Strategy Properties for Cryptographic Protocols

Dissertation

zur Erlangung des akademischen Grades
Doktor der Naturwissenschaften
(Dr. rer. nat.)
der Technischen Fakultät
der Christian-Albrechts-Universität zu Kiel

Detlef Kähler

Kiel
2008

1. Gutachter: Prof. Dr. Thomas Wilke

2. Gutachter: Prof. Dr. Ralf Küsters

3. Gutachter: Dr. Steve Kremer

Datum der mündlichen Prüfung: 17. Juli 2008

Contents

1 Introduction **7**

2 Preliminaries **17**

2.1 Terms and Messages 17

2.2 ASW Protocol 19

3 Algorithm for Strategy Properties **23**

3.1 The Protocol and Intruder Model 24

3.1.1 Terms and Messages 25

3.1.2 Intruder 26

3.1.3 Principals and Protocols 27

3.1.4 The Transition Graph of a Protocol 28

3.2 Modeling the Originator of the ASW Protocol 30

3.2.1 The Principal O 30

3.3 Strategy Properties and Strategy Problems 32

3.3.1 Intruder Strategies and Strategy Trees 33

3.3.2 Strategy Problems 35

3.4 Constraint Solving 36

3.5 The Constraint-Based Algorithm 39

3.5.1 Guessing the Symbolic Branching Structure 41

3.5.2 Constructing and Solving the Induced Constraint System 44

3.5.3 Checking the Induced Substitutions 49

3.6 Proof of the Main Theorem 51

3.6.1 Further Terminology 51

3.6.2 Soundness of SolveStrategy 53

3.6.3 Completeness of SolveStrategy 60

4 AMC for Cryptographic Protocols **71**
4.1 AMC and Parity Games 73
4.1.1 Concurrent Game Structures 73
4.1.2 AMC 74
4.1.3 Parity Games and AMC-Model Checking 76
4.2 Our Protocol and Intruder Model 80
4.2.1 Terms and Messages 81
4.2.2 Derivation of Messages 81
4.2.3 Channels, Principals, and Protocols 82
4.2.4 Example: The ASW Protocol 85
4.2.5 The Concurrent Game Structure Induced by a Protocol 85
4.3 Main Results 91
4.4 Example Properties 94
4.5 Proof of Theorem 4.3.1 97
4.6 Proof of Theorem 4.3.2 101
4.7 Proof of Theorem 4.3.3 105
4.7.1 Poof of Theorem 4.3.3 Using Key Lemmas 106
4.7.2 Properties of Concurrent Game Structures for Protocols 108
4.7.3 Some Properties of Parity Games for Protocols 110
4.7.4 Proof of Lemma 4.7.2 112
4.7.5 Proof of Lemma 4.7.3 115
4.7.6 Proof of Lemma 4.7.4 119
4.7.7 Lower Bound 130

5 Impossibility of Balance **135**
5.1 ATL with Move Selectors 135
5.1.1 Concurrent Game Structures 136
5.1.2 ATL with Move Selectors 138
5.2 Impossibility of Balance 141
5.2.1 Advantage and Balance 142
5.2.2 Security Assumptions 143
5.2.3 Additional Axioms 144
5.2.4 Main Result 146
5.3 Impossibility Result of Chadha et al. 148

6 Conclusion **159**

Chapter 1

Introduction

In the last fifteen years the Internet has grown dramatically—on the one hand in terms of the number of users and on the other hand in terms of the number of available services. Today it is common to do online banking, to purchase goods or to take part in auctions online. With the increasing use of online services it has gotten more and more important to ensure the security of these services. The users of such services want to be sure that the service they are using is *secure*. But what does it mean for a service to be secure? What are the possible attack scenarios these services have to withstand? How can users be protected against these attacks? All these questions naturally arise in the context of security of services. To ensure security of services *cryptographic protocols* are used for the communication between the parties involved, e.g., the online bank and the customer.

Cryptographic protocols are communication protocols which are designed to provide security assurances of various kinds, using cryptographic mechanisms. According to [MvV97] a cryptographic protocol is “a distributed algorithm defined by a sequence of steps precisely specifying the actions required of two or more entities to achieve a specific security objective”.

There are cryptographic protocols for various kinds of tasks—from very fundamental ones such as key exchange protocols to more complex ones such as contract signing protocols. As the complexity of the protocols increases the possible attacks and security objectives are getting more involved and subtle. To give a flavor of what “security objectives” may look like we explain two examples: the *key exchange problem* and the *contract signing problem*.

The key exchange problem for two parties can be described as follows:

There are two participants, A (= Alice) and B (= Bob), which want to exchange a secret over an insecure channel such that at the end of the communication A and B share a secret and the secret is only known to A and B.

The contract signing problem for two parties is the following: two participants (called signers) A and B have agreed upon a contractual text and they want to exchange their signatures, i.e., their commitments on the contractual text.

Of course, for both problems mentioned above the solutions are more or less trivial if Alice and Bob are sitting together face to face: For the key exchange problem they may agree upon a secret directly and for the contract signing problem the usual procedure where Alice and Bob simultaneously sign a copy of the contract and than exchange these copies works out.

How can these real world solutions of the key exchange problem and the contract signing problem be transfered to the digital world in which the communication between Alice and Bob is carried out over some communication channel that is not secure in some sense, i.e., what are secure key exchange and secure contract signing protocols?

If, for example, we try to model the real world procedure of signing contracts as a protocol in a naïve way one could describe this contract signing protocol as follows: Both signers have copies of the contractual text and both send a signed copy to the other signer. But, of course, there is an asymmetry in this simple protocol. If Alice signs her copy of the contract and sends it to Bob before Bob sends his signed copy to Alice it may happen that Bob receives a valid contract but Alice is not able to enforce getting a signed contract as well. This is not desirable, it is an *unfair* situation for Alice. Thus, a first security objective for a contract signing protocol is *fairness*. One way of defining fairness is: A contract signing protocol is fair if whenever the execution of the protocol is completed either both signers have a valid contract or neither does [ASW98].

Due to the distributed nature of the execution of cryptographic protocols, constructing and analyzing them is highly error-prone. One of the most prominent examples is the key exchange protocol of Needham and Schroeder [NS78]. The protocol was believed to be secure until after 17 years Lowe found an attack on this protocol with an automatic tool [Low95]. This and other examples show that a rigorous analysis of cryptographic protocols is indispensable. To start an analysis of a cryptographic protocol one first has to decide how protocols and their execution (including the capabilities of the intruder) are modeled and how security properties are specified. Then, of course, many questions arise. Among the most important ones are the following:

Decidability When formulating the question if a given cryptographic protocol is secure as a decision problem, we ask: Is this problem decidable or undecidable?

Algorithms For decidable decision problems we may ask: Is there a practical algorithmic solution to this problem?

Impossibility Are there security objectives that cannot be achieved by cryptographic protocols?

