\mathtt{SENDm_i} \rightarrow \mathtt{M_i} := \mathtt{true}$. Note that in order to delay this message a communication channel may choose to execute an idle command. Reception of the message is modeled by adding the following command to $\mathtt{Y}$'s description $\mathtt{M_i} \rightarrow \mathtt{m_{i1}} := \mathtt{true}; \ldots; \mathtt{m_{in}} := \mathtt{true}$. For each element of $\mathtt{M_i}$ being set to true for the first time in $\mathtt{Y}$'s description, we also add all useful cryptographic operations that are needed to build later messages of the protocol.

To get the dishonest variant of a player, we relax the guards of the honest description, so that no predefined order is not imposed anymore. Moreover, we add some guarded commands that enable the malicious party to read the messages sent between the honest party and the TTP and to learn their content. The only condition for sending a message is to know all elements of the message. Contract signing protocols, as well as non-repudiation and fair exchange protocols have several particularities that allow us to make some simplifications to the standard Dolev-Yao intruder. As also noted in [4] all messages are generally protected by digital signatures or an equivalent mechanism. If we suppose strong typing, we can consider that only well formed messages can be sent. Moreover each protocol execution can be uniquely identified by the identity of the participating entities and a special label. When the label contains some well chosen information related to an execution, as proposed for instance in [21], it is possible to show that different parallel executions can not interfere. A unique identification is inherent to optimistic protocols, as the TTP must be able to identify the protocol run, in case of an abort or recovery request. Therefore we do not need to verify the execution of several parallel runs. This makes the number of messages that can be constructed finite, and makes the analysis using a finite state tool possible. For each message that is received, the entity can retrieve all of its elements and apply cryptographic operations in order to get the different elements for possible messages, with respect to the protocol definition. Now, once an acceptable message can be constructed, the dishonest party can send it on the communication channels. This specification takes into account all possible executions of subprotocols corresponding to a given initial knowledge, as we do not give any predefined order to these guarded commands. At each point in the protocol execution all messages that could possibly be sent are determined. Thus, the specification enables a malicious entity to choose a different order of execution and construct a possible attack. We model the two players Alice and Bob with this approach.

As an illustration, let us consider the guarded commands shown in figure 2. The first guarded command models the encryption of message `Ma` under key `Ka` resulting in the cipher `Ca`. The second one expresses that "if Alice has knowledge of `L`, `T`, `C` and `EOO`, she can send message 1". The third guarded command models that the communication channel deliver the message 1 (`M1` becomes true). The fourth one expresses that Bob increases his knowledge when he receives the message `M1`. The two last ones show an example of how we relax the guards to differentiate the honest from the malicious behavior. While the honest Bob only sends the recovery request `SendMrb1`, after having sent the second message of the main protocol, the malicious description of Bob, enables him to send the recovery request as soon as he got all the necessary elements to construct the message, without requiring that he first sends the second message of the main protocol.

6 Properties of contract signing protocols

In this section we show how the different properties that a contract signing protocol is requested to provide can be naturally stated as strategies for the different players and easily be formulated as ATL formulas. One of the major advantages of using the notion of strategies is the possibility to directly and formally include adversarial or cooperative behaviors between the different entities taking part in a protocol run. We show, that also abuse-freeness, which is a rather complicated property finds a natural specification in our framework.

We start defining fairness. Generally, fairness is defined as follows. *"At the end of the protocol, either Alice and Bob both receive valid contracts or none of them received any valuable information"*. We split this definition in two parts to individually express fairness for Alice and Bob, and rephrase these properties in terms of strategies. We say that *"a protocol is fair for Alice, if Bob in collaboration with the communication channels does not have a strategy to receive a signed contract without Alice also having a strategy, against Bob and the communication channels, to re-*

```
Alice :
⋮
[] Ma ∧ Ka → Ca′ := true
⋮
[] La ∧ Ta ∧ Ca ∧ EOOa ∧ ¬STOPa ∧ ¬SENDm1 → SENDm1′ := true;
⋮
Communication Channel :
⋮
[] SENDm1 ∧ ¬M1 → M1′ := true;
⋮
Bob_H :
⋮
[] M1 ∧ ¬EOOb ∧ ¬STOPb → Lb′ := true; Tb′ := true; Cb′ := true; EOOb′ := true;
⋮
[] SigM1_b ∧ SendM2 ∧ ¬SendMrb1 ∧ ¬STOPb → SendMrb1′ := true;
⋮
Bob:
⋮
[] SigM1_b ∧ ¬SendMrb1 ∧ ¬STOPb → SendMrb1′ := true;
⋮
```

Figure 2: Example of guarded commands used for protocol modeling.

ceive a signed contract". This can be expressed by the following ATL formula, where Com denotes the communication channels, and $\mathsf{contract_A}$, respectively $\mathsf{contract_B}$, denotes the proof, that Alice, respectively Bob, have signed the contract:

$$\neg\langle\langle B, \mathsf{Com}\rangle\rangle\Diamond(\mathsf{contract_A} \wedge \neg\langle\langle A_h\rangle\rangle\Diamond\ \mathsf{contract_B}) \tag{6}$$

Note that here, we give to Bob all the power he can have, as we use the malicious theory for Bob and the honest one for Alice. If the formula does not hold, it can be interesting to also give Alice arbitrary behavior. If Alice, having malicious behavior has a strategy to avoid Bob breaking fairness, this strategy could possibly be used to fix the protocol. The definition of fairness for Bob, is stated in a similar way.