When modeling cryptographic protocols one typically distinguishes between two views of cryptography: The *computational view* and the *symbolic view*. In the computational view one deals with real implementations of cryptographic primitives, modeled for example by some kind of Turing machines. In this view messages may be viewed as bit strings. This low level view leads to tedious security proofs and there are not many tools for automatically proving security in the computational view (see for example Blanchet [Bla06]). In the symbolic view the level of abstraction is quite high: cryptographic primitives are modeled as black boxes and are assumed to function perfectly. In this view messages are often represented as terms over a suitable signature. One great advantage of the symbolic view is that there are numerous tools that support automatic verification of protocols, see for example Armando et al. [ABB$^+$05]. In this thesis we concentrate on the symbolic view of cryptography.

In the symbolic view of cryptography the capabilities of the intruder are mostly variants of a so called *Dolev-Yao intruder* going back to the work of Dolev and Yao [DY83]. The standard Dolev-Yao intruder is assumed to control the network, can construct messages from his knowledge and can send them to protocol participants of his choice. One of the formalisms (also used in [DY83]) to specify the actions of protocol participant and to describe the capabilities of the intruder to manipulate messages is *term rewriting*, i.e., actions of protocol participants and abilities of the intruder to manipulate messages are described by term rewriting rules.

When answering the three questions stated above the kind of security properties of cryptographic protocols that are investigated has a great impact. The security properties differ in the way they are taking the possible executions of a protocol into account. Here, we distinguish between *reachability properties* (which are special cases of *trace based properties*) and *strategy properties* (which are special cases of *branching time properties*).

Reachability Properties

As mentioned above one of the most fundamental type of cryptographic protocols are key exchange protocols. Of course, at the end of the protocol execution the key being exchanged should only be known to the two exchanging parties involved in the protocol. This security requirement can be formulated as a so called reachability property: is there a possible trace of the protocol execution where the intruder gets to know the secret key at some point? This problem is also referred to as the *secrecy problem.* We now give an overview over the answers given to the questions introduced above in the context of reachability properties.

Decidability The secrecy problem is undecidable when no bounds on the number of sessions are imposed is (see Even and Goldreich [EG83]). There are many classes of restrictions on modeling protocols that lead to decidability of the secrecy problem, for an intensive discussion see Durgin et al. [DLMS04].

Algorithms In [RT03] Rusinowitch and Turuani have shown that the secrecy problem for a bounded number of sessions in presence of a Dolev-Yao-Intruder is NP-complete, but their work does not provide an applicable algorithm for solving the secrecy problem. An algorithmic solution of the secrecy and related reachability problems was given in [MS01] by Millen and Shmatikov. The proposed algorithm is based on constraint solving. The constraint solving approach has been developed further and led to industrial strength tools for verifying cryptographic protocols, see, e.g., [ABB$^+$05, CV02].

Impossibility There is a fundamental impossibility concerning fairness of contract signing protocols. A contract signing protocol is unfair if there is a state reachable such that both signers have finished their execution of the protocol and one of the signers has a valid contract and the other signer does not have a valid contract. The intuition indicates that the asymmetry described above for the naïve approach of implementing the real world contract signing procedure as a protocol can not be resolved. That this is the case was proved in [EY80, PG99] and is often referred to as the impossibility of fair exchange without a trusted third party, i.e., there is no fair contract signing protocol only involving the two signers as participants.

Strategy properties

As an example for a security property that involves strategies we consider *balance* of contract signing protocols. Consider the following situation: During the execution of a contract signing protocol one of the two signers, say Alice, has both (i) the power to get a valid contract and (ii) the power to abort the contract signing procedure. In this situation Alice unilaterally can decide the outcome of the protocol and may use this to get an advantage. Such a state of the execution of a contract signing protocol is called *unbalanced*.

To model balance in a formal way one formalizes the notion of power by "has a strategy". Thus, a state of a protocol execution is called unbalanced for one of the signers if the other signer has two strategies: One strategy to obtain a valid contract as well as one to prevent the other signer from getting a valid contract.

In the following paragraphs we explain to which extend the guiding questions stated above were answered in the context of strategy properties.

Decidability The decidability problem concerning strategy properties as in the case of the secrecy problem was not studied to the same extend as in the secrecy problem. It is, for example, not known if the problem of deciding whether a strategy property is fulfilled by a protocol is decidable when resorting to a bounded number of sessions while imposing no bounds on the message size.

Algorithms In [KR02] Kremer and Raskin utilized alternating time temporal logic (ATL) [AHK02] to specify strategy security properties of contract signing protocols and used a model checker called MOCHA to analyze these protocols. In their analysis they resort to a finite state space for the possible execution of the considered protocols.

Impossibility Chadha et al. have shown in [CMSS05] that if a signer acts *optimistically*, i.e., he waits for messages of the other signer before contacting the trusted third party, the other signer has a strategy to get to a point where he has an advantage against the optimistic signer. In other words it is impossible to come up with a balanced protocol for an optimistic signer.

Contribution of this Thesis

The contribution of this thesis can be summarized as follows—as above, we use the questions stated above as a guideline.

Decidability We introduce the alternating μ-calculus (AMC) for cryptographic protocols and show in which cases this logic is decidable and in which cases it is not. We also give tight complexity bounds for the decidable classes of this problem.

Algorithms We extend the constraint solving approach developed for reachability properties to strategy properties and show how to utilize existing constraint solvers as a black box to decide strategy properties when a bounded number of sessions is considered and no bound on the message length is imposed.

Impossibility We give an alternative proof of the impossibility result given in [CMSS05] based on an axiomatic approach. In order to formulate the properties of protocols we extend ATL by what we call *move selectors*. With move selectors one can talk about different kinds of behaviors (such as honest, dishonest, and optimistic behavior) of participants in a natural way rather than model each kind of possible behavior in an ad hoc fashion.

Structure of this Thesis

In Chapter 2 we give preliminary notation on terms, messages, and rewrite rules. These notions are the basis for the formal modeling of messages and protocol rules in Chapters 3 and 4. We also describe the ASW protocol in more detail as this protocol serves as a running example throughout the rest of this thesis.

In Chapter 3 we show how to extend the constraint solving approach for analyzing reachability properties to a special case of strategy properties with which for example balance can be expressed. Our communication and intruder model is inspired by the one used in [CKS01] but instead of using the multiset rewriting approach utilized there we use the term rewriting approach. In our model participants are described as trees of rewrite rules. We use trees in order to model the fact that at certain points in a protocol execution participants must be able to choose between different alternative actions. Our constraint based algorithm for deciding strategy properties can use any constraint solver as a black box making it possible to extend existing constraint based algorithms for deciding reachability properties to handle strategy properties as well.