Informally timeliness is defined as follows. *"A protocol respects timeliness, if both Alice and Bob always have the ability to reach, in a finite amount of time, a point in the protocol where they can stop the protocol while preserving fairness."* As for fairness, we split the definition in two, timeliness for Alice and timeliness for Bob. Timeliness for Alice can easily be expressed in terms of strategies and is formulated by the following ATL formula:

$$\langle\langle A_h\rangle\rangle\Diamond\,(\mathsf{stop_A} \wedge (\neg\mathsf{contract_B} \rightarrow \neg\langle\langle B, \mathsf{Com}\rangle\rangle\Diamond\ \mathsf{contract_A})) \tag{7}$$

The formula says that a honestly behaving Alice must have a strategy to stop the protocol, and at that point if Alice did not get a signed contract, Bob is not able to get his signed contract later, even if he tries to cheat Alice. As for fairness, we use the malicious theory for Bob, while using the honest theory for Alice. The boolean variable $\mathsf{stop_A}$ is used to reflect the fact that Alice has stopped the protocol, meaning that all her actions become disabled.

Now we will discuss balance and abuse-freeness. These properties are specific to contract signing. Balance is very easy to express in terms of strategies. Informally we say that *"a protocol is balanced for Bob, if Alice does not have at any stage of the protocol both the power to complete the contract and the power to abort it"*. Note that the term *abort* is ambiguous as in optimistic protocols an abort protocol can always be run after a successful run, where the TTP did not intervene. So by having the power of aborting, we mean that the protocol must be aborted and the other party does not already have and cannot obtain a signed contract. The ATL formulation of this property is as follows:

$$\neg \left(\langle\langle A \rangle\rangle \Diamond \left(\mathsf{contract_B}\right) \wedge \langle\langle A \rangle\rangle \Diamond \left(\mathsf{aborted} \wedge \neg \langle\langle B_h \rangle\rangle \Diamond \mathsf{contract_A}\right)\right) \tag{8}$$

This is the property that has also been studied in [18, 19] and [6]. However it is possible to completely model abuse-freeness. In the original paper [8], the authors gave the following definition. *An optimistic contract signing protocol is abuse-free if it is impossible for a single player at any point in the protocol to be able to prove to an outside party that he has the power to terminate (abort) or succesfully complete the contract."* The difficult part is to model the fact that an entity can prove to an external party having a given power on the protocol outcome. Therefore we have to discuss what it means to prove having the power to abort and to complete a protocol run to an external party, that we call Charlie. Another question arising is whether a malicious Alice should have a strategy against a Bob, following honestly the protocol, or whether Alice should be able to prove to Charlie that she has a strategy against a possibly malicious Bob, showing arbitrary behavior. The definition should clearly be stated with a honest Bob: it is easier for Alice to cheat a honest Bob, and this should already be considered as a flaw in the protocol. However, for verification purposes, it can be interesting to verify whether a malicious Bob can defend himself against Alice, and if so, whether Bob's strategy can be used to fix the problem. Now, we discuss how we model the fact that a player has the ability to prove something to Charlie. If an entity has a strategy, then he can prove it to Charlie, by showing this strategy. However, this is not enough in the context of contract signing. We must also prove that a protocol run has been started between Alice and Bob. To prove this fact, we have to show a universally verifiable signature to Charlie, which contains the requested information. When Alice wants to prove to Charlie that she engaged a protocol run with Bob, she has to show a message signed by Bob, claiming that the protocol run is started. As such a signature cannot be forged by Alice, Charlie is convinced that the protocol is started. Here PCS play an essential role: Charlie cannot distinguish, whether a given PCS has been created by Bob or by Alice herself and so a PCS cannot be used to prove anything to Charlie. In our model we introduce a special proposition, $\mathsf{prove2C}$, that is used as a signal and set to true to notify that a given entity has a signature, that can be used to convince Charlie that the protocol is started between Alice and Bob. This proposition must be instantiated for each particular protocol. In the ASW protocol, Alice can set $\mathsf{prove2C}$ to true, once the second or the fourth message of the main protocol are received. Bob has to wait for the first or the third message to set this proposition to true. In the GJM, where we use PCS for the first two messages, Alice must receive the fourth message and Bob the third message before setting $\mathsf{prove2C}$ to true. So, we can model abuse-freeness for Bob by the following ATL formula:

$$\neg\langle\langle A \rangle\rangle \Diamond (\mathsf{prove2C} \wedge \langle\langle A \rangle\rangle \Diamond \mathsf{contract_B} \wedge \langle\langle A \rangle\rangle \Diamond (\mathsf{aborted} \wedge \neg \langle\langle B_h \rangle\rangle \Diamond\ \mathsf{contract_A})) \tag{9}$$

This is the first complete modeling of abuse-freeness for verification purposes. Modeling abuse-freeness and balance in ATL also allows us to formally show the relationship between both properties. Balance has been introduced as a variant of abuse-freeness, but the relationship with abuse-freeness has not clearly been stated. Looking at the logical specification of both the balance property and the abuse-freeness property, it is easily seen that balance implies abuse-freeness. To prove this statement, we consider its contraposition, i.e. the fact that a protcol is not abuse-free implies that the protocol is not balanced. Given the three following proof rules

$$\frac{\langle\langle A\rangle\rangle\Diamond\varphi,\ \varphi\to\psi}{\langle\langle A\rangle\rangle\Diamond\psi} \qquad (R_1)$$

$$\frac{\langle\langle A\rangle\rangle\Diamond(\varphi_1\wedge\varphi_2)}{\langle\langle A\rangle\rangle\Diamond\varphi_1\wedge\langle\langle A\rangle\rangle\Diamond\varphi_2} \qquad (R_2)$$

$$\frac{\langle\langle A\rangle\rangle\Diamond\langle\langle A\rangle\rangle\Diamond\varphi}{\langle\langle A\rangle\rangle\Diamond\varphi} \qquad (R_3)$$