In Chapter 4 we introduce the alternating μ-calculus (AMC) as a logic for specifying security properties of cryptographic protocols. We interpret AMC formulas over concurrent game structures that are induced by protocol specifications. As in Chapter 3 participants are modeled as trees. The main differences to the model of Chapter 3 are that (i) principals act concurrently, i.e., in each step of the system each participant performs a move and that (ii) in one step participants can send and receive more than one message at a time. We introduce a fragment of AMC-formulas called $\mathcal{I}$-monotone. We show in which cases this fragment is decidable and in which cases this fragment is not decidable. For the decidable cases we give tight complexity bounds.

In Chapter 5 we show an impossibility result concerning contract signing protocols by showing that there is no concurrent game structure satisfying a specific combination of axioms. To formulate these axioms we extend ATL by so called move selectors. Move selectors can be used to describe different behaviors of participants in a natural way. We then use our result to give an alternative proof of the impossibility result of Chadha et al. [CMSS05] mentioned earlier in this introduction.

Pre-published Results

Chapter 3 is heavily based on a submitted paper to TOCL, which, in turn, is based on the two papers [KK05, KKW05]. Chapter 4 is based on a technical report [KKT07a] which is a long version of the paper [KKT07b].

Further Related Work

We now discuss related work concerning protocol analysis in the symbolic view.

One alternative computation model that is used for modelling cryptographic protocols and their execution are *process calculi*. The most prominent example is an extension of the π-calculus introduced by Milner et al. in [MPW92]. This extension is called the spi-calculus and was introduced by Abadi and Gordon in [AG97].

Another computation model used to analyze cryptographic protocols is the multiset rewriting (MSR) approach. The MSR approach to analyze cryptographic protocols was introduced by Cervesato et al. in [CDL$^+$99]. Based on this model contract signing protocols were analyzed by hand by Scedrov et al. in [CKS01].

A computation model inspired by graph theoretic means is the strand space model introduced by Guttman et al. in [THG99].

The first formal system for the specification of security properties of communication protocols is the Burrows-Abadi-Needham logic (BAN-logic), see [BAN90]. This logic and variants thereof where utilized to analyze security properties such as authentication and secrecy. Another logic used in the context of protocol analysis is the protocol composition logic (PCL) developed by Durgin et al. in [DMP03]. In the recent past Backes et al. used this logic to analyze contract signing protocols, see [BDD$^+$05]. In their analysis only trace-based properties are really addressed since the definition of the semantics of PCL does not take branching time behavior into account.

Contract signing protocols were analyzed with finite state model checkers, see for example Shmatikov et al. [SM02]. Drielsma and Mödersheim [DM04] analyzed contract signing protocols but the tools they used were not able to treat real branching time.

Multi party contract signing protocols are analyzed for example by Chadha et al. by applying ATL to specify security properties and using the model checker MOCHA to analyze a specific multi party contract signing protocol, see [CKS04].

Chapter 2

Preliminaries

In this chapter we introduce basic definitions and explanations that are needed in subsequent chapters. First, we will define how we model messages in our communication models used in Chapter 3 and Chapter 4. Second, we explain the Asokan-Shoup-Waidner (ASW) contract signing protocol in more detail, see [ASW98]. This protocol will serve as a running example throughout this thesis.

2.1 Terms and Messages

Our abstraction for messages are terms. Roughly speaking, the symbols of the corresponding signature correspond to cryptographic primitives. Which terms we consider is described by a grammar.

From now on let $\mathcal{V}$ be a finite set of variables, $\mathcal{A}$ a finite set of atoms, $\mathbb{K}$ a finite set of public and private keys, $\mathcal{A}_I$ an infinite set of intruder atoms. All these sets are assumed to be pairwise disjoint. The set $\mathbb{K}$ is partitioned into a set $\mathbb{K}_{\mathsf{pub}}$ of public keys and a set $\mathbb{K}_{\mathsf{priv}}$ of private keys and there is a bijective mapping $\cdot^{-1}\colon \mathbb{K} \to \mathbb{K}$ which assigns to every public key the corresponding private key and to every private key its corresponding public key.

Typically, the set $\mathcal{A}$ contains names of principals, atomic symmetric keys, and nonces (i.e., random numbers generated by principals). We note that we will allow non-atomic symmetric keys as well. The atoms in $\mathcal{A}_I$ are the nonces, symmetric keys, etc. the intruder may generate.

The following grammar defines the core of terms that are used in Chapters 3 and 4.

$$\begin{aligned}\mathcal{T}_{\mathsf{mes}} \quad ::= \quad & \mathcal{V} \mid \mathcal{A} \mid \mathcal{A}_I \mid \langle \mathcal{T}_{\mathsf{mes}}, \mathcal{T}_{\mathsf{mes}} \rangle \mid \\ & \{\mathcal{T}_{\mathsf{mes}}\}^s_{\mathcal{T}_{\mathsf{mes}}} \mid \{\mathcal{T}_{\mathsf{mes}}\}^a_{\mathbb{K}_{\mathsf{pub}}} \mid \mathsf{hash}(\mathcal{T}_{\mathsf{mes}}) \mid \\ & \mathsf{sig}(\mathbb{K}_{\mathsf{pub}}, \mathcal{T}_{\mathsf{mes}})\end{aligned}$$

In this grammar, $\mathcal{V}$, $\mathcal{A}$, $\mathcal{A}_I$, and $\mathbb{K}_{\mathsf{pub}}$ generate any of its elements.

Terms without variables are called *messages*. The set of messages will be denoted by $\mathcal{M}$.

As usual, $\langle t, t' \rangle$ is the pairing of t and t', the term $\{t\}^s_{t'}$ stands for the symmetric encryption of t by t' (note that the key t' may be any plain term), $\{t\}^a_k$ is the asymmetric encryption of t by k, the term $\mathsf{hash}(t)$ stands for the hash of t, and $\mathsf{sig}(k, t)$ is the signature on t which can be verified with the public key k.

Note that the above explanations are just to give some intuition on what these messages mean and how they can be used. In our formal model, there will be no restriction on the use of these messages in protocol descriptions except for the rules by which the intruder can derive messages.

By $\mathcal{V}(t)$ we denote the set of variables occurring in a term t. The set $\mathrm{Sub}(t)$ of subterms of a term t and the set $\mathrm{Sub}(E)$ of subterms of the terms in a set E of terms are defined in the obvious way.

A *substitution* assigns terms to variables, that is, it is a function $\mathcal{V} \to \mathcal{T}$. The *domain* of a substitution σ is denoted by $\mathrm{dom}(\sigma)$ and defined by $\mathrm{dom}(\sigma) = \{x \in \mathcal{V} \mid \sigma(x) \neq x\}$. Substitutions are required to have finite domains. A substitution σ is called *ground* if $\sigma(x)$ is a message for each $x \in \mathrm{dom}(\sigma)$. Given two substitutions σ and τ with disjoint domains, their union $\sigma \cup \tau$ is defined in the obvious way. Given a term t and a substitution σ, the term $t\sigma$ is obtained from t by simultaneously replacing each occurrence of a variable x in t by $\sigma(x)$. If σ and τ are two substitutions, then their *composition*, denoted $\sigma \circ \tau$, is the substitution defined by $(\sigma \circ \tau)(x) = (x\tau)\sigma$ for every $x \in \mathcal{V}$.