we have that

$$\begin{array}{ll}
1.\langle\langle A\rangle\rangle\Diamond(\mathsf{prove2C}\wedge\langle\langle A\rangle\rangle\Diamond\mathsf{contract_B}\wedge\langle\langle A\rangle\rangle\Diamond(\mathsf{aborted}\wedge\neg\langle\langle B\rangle\rangle\Diamond\ \mathsf{contract_A})) & \\
2.\langle\langle A\rangle\rangle\Diamond(\langle\langle A\rangle\rangle\Diamond\mathsf{contract_B}\wedge\langle\langle A\rangle\rangle\Diamond(\mathsf{aborted}\wedge\neg\langle\langle B\rangle\rangle\Diamond\ \mathsf{contract_A})) & \text{by 1, } R_1\\
3.\langle\langle A\rangle\rangle\Diamond\langle\langle A\rangle\rangle\Diamond\mathsf{contract_B}\wedge\langle\langle A\rangle\rangle\Diamond\langle\langle A\rangle\rangle\Diamond(\mathsf{aborted}\wedge\neg\langle\langle B\rangle\rangle\Diamond\ \mathsf{contract_A}) & \text{by 2, } R_2\\
4.\langle\langle A\rangle\rangle\Diamond\mathsf{contract_B}\wedge\langle\langle A\rangle\rangle\Diamond(\mathsf{aborted}\wedge\neg\langle\langle B\rangle\rangle\Diamond\ \mathsf{contract_A}) & \text{by 3, } R_3
\end{array}$$

This also means that to prove that a protocol is abuse-free, it is sufficient to prove that the protocol is balanced. This corresponds to the intuition that when a protocol is for instance balanced for Bob, i.e. Alice does not have the power to choose between aborting or succesfully finishing the protocol, she cannot prove having this power to Charlie, as she does actually not have it.

7 Analysis of the protocols using Mocha

We used the model checker Mocha and its script language Slang to analyse both the ASW and GJM protocols. We use the script language to compute the fix points corresponding to the given formulae and verify that the formulae hold in each reachable state of the protocol.

The first property we verified is fairness. We have to slightly instantiate the formula given in the previous section to make it fit to each of the protocols. For instance for the GJM protocol we verify the following formula:

$$\neg\langle\langle A,\mathsf{Com}\rangle\rangle\Diamond(Sig_B\wedge\neg\langle\langle B_h\rangle\rangle\Diamond\ Sig_A)$$

Both the ASW, as well as the GJM protocols, i.e. the modified version proposed by Shmatikov and Mitchell, are fair for our model of a malicious Alice, respectively malicious Bob. We also verified timeliness for both protocols, and the property holds. When verifying the protocols, we suppose that the communication channels are resilient. This is a necessary condition for the protocols to provide the requested properties. When the channels are unreliable, it is easy to see that neither fairness nor timeliness hold, as the malicious party could for instance block the TTP's channel, making a recovery impossible. When the channels are supposed to be operational, the properties also hold. However these channels are unrealistic in practice and so analyzing the protocol under these assumptions is of less interest.

The major contribution of our work is the analysis of balance and abuse-freeness for the two protocols. When looking at the ATL formulae we gave in the previous section to define those properties, the careful reader may notice that we did not include the communication channels to collaborate either with Alice or Bob. Generally it is supposed that the malicious party controls the communication channels. However on resilient channels, an attacker can only introduce finite delays, not being able to remove a message or delay it for an infinite amount of time. As it seems rather strange that an attacker can delay messages without removing them, we believe that the collaboration of a dishonest participant with the communication channel models the worst-case behavior, rather than a complete control. The distinction between worst-case behavior and controlling the communication channel is of no interest when verifying fairness and timeliness, as these properties must hold in all circumstances, i.e. for any behaviour of the communication channels. When studying balance and abuse-freeness, the dishonest party must prove to Charlie that he has a strategy to either abort or successfully finish the protocol run. In order to prove this, he should have this power in all circumstances, no matter what delays are introduced on the communication channels. Otherwise, he can only prove that he may be able to chose between aborting or successfully signing the contract, depending on the behavior of the communication channels. Resilient channels are a source of non-determinism, and so two cases are possible. Either they are in coalition with the malicious party and they help him to cheat the honest party, or they are in coalition with the honest party and help this party to avoid being cheated. For fairness and timeliness, it is clear that those properties must hold, even when the communication channels react in favour of the malicious party. For balance and abuse-freeness this is not so clear. Can a dishonest party prove its power to Charlie, if this power depends on the behaviour of the communication channels ? Hence, when verifying these properties, it is interesting to consider all the possible cases: where the communication channels collaborate or not with the malicious party.

We have verified balance and abuse-freeness in both scenarios where the communication channels collaborate with either the dishonest party, or the honest party. Then we propose a new definition of abuse-freeness, and also discuss this case. We first discuss our analysis when the dishonest party completely controls the communication channels.

Dishonest party controlling the communication channels. For instance, to verify whether the GJM protocol is balanced for Bob we verify the following formula:

$$\neg(\langle\!\langle A, \mathsf{Com}\rangle\!\rangle\Diamond\ (Sig_B) \wedge \langle\!\langle A, \mathsf{Com}\rangle\!\rangle\Diamond(\mathsf{aborted} \wedge\ \langle\!\langle B_h\rangle\!\rangle\Diamond\ Sig_B))$$

Under the assumption that the communication channels are completely controlled by the dishonest party, using either operational channels or resilient channels, neither the GJM, nor the ASW protocols are balanced. This is due to the fact, that once Alice has received the second message of the main protocol she has all needed information to either abort the protocol, by sending an abort request to the TTP, or to successfully complete the protocol, by sending to Bob the third message of the main protocol, or sending a recovery request to the TTP. This result seems to be in contradiction with the result presented in [6] where Chadha et al. prove balance for the GJM protocol on operational channels. This can be explained by the fact that we have a different view related to the time-outs intervening in the protocol. When Bob sends the second message of the main protocol, he starts a timer. Only when this timer times out and Bob did not receive the third message of the main protocol before, he launches a recovery protocol. The argument given by Chadha et al. is that Bob has the possibility to launch a recovery protocol and avoid the abort. This is equivalent to asking *"does there exist a time-out for which the protocol is balanced"*. Our approach differs on this point as we ask whether there exists a time-out for which the protocol is *not* balanced.

When verifying abuse-freeness we find that the ASW protocol is not abuse-free, while the property holds on the GJM protocol. Abuse-freeness does not hold on the ASW protocol as the protocol is not balanced and Alice can prove to an external party that the protocol has been started, once she received Bob's first message. Here we see why PCS are useful in the GJM protocol. In the GJM protocol, Alice does not get an evidence, that the protocol has been started with Bob, before having either aborted or successfully finished the protocol. These results reflect, what was widely believed and claimed in the paper, where the GJM protocol was presented, namely that PCS are needed to achieve abuse-free contract signing.

Honest party controlling the communication channels. Now we are changing the assumptions, so that the dishonest party does not collaborate with the communication channels any more, but has to prove to Charlie, that he has the power to abort or successfully sign a contract with Bob, under all circumstances, independent of the way the communication channels react. We believe that these assumptions better reflect reality. Now, to verify whether, for instance, the GJM protocol is balanced for Bob or not we verify the following formula:

$$\neg(\langle\!\langle A\rangle\!\rangle\Diamond\ (Sig_B) \wedge \langle\!\langle A\rangle\!\rangle\Diamond(\mathsf{aborted} \wedge\ \langle\!\langle B_h, \mathsf{Com}\rangle\!\rangle\Diamond\ Sig_A))$$

Under these assumptions both protocols are balanced. Now, if Alice gets the second message of the main protocol, she does not have a strategy anymore to abort the protocol. Consider the following scenario, where Alice tries to abort the protocol. Alice sends an abort request to the TTP, but the request is delayed. In between, Bob, not having received the third message of the main protocol, times out and sends a recovery request to the TTP. Bob's recovery request may arrive at the TTP before Alice's abort request, and hence Alice's attempt to abort the protocol fails. Note, that fairness is not broken by this scenario, as in case of a recovery both Alice and Bob, get signed contracts and in case of an abort none of them gets a signed contract. So we see that the protocols are balanced, and hence are also abuse-free. This means that under these assumptions, no PCS are needed to achieve abuse-freeness, as it is defined above.

A novel definition of abuse-freeness. Although the ASW protocol may be abuse-free with the weaker definition, the fact that Alice can prove to Charlie that a contract signing protocol has been started with Bob, can be considered as a problem. Alice may not have a strategy to choose between aborting or signing, but Bob does not control the communication channels neither and so Alice's success to choose between aborting and signing depends on the way the communication channels react. We think that this should be considered as a problem. Therefore we propose a new definition of abuse-freeness, which is slightly stronger, but deals with the same initial problem. We redefine abuse-freeness for Bob by the following ATL formula.

$$\neg\langle\!\langle A\rangle\!\rangle\Diamond(\mathsf{prove2C} \wedge \neg\langle\!\langle B_h\rangle\!\rangle\Diamond\ \mathsf{contract_A})$$

The new definition requires that Alice cannot reach a state in the protocol, where she can prove to Charlie that the protocol has been initiated with Bob, without Bob having a strategy to successfully complete the protocol. Abuse-freeness for Alice is defined in the same way. With the following additional proof rule

$$\frac{\langle\!\langle A\rangle\!\rangle\Diamond(p\wedge\langle\!\langle A\rangle\!\rangle\Diamond\varphi),\ p\rightarrow\forall\Box p}{\langle\!\langle A\rangle\!\rangle\Diamond(p\wedge\varphi)}\quad (R_4)$$

we can formally show that the new definition implies the former one. Again, we prove this statement by proving its contraposition.

$$
\begin{array}{ll}
1.\langle\!\langle A\rangle\!\rangle\Diamond(\mathsf{prove2C} \wedge \langle\!\langle A\rangle\!\rangle\Diamond\mathsf{contract_B} \wedge \langle\!\langle A\rangle\!\rangle\Diamond(\mathsf{aborted} \wedge \neg\langle\!\langle B\rangle\!\rangle\Diamond\ \mathsf{contract_A})) & \\
2.\langle\!\langle A\rangle\!\rangle\Diamond(\mathsf{prove2C} \wedge \langle\!\langle A\rangle\!\rangle\Diamond(\mathsf{aborted} \wedge \neg\langle\!\langle B\rangle\!\rangle\Diamond\ \mathsf{contract_A})) & \text{by } 1,\ R_1 \\
3.\langle\!\langle A\rangle\!\rangle\Diamond(\mathsf{prove2C} \wedge \langle\!\langle A\rangle\!\rangle\Diamond\neg\langle\!\langle B\rangle\!\rangle\Diamond\ \mathsf{contract_A}) & \text{by } 2,\ R_1
\end{array}
$$

By the fact that prove2C is monotonic, i.e. once prove2C is true it remains true, we have that

$$4.\langle\!\langle A\rangle\!\rangle\Diamond(\mathsf{prove2C} \wedge \neg\langle\!\langle B\rangle\!\rangle\Diamond\ \mathsf{contract_A}) \quad \text{by}3,\ R_4$$

With this new definition, we have that the GJM protocol is still abuse-free, but the ASW protocol is not abuse-free anymore, no matter whether the communication channels collaborate with Alice or Bob. Abuse-freeness for the GJM protocol is assured by the properties of PCS. Bob receives his first proof that the protocol has been started with Alice, only after the third message has been sent. So Bob receives this proof at the same moment he gets his signed contract. Alice needs to wait for the fourth message of the protocol to prove to Charlie that the protocol is engaged with Bob. This also correponds to the state of the protocol where she gets her signed contract. Hence, both Alice and Bob, receive their proofs when they get their signed contract.