We say that a term t is used as a *verification key* in a term t' if t' has a subterm of the form $\{s\}^s_t$, $\{s\}^a_{t^{-1}}$, or $\mathsf{sig}(t, s)$ for some term s. Intuitively, a verification key is needed to decrypt messages or verify signatures. For example, if a principal A receives the message $m = \{\langle A, B \rangle\}^s_a$ and wants to check whether the plain text matches with $t' = \langle A, x \rangle$, then the principal first needs to decrypt m with the verification key a.

2.2 ASW Protocol

In this section we explain a well known two party contract signing protocol: The Asokan-Shoup-Waidner (ASW) protocol. Our informal description of the ASW protocol follows [SM02] (see this work or [ASW98] for more details).

We write $\mathsf{sig}[k, m]$ as abbreviation for $\langle m, \mathsf{sig}(k, m)\rangle$ and sometimes write $\langle m_1, \ldots, m_n\rangle$ instead of $\langle m_1, \langle m_2, \langle \cdots \langle m_{n-1}, \rangle\rangle\rangle\rangle$. We denote the public or verification key of a principal A by k_A.

The ASW protocol enables two principals A (the originator) and B (the responder) to obtain each other's signature on a previously agreed contractual text $\mathsf{contract}$ with the help of a trusted third party (TTP) T, which however is only invoked in case of problems. In other words, the ASW protocol is an optimistic two-party contract-signing protocol.

There are two kinds messages that are considered a valid contract in the ASW protocol:

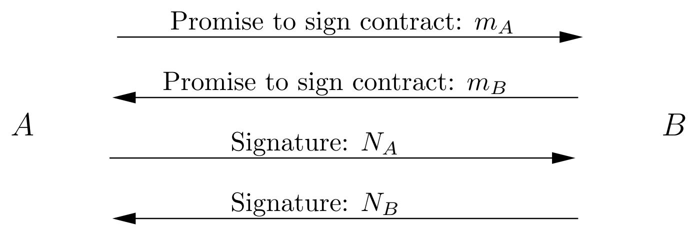
the standard contract $\langle \mathsf{sig}[k_A, m_A], N_A, \mathsf{sig}[k_B, m_B], N_B\rangle$ and

the replacement contract $\mathsf{sig}[k_T, \langle \mathsf{sig}[k_A, m_A], \mathsf{sig}[k_B, m_B]\rangle]$,

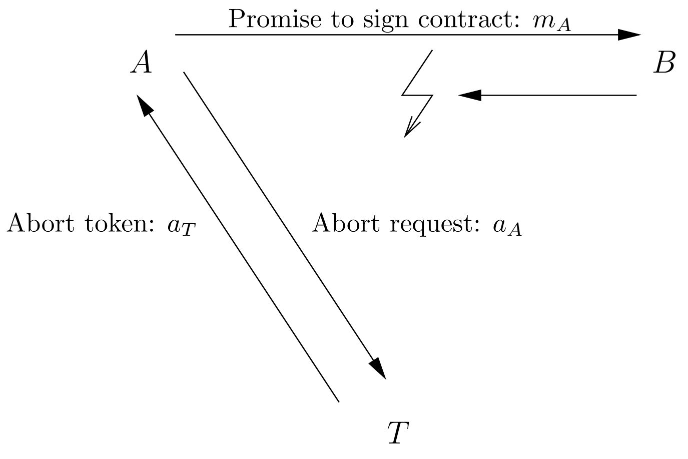
where $m_A = \langle k_A, k_B, k_T, \mathsf{contract}, \mathsf{hash}(N_A)\rangle$, $m_B = \langle \mathsf{sig}[k_A, m_A], \mathsf{hash}(N_B)\rangle$, and N_A and N_B are nonces.

The ASW protocol consists of four interdependent subprotocols: the exchange and abort protocol and two resolve protocols. These subprotocols are explained next.

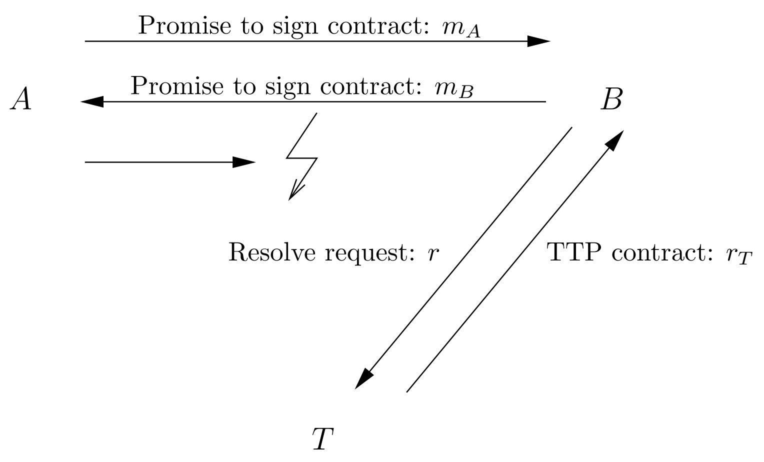
Exchange protocol. The basic idea of the *exchange protocol* is that A first indicates her interest to sign the contract. To this end, she sends to B the message $\mathsf{sig}[k_A, m_A]$ as defined above, where N_A is a nonce generated by A. By sending this message, A "commits" to signing the contract. Then, similarly, B indicates his interest to sign the contract by generating a nonce N_B and sending the message $\mathsf{sig}[k_B, m_B]$ to A. Finally, first A and then B reveal N_A and N_B, respectively. The following diagram shows how the exchange protocol functions:


Promise to sign contract: m_A
Promise to sign contract: m_B
A
B
Signature: N_A
Signature: N_B

Abort protocol. If, after A has sent her first message, B does not respond or B's answer is not delivered by the network, A may contact T to abort, i.e., A runs the abort protocol with T. In the abort protocol, A first sends the message $a_A = \mathsf{sig}[k_A, \langle \mathsf{aborted}, \mathsf{sig}[k_A, m_A]\rangle]$ as an abort request. If T has not received a resolve request before (see below), then T sends back to A the *abort token* $a_T = \mathsf{sig}[k_T, \langle \mathsf{aborted}, a_A\rangle]$. Otherwise (if T received a resolve request, which in particular involves the messages $\mathsf{sig}[k_A, m_A]$ and $\mathsf{sig}[k_B, m_B]$ from above), it sends the *replacement contract* $r_T = \mathsf{sig}[k_T, r]$ to A with $r = \langle \mathsf{sig}[k_A, m_A], \mathsf{sig}[k_B, m_B]\rangle$. The following diagram shows one possible situation in which A starts the abort protocol:



Resolve protocol. If, after A has sent the nonce N_A, B does not respond, A may contact T to resolve, i.e., A runs the resolve protocol with T. In the resolve protocol, A sends the message $r = \langle \mathsf{sig}[k_A, m_A], \mathsf{sig}[k_B, m_B]\rangle$ to T as a resolve request. If T has not sent out an abort token before, then T returns the replacement contract r_T, and otherwise T returns the abort token a_T. Analogously, if, after B has sent his commitment to sign the contract, A does not respond, B may contact T to resolve, i.e., B runs the resolve protocol with T similarly to the case for A. Note that contacting T is again a non-deterministic action of B. This situation is depicted in the following diagram:


Promise to sign contract: m_A
Promise to sign contract: m_B
A
B
Resolve request: r
TTP contract: r_T
T

We note that the communication with T (for both A and B) is carried out over secure channels.