Considering the new definition, we can show that the ASW protocol is not abuse-free for Alice by looking at the following scenario. Alice sends the first message to Bob. As Alice has not yet received the second message of the protocol, she can only launch an abort protocol or wait for the second message to arrive, and has no strategy to successfully complete the protocol. So, if Bob does not send the second message, he reaches a state where he can prove to Charlie that the protocol has been started, as he received the first message signed by Alice, without Alice having a strategy to successfully complete the protocol. This scenario is independent of the reaction of the communication channels and violates our new definition of abuse-freeness. As in the ASW protocol, conventional signatures that can be shown as proofs to Charlie, are used from the beginning, when it is still possible to abort the protocol, this protocol should not be considered as abuse-free. However, considering the former definition of abuse-freeness, under certain assumptions the protocol respects abuse-freeness. Intuitively, this is due to the fact that the power to both complete and abort the protocol depends on the reaction of the communication channels. In our new definition however the result of the verification of the ASW protocol, as well as the GJM protocol is independent of whether the communication channels collaborate with the honest party or the malicious party. Therefore we believe that the above given definition is more concise and better reflects the initial intuition of the property than the classical one.

8 Conclusion

In this paper we have shown how the model of alternating-time systems, alternating-time temporal logic and their guarded command language can be used to analyse contract signing protocols. These protocols are quite particular as, on one hand they are branching, and on the other hand imply potentially malicious protocol entities. Using ATL, we have defined in a natural way the different properties a contract signing protocol has to respect. In particular we have modeled two properties that have been considered as rather tricky to specify, which are balance and abuse-freeness. We have considered two variants of the original definitions of balance and abuse-freeness and we have proposed a new slightly stronger definition of abuse-freeness. This is the first complete specification of abuse-freeness for verification purposes. Moreover, the use of ATL for specifying the properties formally shows the link between balance and abuse-freeness. We prove that balance implies abuse-freeness. Using the tool MOCHA and the script language SLANG, we have automated the analysis of two contract signing protocols, the ASW and the GJM protocol. The main work focused on the verification of balance and abuse-freeness. The analysis shows that when the communication channels are not entirely controlled by the malicious entity, which we believe is rather realistic on resilient communication channels, the ASW protocol is abuse-free with respect to the original definition. It was believed that all abuse-free protocols had to be based on private contract signatures. However, as the ASW protocol is not using PCS, a problem remains due to the fact that the dishonest party can prove to an external party that the protocol has been started, although the outcome of the protocol has not yet been determined. Therefore we give a new, stronger definition, that avoids this problem: once a party can prove to Charlie that the protocol has been engaged, the other party must have a strategy to successfully complete the protocol. We think that the use of formal methods, and in particular ATS, ATL and MOCHA, allowed us to better understand balance and abuse-freeness, two quite complicated and subtle properties of contract signing protocols. So, formal methods are not only useful here to verify automatically the protocols but also to better understand their requirements.

In future works, we are going to look at the strategy synthesis problem. Strategy synthesis would enable us to automatically generate attack scenarios. Moreover we are interested in a more formal justification of the completeness of the model we use for malicious entities in the context of optimistic exchange protocols.

References

[1] R. Alur, T. Henzinger, and O. Kupferman. Alternating-time temporal logic. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science*, pages 100–109. IEEE Computer Society Press, 1997.

[2] N. Asokan, M. Schunter, and M. Waidner. Optimistic protocols for fair exchange. In T. Matsumoto, editor, *4th ACM Conference on Computer and Communications Security*, pages 6, 8–17, Zurich, Switzerland, Apr. 1997. ACM Press.

[3] N. Asokan, V. Shoup, and M. Waidner. Asynchronous protocols for optimistic fair exchange. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, pages 86–99, Oakland, CA, May 1998. IEEE Computer Society, Technical Committee on Security and Privacy, IEEE Computer Society Press.

[4] C. Boyd and P. Kearney. Exploring fair exchange protocols using specification animation. In *The Third International Workshop on Information Security - ISW2000*, Lecture Notes in Computer Science, Australia, Dec. 2000. Springer-Verlag.

[5] M. Burrows, M. Abadi, and R. Needham. A logic of authentication, from proceedings of the royal society, volume 426, number 1871, 1989. In *William Stallings, Practical Cryptography for Data Internetworks, IEEE Computer Society Press, 1996*. 1996.

[6] R. Chadha, M. Kanovich, and A. Scedrov. Inductive methods and contract-signing protocols. In P. Samarati, editor, *8-th ACM Conference on Computer and Communications Security*. ACM Press, Nov. 2001.

[7] S. Even and Y. Yacobi. Relations among public key signature systems. Technical Report 175, Technion, Haifa, Israel, Mar. 1980.

[8] J. A. Garay, M. Jakobsson, and P. MacKenzie. Abuse-free optimistic contract signing. In *Advances in Cryptology: Proceedings of Crypto'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 449–466. Springer-Verlag, 1999.

[9] T. Henzinger, R. Manjumdar, F. Mang, and J.-F. Raskin. Abstract interpretation of game properties. In *SAS 2000: Intertional Symposium on Static Analysis*, Lecture Notes in Computer Science. Springer-Verlag, 2000.

[10] S. Kremer and J.-F. Raskin. A game-based verification of non-repudiation and fair exchange protocols. In K. Larsen and M. Nielsen, editors, *Concurrency Theory - Concur 2001*, volume 2154 of *Lecture Notes in Computer Science*, Aalborg, Denmark, Aug. 2001. Springer-Verlag.

[11] G. Lowe. An attack on the Needham-Schroeder public-key authentication protocol. *Information Processing Letters*, 56(3):131–133, Nov. 1995.

[12] G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. *Lecture Notes in Computer Science*, 1055:147–157, 1996.

[13] C. A. Meadows. Analyzing the Needham-Schroeder public-key protocol: A comparison of two approaches. *Lecture Notes in Computer Science*, 1146:351–365, 1996.

[14] S. Micali. Certified E-mail with invisible post offices. Available from author; an invited presentation at the RSA '97 conference, 1997.

[15] L. C. Paulson. Proving properties of security protocols by induction. In *Proceedings of the 10th Computer Security Foundations Workshop*, pages 70–83. IEEE Computer Society Press, June 1997.

[16] S. Schneider. Formal analysis of a non-repudiation protocol. In *Proceedings of the 11th IEEE Computer Security Foundations Workshop (CSFW '98)*, pages 54–65, Washington - Brussels - Tokyo, June 1998. IEEE.

[17] V. Shmatikov and J. Mitchell. Analysis of a fair exchange protocol. In *Symposium on Network and Distributed Systems Security (NDSS '00)*, pages 119–128, San Diego, CA, Feb. 2000. Internet Society.

[18] V. Shmatikov and J. Mitchell. Analysis of abuse-free contract signing. In *Financial Cryptography '00*, Anguilla, 2000.

[19] V. Shmatikov and J. Mitchell. Finite-state analysis of two contract signing protocols. *Special issue of Theoretical Computer Science on security*, 2001.

[20] J. Zhou and D. Gollmann. Towards verification of non-repudiation protocols. In *Proceedings of 1998 International Refinement Workshop and Formal Methods Pacific*, pages 370–380, Canberra, Australia, Sept. 1998. Springer.

[21] Y. Zhou and D. Gollmann. An efficient non-repudiation protocol. In *PCSFW: Proceedings of The 10th Computer Security Foundations Workshop*. IEEE Computer Society Press, 1997.

A Formal Definitions

Alternating transition systems. The formalism that we use to model exchange protocols as games is the model of alternating transition systems [1], ATS for short. ATS are a game variant of usual Kripke structures. An ATS is a 5-tuple $S = \langle \Pi, \Sigma, Q, \pi, \delta \rangle$ with the following components:

- Π is a set of propositions.
- Σ is a set of players.
- Q is a set of states.
- $\pi : Q \to 2^{\Pi}$ maps each state to the set of propositions that are true in the state.
- $\delta : Q \times \Sigma \to 2^{2^Q}$ is a transition function that maps a state and a player to a nonempty set of choices, where each choice is a set of possible next states. Whenever the system is in state q, each player a chooses a set $Q_a \in \delta(q, a)$. In this way, a player a ensures that the next state of the system will be in its choice Q_a. However, which state in Q_a will be next depends on the choices made by the other players, because the successor of q must lie in the intersection $\bigcap_{a \in \Sigma} Q_a$ of the choices made by all players. The transition function is non-blocking and the players together choose a unique next state. Thus, we require that this intersection always contains a unique state: assuming $\Sigma = \{a_1, \ldots, a_n\}$, then for every state $q \in Q$ and every set $Q_1, \ldots, Q_n$ of choices $Q_i \in \delta(q, a_i)$, the intersection $Q_1 \cap \ldots \cap Q_n$ is a singleton.

For two states q and q' and a player a, we say that q' is an *a-successor* of q if there exists a set $Q' \in \delta(q, a)$ such that $q' \in Q'$. We denote by $succ(q, a)$ the set of a-successors of q. For two states q and q', we say that q' is a *successor* of q if for all players $a \in \Sigma$, we have $q' \in succ(q, a)$. Thus, q' is a successor of q iff whenever the system S is in state q, the players in Σ can cooperate so that q' will be the next state. A *computation* of S is an infinite sequence $\lambda = q_0, q_1, q_2, \ldots$ of states such that for all positions $i \geq 0$, the state q_{i+1} is a successor of the state q_i. We refer to a computation starting at state q as a *q-computation*. For a computation λ and a position $i \geq 0$, we use $\lambda[i]$, $\lambda[0, i]$, and $\lambda[i, \infty]$ to denote the i-th state in λ, the finite prefix $q_0, q_1, \ldots, q_i$ of λ, and the infinite suffix $q_i, q_{i+1}, \ldots$ of λ, respectively.

Alternating-time temporal logic. The *Alternating-time Temporal Logic* [1] (ATL for short) is defined with respect to a finite set Π of *propositions* and a finite set Σ of *players*. An ATL formula is one of the following:

- p, for propositions $p \in \Pi$.
- $\neg\varphi$ or $\varphi_1 \vee \varphi_2$, where φ, φ_1, and φ_2 are ATL formulas.
- $\langle\langle A \rangle\rangle \bigcirc \varphi$, $\langle\langle A \rangle\rangle \Box \varphi$, or $\langle\langle A \rangle\rangle \varphi_1 \mathcal{U} \varphi_2$, where $A \subseteq \Sigma$ is a set of players, and φ, φ_1, and φ_2 are ATL formulas.

The operator $\langle\langle\ \rangle\rangle$ is a *path quantifier*, and $\bigcirc$ ("next"), $\Box$ ("always"), and $\mathcal{U}$ ("until") are *temporal operators*. The logic ATL is similar to the branching-time temporal logic CTL, only that path quantifiers are parameterized by sets of players. In fact, the parameterized path quantifier $\langle\langle A \rangle\rangle$ is a generalization of the path quantifiers of CTL: the existential path quantifier $\exists$ corresponds to $\langle\langle \Sigma \rangle\rangle$, and the universal path quantifier $\forall$ corresponds to $\langle\langle \emptyset \rangle\rangle$. Sometimes we write $\langle\langle a_1, \ldots, a_n \rangle\rangle$ instead of $\langle\langle \{a_1, \ldots, a_n\} \rangle\rangle$. Additional boolean connectives are defined from $\neg$ and $\vee$ in the usual manner.

As in CTL, we write $\langle\langle A\rangle\rangle\Diamond\varphi$ for $\langle\langle A\rangle\rangle \mathit{true}\,\mathcal{U}\varphi$. Extensions of ATL as ATL^* and the alternating μ-calculus are defined in [1].

We interpret ATL formulas over the states of a given ATS S that has the same propositions and players. The labeling of the states of S with propositions is used to evaluate the atomic formulas of ATL. The logical connectives $\neg$ and $\vee$ have the standard interpretation. To evaluate a formula of the form $\langle\langle A\rangle\rangle\psi$ at a state q of S, consider the following two-player game between a protagonist and an antagonist. The game proceeds in an infinite sequence of rounds, and after each round, the position of the game is a state of S. The initial position is q. Now consider the game in some position u. To update the position, first the protagonist chooses for every player $a \in A$, a set $Q_a \in \delta(u, a)$. Then, the antagonist chooses a successor v of u such that $v \in Q_a$ for all $a \in A$, and the position of the game is updated to v. In this way, the game continues forever and produces a computation. The protagonist wins the game if the resulting computation satisfies the subformula ψ, read as a linear temporal formula whose outermost operator is $\bigcirc$, $\Box$, or $\mathcal{U}$. The ATL formula $\langle\langle A\rangle\rangle\psi$ holds at the state q if the protagonist has a winning strategy in this game.