Chapter 3

Constraint-based Algorithm for Strategy Properties

As mentioned in the introduction constraint solving procedures have proven to be useful in the automatic analysis of reachability properties of cryptographic protocols, see for example [MS01, CV01, BMV03]. In this chapter we show how the constraint based approach can be extended in a way that automatic analysis of a specific type of branching time properties of cryptographic protocols becomes possible.

The protocol and intruder model that we suggest to use is similar to a model proposed in [CKS01]; it contains different features important for contract signing protocols, which are absent in the models for authentication and key exchange protocols referred to above. First, as in [CKS01], we model write-protected channels which are *not* under the control of the intruder. For simplicity, we call these channels *secure channels.* Second, for protocols in our model we explicitly define the induced transition systems. These transition systems have infinitely many states and are infinitely branching, but have paths of bounded length; they allow us to state crucial properties of contract-signing properties.

Our main result is that for the transition system induced by a cryptographic protocol properties expressing the existence of certain strategies for the intruder are decidable.

In a nutshell, our constraint-based algorithm works as follows: Given a protocol along with a game-theoretic security property, first the algorithm guesses what we call a symbolic branching structure. This structure represents a potential attack on the protocol. In the second step of the algorithm, the symbolic branching structure is turned into a constraint system. Then a standard constraint-solving procedure is used to compute a finite sound and complete set of so-called simple constraint systems. A simple constraint system in such a set represents a (possibly infinite) set of solutions of the original constraint system and a sound and complete set of these simple constraint systems represents the set of *all* solutions of the original constraint system. Finally it is checked whether (at least) one of the computed simple constraint systems satisfies certain requirements. The first step as described above is analogous to the first step in a constraint-based algorithm for checking reachability properties of cryptographic protocols, where an interleaving of the actions of the individual principals is guessed.

3.1 The Protocol and Intruder Model

As mentioned in the introduction, our model is quite similar to the one by Chadha et al. [CKS01]. When it comes to the technical exposition, our approach is, however, inspired by the term-rewriting approach of [RT03] rather than the multi-set rewriting approach of [CKS01]. We start with a rather precise, but informal description of our model.

In our model, a protocol is a finite set of principals and every principal is a finite tree, which represents all possible behaviors of the principal. Each edge of such a tree is labeled by a rewrite rule, which describes the receive-send action that is performed when the principal takes this edge in a run of the protocol.

When a principal carries out a protocol, it traverses its tree, starting at the root. In every node, the principal takes its current input, chooses one of the edges leaving the node, matches the current input with the left-hand side of the rule the edge is labeled with, sends out the message which is determined by the right-hand side of the rule, and moves to the node the chosen edge leads to. Whether or not a principal gets an input and which input it gets is determined by the intruder and the secure channels. The intruder receives every message sent by a principal, can use all the messages he has received to construct new messages, and can provide input messages to any principal he wants.

The above is very similar to standard Dolev-Yao models for reachability properties (see, e.g., [RT03, MS01]). There are, however, two important ingredients that are not present in these models: secure channels and an explicit branching structure. In the remainder of this introduction to our model, we will explain these two ingredients in more detail.

Unlike in the standard Dolev-Yao models, in our model the input of a principal may not only come from the intruder but also from a so-called secure channel. While, as in [CKS01], a secure channel is not read-protected (the intruder can read the messages written onto this channel), the intruder does not control this channel. That is, he cannot delay, duplicate, or remove messages, or write messages onto this channel under a fake identity (unless he has corrupted a party).

As mentioned in the introduction, unlike authentication and key-exchange protocols, properties of contract-signing and related protocols cannot be stated as reachability properties, i.e., in terms of single runs of a protocol alone. One rather has to consider branching properties. We therefore describe the behavior of a protocol as an infinite-state transition system which comprises all runs of a protocol. To be able to express properties of contract-signing protocols we distinguish several types of transitions: there are intruder transitions (just as in [RT03, MS01]); there are ε-transitions, which can be used to model that a subprotocol is spawned without waiting for input from the intruder; and there are secure channel transitions, which model communication via secure channels. Since the intruder can construct an infinite number of messages, the transition system will have an infinite number of states, but it will have paths of a bounded length.

In the following subsections, we present a formal definition of our model.

3.1.1 Terms and Messages

In the model used in this chapter messages do not only contain ordinary message content but also specify by which means the message is communicated (i) over the network or (ii) over a secure channel. This is the reason why we extend the core grammar given in Chapter 2 by *secure channel terms*:

$$\begin{aligned}\mathcal{T}_{\mathsf{sc}} &::= \mathsf{sc}(\mathcal{N}, \mathcal{N}, \mathcal{T}_{\mathsf{mes}})\\ \mathcal{T} &::= \mathcal{T}_{\mathsf{mes}} \mid \mathcal{T}_{\mathsf{sc}} \mid \mathcal{N}\end{aligned}$$

Here the finite set $\mathcal{N}$ comprises all principle names that are present in the protocol.

A secure channel term of the form $\mathsf{sc}(n, n', t)$ stands for a term t on the secure channel from n to n'. A principal may only generate such a term if he knows n and t (but does not need to know n'). This guarantees that a principal cannot impersonate other principals on secure channels. Hence, knowing n grants access to secure channels with sender address n.

3.1.2 Intruder

Given a set $\mathcal{K}$ of terms, the (infinite) set $d(\mathcal{K})$ of terms the intruder can derive from $\mathcal{K}$ is the smallest set satisfying the following conditions:

(D1) $\mathcal{K} \subseteq d(\mathcal{K})$.

(D2) *Composition and decomposition*: If $t, t' \in d(\mathcal{K})$, then $\langle t, t'\rangle \in d(\mathcal{K})$. Conversely, if $\langle t, t'\rangle \in d(\mathcal{K})$, then $t \in d(\mathcal{K})$ and $t' \in d(\mathcal{K})$.