In order to define the semantics of ATL formally, we first define the notion of strategies. Consider an ATS $S = \langle \Pi, \Sigma, Q, \pi, \delta\rangle$. A *strategy* for a player $a \in \Sigma$ is a mapping $f_a : Q^+ \to 2^Q$ such that for all $\lambda \in Q^*$ and all $q \in Q$, we have $f_a(\lambda \cdot q) \in \delta(q, a)$. Thus, the strategy f_a maps each finite prefix $\lambda \cdot q$ of a computation to a set in $\delta(q, a)$. This set contains possible extensions of the computation as suggested to player a by the strategy. Each strategy f_a induces a set of computations that player a can enforce. Given a state q, a set A of players, and a set $F_A = \{f_a \mid a \in A\}$ of strategies, one for each player in A, we define the *outcomes* of F_A from q to be the set $\mathit{out}(q, F_A)$ of all q-computations that the players in A can enforce when they cooperate and follow the strategies in F_A; that is, a computation $\lambda = q_0, q_1, q_2, \ldots$ is in $\mathit{out}(q, F_A)$ if $q_0 = q$ and for all positions $i \geq 0$, the state q_{i+1} is a successor of q_i satisfying $q_{i+1} \in \bigcap_{a\in A} f_a(\lambda[0, i])$.

We can now turn to a formal definition of the semantics of ATL. We write $S, q \models \varphi$ ("state q satisfies formula φ in the structure S") to indicate that the formula φ holds at state q of S. When S is clear from the context we omit it and write $q \models \varphi$. The relation $\models$ is defined, for all states q of S, inductively as follows:

- For $p \in \Pi$, we have $q \models p$ iff $p \in \pi(q)$.
- $q \models \neg\varphi$ iff $q \not\models \varphi$.
- $q \models \varphi_1 \vee \varphi_2$ iff $q \models \varphi_1$ or $q \models \varphi_2$.
- $q \models \langle\langle A\rangle\rangle\bigcirc\varphi$ iff there exists a set F_A of strategies, one for each player in A, such that for all computations $\lambda \in \mathit{out}(q, F_A)$, we have $\lambda[1] \models \varphi$.
- $q \models \langle\langle A\rangle\rangle\Box\varphi$ iff there exists a set F_A of strategies, one for each player in A, such that for all computations $\lambda \in \mathit{out}(q, F_A)$ and all positions $i \geq 0$, we have $\lambda[i] \models \varphi$.
- $q \models \langle\langle A\rangle\rangle\varphi_1\,\mathcal{U}\varphi_2$ iff there exists a set F_A of strategies, one for each player in A, such that for all computations $\lambda \in \mathit{out}(q, F_A)$ there exists a position $i \geq 0$ such that $\lambda[i] \models \varphi_2$ and for all positions $0 \leq j < i$, we have $\lambda[j] \models \varphi_1$.

Note that the next operator $\bigcirc$ is local: $q \models \langle\langle A\rangle\rangle\bigcirc\varphi$ iff for each player $a \in A$ there exists a set $Q_a \in \delta(q, a)$ such that for each state $q' \in \bigcap_{a\in A} Q_a$, if q' is a successor of q, then $q' \models \varphi$.

Fairness. As in the usual temporal logic setting, fairness can be used to rule out computations. For example, to evaluate a formula of the form $\langle\langle a \rangle\rangle \Diamond \phi$, we may want to only consider computations where the antagonists, that is the agents in $\Sigma \setminus \{a\}$, respect their fairness constraints. A *fairness constraint* is a function $\gamma : Q \times \Sigma \to 2^{2^Q}$ such that for each $q \in Q$ and $a \in \Sigma$, we have $\gamma(q, a) \subseteq \delta(q, a)$. Given a computation $\lambda = q_0 q_1 \ldots q_n \ldots$, we say that γ is *a-enabled* at position $i \geq 0$ if $\gamma(q_i, a) \neq \emptyset$. We say that γ is *a-taken* at position $i \geq 0$ if there exists a set $Q' \in \gamma(q_i, a)$ such that $q_{i+1} \in Q'$. Finally given a set $A \subseteq \Sigma$, we say that λ is *weakly* $\langle \gamma, A \rangle$*-fair* [4] if for each agent $a \in A$, either there are infinitely many positions of λ at which γ is not a-enabled, or there are infinitely many positions of λ at which γ is a-taken. A *fairness condition* Γ is a set of fairness constraints. So, for a given set $A \subseteq \Sigma$, we say that λ is *weakly* $\langle \Gamma, A \rangle$*-fair*, if λ is weakly $\langle \gamma, A \rangle$-fair for all fairness constraints $\gamma \in \Gamma$.

Given an ATS S, and a (weak) fairness condition Γ, we define the semantics of Fair ATL as follows:

- $q \models_F \langle\langle A \rangle\rangle \bigcirc \phi$ iff there exists a set F_A of strategies, one for each agent in A, such that for all $\langle \Gamma, \Sigma \setminus A \rangle$-fair computations $\lambda \in out(q, F_A)$, we have $\lambda[1] \models_F \phi$.
- $q \models_F \langle\langle A \rangle\rangle \Box \varphi$ iff there exists a set F_A of strategies, one for each player in A, such that for all $\langle \Gamma, \Sigma \setminus A \rangle$-fair computations $\lambda \in out(q, F_A)$ and all positions $i \geq 0$, we have $\lambda[i] \models_F \varphi$.
- $q \models_F \langle\langle A \rangle\rangle \varphi_1 \mathcal{U} \varphi_2$ iff there exists a set F_A of strategies, one for each player in A, such that for all $\langle \Gamma, \Sigma \setminus A \rangle$-fair computations $\lambda \in out(q, F_A)$ there exists a position $i \geq 0$ such that $\lambda[i] \models_F \varphi_2$ and for all positions $0 \leq j < i$, we have $\lambda[j] \models_F \varphi_1$.
- the semantics of the other operators is as in standard ATL.

In [1], a model-checking algorithm handling fairness constraints is presented.

Game guarded command language. Modeling a protocol directly with an alternating transition system would be cumbersome. Instead we use a simple modeling formalism based on Dijkstra's guarded command language, previously introduced in [9]. This simple modeling language is a subset of the reactive modules used by the verification tool Mocha. Let X be a set of variables interpreted over some possibly infinite domains. We denote by $X' = \{x' \mid x \in X\}$ the set of variables obtained by priming each variable in X. Those variables are used to refer to the value of the variables in the next state. A valuation v of the variables X is a function which maps each of the variables in X into a value in its domain. Denote by $\mathcal{V}_X$ the set of all possible valuations of the variables X. For any valuation v, we write v' the valuation obtained by priming each domain variable of v. For any predicate φ over the variables X and a valuation $v \in \mathcal{V}_X$, denote by $\varphi[\![v]\!]$ the truth value of the predicate with all the free variables of the program P interpreted according to the valuation v. For $Y \subseteq X$, we define $v_{|Y}$ as the valuation with $dom(v_{|Y}) = Y$, such that $\forall y \in Y : v_{|Y}(y) = v(y)$.

Given a set of program variables X, a Y*-action* ξ (guarded command) has the form $[\!]$ *guard* $\to$ *update*, where $Y \subseteq X$, the guard *guard* is a boolean predicate over X, and the update relation *update* is a boolean predicate over $X \cup Y'$. We also write $guard_\xi$ and $update_\xi$ to represent the guard and update relation of ξ respectively. Given a valuation $v \in \mathcal{V}_X$, the action ξ is said to be *enabled* if the guard $guard_\xi[\![v]\!]$ evaluates to true. We assume that for every Y-action, the update relation is functional, that is, for all valuation functions $v \in \mathcal{V}_X$, there is exactly one valuation function $u \in \mathcal{V}_Y$ such that $update_\xi[\![v \cup u']\!]$ holds. A Y*-process* Φ is a finite set of Y-actions. We say that the

[4] Note that there exists also, as in the usual temporal logic setting a notion of strong fairness, the interested reader is referred to [1] for the definition. We will not need the notion of strong fairness in this paper.

set of variables Y is *the controlled variables* of Φ. We require that for any valuation of the program variables, at least one action in the process is enabled.

To model multi-player games, we partition the program variables X into disjoint sets $X_1, X_2, \ldots, X_n$, i.e., $X = X_1 \uplus X_2 \uplus \cdots \uplus X_n$. Intuitively, X_i contains the variables updated by player i. A *program* $P = (\Phi_1, \Phi_2, \ldots, \Phi_n)$ over X is a tuple of processes such that Φ_i is an X_i-process. Each process of the program can be defined by composing smaller processes, each of which controls a subset of the variables.

The *semantics* of a program $P = \{\Phi_1, \Phi_2, \ldots \Phi_n\}$ over the program variables X is the alternating transition system S_P

- $\Sigma = \{1, 2, \ldots, n\}$ (the n players of the program P).
- $Q = \mathcal{V}_X$, i.e., a state of S_P is a valuation of the program variables.
- $\delta : \mathcal{V}_X \times \Sigma \to 2^{2^{\mathcal{V}_X}}$ is the transition function defined as
  $\delta(v, i) = \big\{\{r \mid r_{|X_i} = w \wedge r \in \mathcal{V}_X\} \mid \xi \in \Phi_i \wedge \mathit{guard}_\xi[\![v]\!] \wedge \mathit{update}_\xi[\![v \cup w']\!]\big\}$
- Π is a set of boolean predicates over the program variables X.
- π maps a state q, which is a valuation, to the set $\pi(q)$ of propositions that the valuation satisfies.

Intuitively, for $i \in \Sigma = \{1, 2, \ldots, n\}$, player i controls the actions in Φ_i.

A program is run in steps. At each step, player 1 chooses an enabled action $\xi \in \Phi_1$, and updates the values of the variables in X_1 according to the predicate update_ξ. Independently and simultaneously, player 2 chooses an enabled action $\eta \in \Phi_2$ and updates the values of the variables in X_2 according to update_η. And so on for the $n-2$ other players.

We introduce fairness in our modeling language at the level of actions. Given a set of actions Φ_i of player i, a fairness constraint for player i is a subset $\Gamma_i \subseteq \Phi_i$ of actions in regard of which, player i must be (weakly) fair. Intuitively, an action that belongs to Γ_i can not be constantly enabled without being taken by player i. So, a fairness condition for a program $P = \{\Phi_1, \Phi_2, \ldots, \Phi_n\}$ is a set $\{\Gamma_1, \Gamma_2, \ldots, \Gamma_n\}$ where $\Gamma_i \subseteq \Phi_i$ for $i : 1 \leq i \leq n$. This set of subsets of actions of the program P induces the fairness condition Γ on the ATS defining the semantics of P whose transition relation is δ. Γ contains a fairness constraint γ_ξ, for each action $\xi \in \Gamma_i$, defined as follows:

$$\gamma_\xi(v, j) = \begin{cases} \emptyset & \text{if } i \neq j \\ \emptyset & \text{if } i = j \text{ and } \neg \mathit{guard}_\xi[\![v]\!] \\ \{Q' \mid Q' \in \delta(v, j) \wedge \forall u' \in Q' : \mathit{update}_\xi[\![v \cup u']\!]\} & \text{if } i = j \text{ and } \mathit{guard}_\xi[\![v]\!]. \end{cases}$$