(D3) *Symmetric encryption and decryption*: If $t, t' \in d(\mathcal{K})$, then $\{t\}^s_{t'} \in d(\mathcal{K})$. Conversely, if $\{t\}^s_{t'} \in d(\mathcal{K})$ and $t' \in d(\mathcal{K})$, then $t \in d(\mathcal{K})$.

(D4) *Asymmetric encryption and decryption*: If $t \in d(\mathcal{K})$ and $k \in d(\mathcal{K}) \cap \mathbb{K}_{\mathsf{pub}}$, then $\{t\}^a_k \in d(\mathcal{K})$. Conversely, if $\{t\}^a_k \in d(\mathcal{K})$ and $k^{-1} \in d(\mathcal{K})$, then $t \in d(\mathcal{K})$.

(D5) *Hashing*: If $t \in d(\mathcal{K})$, then $\mathsf{hash}(t) \in d(\mathcal{K})$.

(D6) *Signing*: If $t \in d(\mathcal{K})$ and $k^{-1} \in d(\mathcal{K}) \cap \mathbb{K}_{\mathsf{priv}}$, then $\mathsf{sig}(k, t) \in d(\mathcal{K})$. (The signature contains the public key but can only be generated if the corresponding private key is known.)

(D7) *Writing onto secure channels:* If $t \in d(\mathcal{K})$, $n \in d(\mathcal{K}) \cap \mathcal{N}$, and $n' \in \mathcal{N}$, then $\mathsf{sc}(n, n', t) \in d(\mathcal{K})$.

(D8) *Generating fresh constants*: $\mathcal{A}_I \subseteq d(\mathcal{K})$.

Each of the above rules only applies when the resulting expression is a term according to the grammar stated above. For instance, a hash of a secure channel term is not a term, so (D5) does not apply when t is of the form $\mathsf{sc}(n, n', t')$.

Note that the above definition is slightly more general than in other papers: Typically, one only considers sets $\mathcal{K}$ which consist exclusively of messages. Later, the more general definition, which allows arbitrary sets of terms for $\mathcal{K}$, will be useful.

In (D7), the precondition $n \in d(\mathcal{K}) \cap \mathcal{N}$ means that the intruder has corrupted the principal with address n and therefore can impersonate this principal when writing onto the secure channel.

3.1.3 Principals and Protocols

Principal rules are of the form $R \Rightarrow S$ where $R \in \mathcal{T} \cup \{\varepsilon\}$ and $S \in \mathcal{T}$. We refer to R as the *left-hand side (LHS)* of $R \Rightarrow S$ and to S as the *right-hand side (RHS)* of $R \Rightarrow S$.

A *rule tree* is a finite rooted tree where the edges are labeled by principal rules. Formally, a rule tree is a tuple $\Pi = (V, E, r, \ell)$ where (V, E) is a finite rooted tree as usual (in particular, $E \subseteq V \times V$), r is the root of (V, E), and ℓ maps every edge $(v, v') \in E$ to a principal rule, *the principal rule associated with* (v, v').

A *principal* is a rule tree Π as above which satisfies the following two conditions for every vertex $v \in V$ where we assume that $v_0, \ldots, v_n$ is the unique path from the root r to v and where $R_i \Rightarrow S_i$ is the principal rule associated with (v_i, v_{i+1}), for every $i < n$.

1. *Well-formedness:* $\mathcal{V}(S_{n-1}) \subseteq \bigcup_{i<n} \mathcal{V}(R_i)$.
2. *Feasibility:* $t \in d(\{R_0, \ldots, R_{n-1}\} \cup \mathcal{M})$ for every verification key t in R_{n-1}.

Well-formedness requires that every variable occurring on the RHS of a principal rule of an edge e also occurs on the LHS of the principal rule of some edge on the path from the root to e (inclusive). It guarantees that any output produced by a principal only depends on what he has received, which is a minimum requirement for a reasonable protocol specification. This condition is standard in Dolev-Yao models, see, for instance, [RT03, MS01].

Feasibility makes sure that a principal cannot match variables in keys, which would not be feasible in practice. For example, this condition is not satisfied for a principal whose rule tree only contains one edge and where this edge is labeled with the principal rule $\{y\}^s_x \Rightarrow x$. On the other hand, it is satisfied for a principal whose rule tree is a path of length two and where the first edge is labeled by $x \Rightarrow \langle x, x\rangle$ and the second edge is labeled as before (by $\{y\}^s_x \Rightarrow x$).

Let Π be a rule true. When v is a vertex of Π, we write $\Pi{\downarrow}v$ for the subtree of Π rooted at v. When σ is a substitution, we write $\Pi\sigma$ for the rule tree obtained from Π by simultaneously applying σ to all the principal rules (each side of them) of Π.

A *protocol* is a tuple $P = ((\Pi_1, \ldots, \Pi_n), \mathcal{K})$ which consists of a tuple of principals and a finite set $\mathcal{K} \subseteq \mathcal{M} \cup \mathcal{A} \cup \mathcal{N}$ of messages, the *initial intruder knowledge*. We require that each variable occurs in the rules of at most one principal, i.e., different principals must have disjoint sets of variables. We assume that intruder atoms, i.e., elements of $\mathcal{A}_I$, do not occur in P (neither in any principal rule nor in $\mathcal{K}$).

3.1.4 The Transition Graph of a Protocol

The transition graph $\mathcal{G}_P$ induced by a protocol P comprises all runs of a protocol. To define this graph formally, we first introduce states and transitions between these states.

A *state* is of the form $((\Pi_1, \ldots, \Pi_n), \sigma, \mathcal{K}, \mathcal{S})$ where

1. σ is a ground substitution,
2. Π_i is a rule tree such that $\Pi_i\sigma$ is a principal for each i,
3. $\mathcal{K}$ is a finite set of messages, the *intruder knowledge*, and
4. $\mathcal{S}$ is a finite multi-set of secure channel messages, the *secure channel.*

The idea is that when the transition system is in such a state, then the substitution σ has been performed, the accumulated intruder knowledge is what can be derived from $\mathcal{K}$, the secure channels hold the messages in $\mathcal{S}$, and for each i, $\Pi_i\sigma$ is the "remaining protocol" to be carried out by principal i. This also explains why $\mathcal{S}$ is a multi-set: messages sent several times should be delivered several times.

There are three kinds of transitions between states: intruder, secure channel, and ε-transitions. In what follows, for every $i \in \{1, \ldots, n\}$, let

$\Pi_i = (V_i, E_i, r_i, \ell_i)$ and $\Pi'_i = (V'_i, E'_i, r'_i, \ell'_i)$ denote rule trees. A transition

$$((\Pi_1, \dots, \Pi_n), \sigma, \mathcal{K}, \mathcal{S}) \xrightarrow{\tau} ((\Pi'_1, \dots, \Pi'_n), \sigma', \mathcal{K}', \mathcal{S}')$$

with label τ exists if one of the three following conditions is satisfied:

1. *Intruder transition,* $\tau = [i, m, I]$: There exists a vertex $v \in V_i$ such that $(r_i, v) \in E_i$ and $\ell_i(r_i, v) = R \Rightarrow S$, and there exists a ground substitution σ'' with $\mathrm{dom}(\sigma'') \subseteq \mathcal{V}(R\sigma)$ such that
   (a) $m \in d(\mathcal{K})$,
   (b) $\sigma' = \sigma \cup \sigma''$,
   (c) $R\sigma' = m$,
   (d) $\Pi'_j = \Pi_j$ for every $j \neq i$, $\Pi'_i = \Pi_i{\downarrow}v$,
   (e) either $S \in \mathcal{T}_{\mathsf{mes}}$ and $\mathcal{K}' = \mathcal{K} \cup \{S\sigma'\}$, or $S = \mathsf{sc}(\cdot, \cdot, t)$ for some term t and $\mathcal{K}' = \mathcal{K} \cup \{t\sigma'\}$, and
   (f) either $S \in \mathcal{T}_{\mathsf{mes}}$ and $\mathcal{S}' = \mathcal{S}$, or $S \in \mathcal{T}_{\mathsf{sc}}$ and $\mathcal{S}' = \mathcal{S} \cup \{S\sigma'\}$.

   This transition models that principal i receives the the message m from the intruder (i.e., from the public network). Note that the second clause in (e) accounts for the fact that our secure channels are not read-protected.

2. *Secure channel transition,* $\tau = [i, m, \mathsf{sc}]$: There exists a vertex $v \in V_i$ with $(r_i, v) \in E_i$ and $\ell_i(r_i, v) = R \Rightarrow S$, and there exists a ground substitution σ'' with $\mathrm{dom}(\sigma'') \subseteq \mathcal{V}(R\sigma)$ such that
   (a) $m \in \mathcal{S}$,
   (b)–(e) as above, and
   (f) either $S \in \mathcal{T}_{\mathsf{mes}}$ and $\mathcal{S}' = \mathcal{S} \setminus \{m\}$, or $S \in \mathcal{T}_{\mathsf{sc}}$ and $\mathcal{S}' = (\mathcal{S} \setminus \{m\}) \cup \{S\sigma'\}$.

   This transition models that principal i reads message m from a secure channel.

3. *ε-transition,* $\tau = [i]$: There exists a vertex $v \in V_i$ with $(r_i, v) \in E_i$ and $\ell_i(r_i, v) = \varepsilon \Rightarrow S$ such that $\sigma' = \sigma$ and (d)–(f) as given above for intruder transitions.

   This transition models that principal i performs a step where neither a message is read from the intruder nor from a secure channel.

We call i the *principal associated with the transition*, $R \Rightarrow S$ (for the intruder and secure channel transitions) and $\varepsilon \Rightarrow S$ (for the ε-transitions), respectively, the *principal rule associated with the transition*, and v the *principal vertex associated with the transition*.

Definition 3.1.1 (transition graph) *Let $P = ((\Pi_1, \ldots, \Pi_n), \mathcal{K})$ be a protocol.*

The initial state *of P is $q_P = ((\Pi_1, \ldots, \Pi_n), \sigma, \mathcal{K}, \emptyset)$ where σ is the substitution with empty domain. The* set of states *of P, denoted S_P, is the set of states which are reachable from q_P by a sequence of transitions.*

The transition graph *$\mathcal{G}_P$ of P is the tuple (S_P, E_P, q_P) where q_P is the initial state of P, S_P is the set of states of P, and E_P is the set of all transitions among the states of P.*

For convenience, we write $q \in \mathcal{G}_P$ when q is a state in $\mathcal{G}_P$ and we write $q \xrightarrow{\tau} q' \in \mathcal{G}_P$ when $q \xrightarrow{\tau} q'$ is a transition in $\mathcal{G}_P$.

Remark 3.1.2 *The transition graph $\mathcal{G}_P$ of P is a DAG since by performing a transition the size of the first component of a state decreases. While the graph may be infinite branching, the maximal length of a path in this graph is bounded by the total number of edges in the principals Π_i of P.*

3.2 Modeling the Originator of the ASW Protocol

To demonstrate that our framework can actually be used to analyze contract-signing protocols, we show how the originator of the Asokan-Shoup-Waidner (ASW) protocol [ASW98] can be modeled. In a similar fashion, other contract-signing protocols can be dealt with.

3.2.1 The Principal O

The principal O is defined by the tree Π_O depicted in Figure 3.1 where the edge labels for the principal rules defined below. Rules $e1$, $e2$, and $e3$ belong to the exchange protocol, rules $a1$, $a2$, and $a3$ belong to the abort protocol, and rules $r1$, $r2$, and $r3$ belong to the resolve protocol of O.

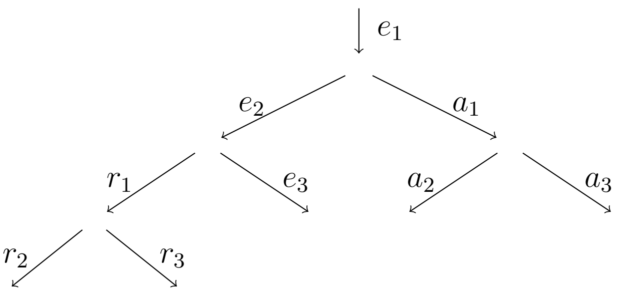


Figure 3.1: A model of the Originator O in the ASW protocol

Exchange protocol. The actions performed in the exchange protocol have informally been discussed above.

Abort protocol. If, after the first step of the exchange protocol, O does not get an answer back from R, the principal O may start the abort protocol, i.e., send an abort request via a secure channel to T (rule $a1$). Then, T will either confirm the abort of the protocol by returning an abort token—in this case O will continue with rule $a3$—or send a replacement contract—in this case O will continue with rule $a2$. (The trusted third party T sends a replacement contract if R previously contacted T to resolve the protocol run.)

Resolve protocol. If after rule $e2$, i. e., after sending N_O, the principal O does not get an answer back from R, then O can start the resolve protocol by sending a resolve request to T via the secure channel (rule $r1$). After that, depending on the answer returned from T (which again will return an abort token or a replacement contract), one of the rules $r2$ or $r3$ is performed.

We now present the principal rules for O where the numbering corresponds to the one in Figure 3.1.

(e_1) $\varepsilon \Rightarrow me_1$ where

$$me_1 = \mathsf{sig}[k_O, me_2] \quad \text{and} \quad me_2 = \langle k_O, k_R, k_T, \mathsf{contract}, \mathsf{hash}(N_O)\rangle\,.$$

(e_2) $\mathsf{sig}[k_R, me_3] \Rightarrow N_O$ where $me_3 = \langle me_1, \mathsf{hash}(x)\rangle$.

(e_3) $x \Rightarrow \mathsf{OHasValidContract}$.

(a_1) $\varepsilon \Rightarrow \mathsf{sc}(O, T, ma_1)$ where $ma_1 = \mathsf{sig}[k_O, \langle \mathsf{aborted}, me_1\rangle]$.

(a_2) $\mathsf{sc}(T, O, ma_2) \Rightarrow \mathsf{OHasValidContract}$ where

$$ma_2 = \mathsf{sig}[k_T, \langle me_1, me_4\rangle] \quad \text{and} \quad me_4 = \mathsf{sig}[k_R, \langle me_1, z_1\rangle].$$

$$\begin{tikzcd}
\Pi_1 : & & f_1 \arrow[d, "\mathsf{sc}(2,1,\langle x,b\rangle) \Rightarrow x"'] & & \Pi_2 : & & g_1 \arrow[dl, "\epsilon \Rightarrow \mathsf{sc}(2,1,\langle a,b\rangle)"'] \arrow[dr, "\epsilon \Rightarrow \mathsf{sc}(2,1,\langle b,b\rangle)"] & \\
& & f_2 \arrow[d, "\{y\}_k^s \Rightarrow y"'] & & & g_2 & & g_3 \\
& & f_3 \arrow[dl, "a \Rightarrow c_1"'] \arrow[d, "x \Rightarrow c_2"] \arrow[dr, "y \Rightarrow c_2"] & & & & & \\
& f_4 & f_5 & f_6 & & & &
\end{tikzcd}$$

Figure 3.2: Protocol $P_{\mathsf{ex}} = (\{\Pi_1, \Pi_2\}, \mathcal{I}_0)$ with $\mathcal{I}_0 = \{\{a\}_k^s, \{b\}_k^s\}$

(a_3) $\mathsf{sc}(T, O, \mathsf{sig}[k_T, \langle \mathsf{aborted}, ma_1\rangle]) \Rightarrow \mathsf{OHasAbortToken}$.

(r_1) $\varepsilon \Rightarrow \mathsf{sc}(O, T, \langle me_1, \mathsf{sig}[k_R, me_3]\rangle)$.

(r_2) $\mathsf{sc}(T, O, \mathsf{sig}[k_T, \langle \mathsf{aborted}, ma_1\rangle]) \Rightarrow \mathsf{OHasAbortToken}$.

(r_3) $\mathsf{sc}(T, O, mr_1) \Rightarrow \mathsf{OHasValidContract}$ where

$$mr_1 = \mathsf{sig}[k_T, \langle me_1, mr_2\rangle] \quad \text{and} \quad mr_2 = \mathsf{sig}[k_R, \langle me_1, z_2\rangle].$$

3.3 Strategies, Strategy Properties, and Strategy Problems

In this section, we define intruder strategies on transition graphs of protocols, the type of goal these strategies try to achieve, and the formal decision problems we are interested in. We start with an informal explanation following an example and then turn to precise statements.

Throughout this and the following sections, we will use the protocol P_{ex} depicted in Figure 3.2 as our running example. This protocol consists of two principals Π_1 and Π_2 and the initial knowledge $\mathcal{K}_0 = \{\{a\}_k^s, \{b\}_k^s\}$ of the intruder. Informally speaking, Π_2 can, without waiting for input from the secure channel or the intruder, decide whether to write $\langle a, b\rangle$ or $\langle b, b\rangle$ into the secure channel from Π_2 to Π_1. While the intruder can read the message written into this channel, he cannot modify or delay this message. Also, he cannot insert his own message into this channel, for he does not have the principal address 2 in his intruder knowledge, and hence, cannot generate messages of the form $\mathsf{sc}(2, n, t)$ for any name n. Consequently, such messages must come from Π_2. Principal Π_1 first waits for a message of the form $\langle x, b\rangle$ in the secure channel from Π_2 to Π_1. In case Π_2 wrote, say, $\langle a, b\rangle$ into this channel, x is substituted by a, and this message is written into the network, and hence, given to the intruder. Next, Π_1 waits for input of the form $\{y\}^s_k$. This is not a secure channel term, and thus, comes from the intruder. In case the intruder sends $\{b\}^s_k$, say, then y is substituted by b. Finally, Π_1 waits for input of the form a (in the edges from f_3 to f_4 and f_3 to f_5) or b (in the edge from f_3 to f_6). Recall that x was substituted by a and y by b. If the intruder sends b, say, then Π_2 takes the edge from f_3 to f_6 and outputs c_2 into the network. If the intruder had sent a, Π_1 could have chosen between the first two edges. We note that this protocol is not meant to perform a useful task. It is rather designed to illustrate different aspects of our constraint-based algorithm that would be more cumbersome to demonstrate by realistic protocols, such as the ASW protocol.

3.3.1 Intruder Strategies and Strategy Trees

To define intruder strategies, we introduce the notion of a strategy tree, which captures that the intruder has a way of acting such that regardless of how the other principals act he achieves a certain goal, where goal in our context means that a state will be reached where the intruder can derive certain constants and cannot derive others, e.g., for balance, the intruder tries to obtain $\mathsf{IntruderHasContract}$ but tries to prevent $\mathsf{HonestPartyHasContract}$ from occurring. More concretely, let us consider the protocol P_{ex} depicted in Figure 3.2. We want to know if the intruder has a strategy to get to a state where he can derive atom c_2 but not atom c_1 (no matter what the principals Π_1, Π_2, and the secure channels do). Such a strategy of the intruder has to deal with both decisions principal Π_2 may make in the first step because the intruder cannot control which edge is taken by Π_2. It turns out that regardless of which message is sent by principal Π_2 in its first step, the following simple strategy allows the intruder to achieve his goal: The intruder can send $\{b\}^s_k$ to principal Π_1 in the second step of Π_1 and in the last step of Π_1, the intruder sends b to principal Π_1. This guarantees that in the last step of Π_1, the left-most edge is never taken, and thus, c_1 is not returned, but

$$\begin{tikzcd}
\mathcal{T}_{ex}: & h_1\ ((f_1,g_1),\emptyset,\mathcal{K}_0,\emptyset) \arrow[dl, "{[2]}"'] \arrow[dr, "{[2]}"] & & \\
((f_1,g_2),\emptyset,\mathcal{K}_1,\{\mathsf{sc}(2,1,\langle a,b\rangle)\})\ h_2 \arrow[d, "{[1,\mathsf{sc}(2,1,\langle a,b\rangle),\mathsf{sc}]}"'] & & h_6\ ((f_1,g_3),\emptyset,\mathcal{K}_2,\{\mathsf{sc}(2,1,\langle b,b\rangle)\}) \arrow[d, "{[1,\mathsf{sc}(2,1,\langle b,b\rangle),\mathsf{sc}]}"] & \\
((f_2,g_2),\sigma_1,\mathcal{K}_1\cup\{a\},\emptyset)\ h_3 \arrow[d, "{[1,\{b\}^s_k,I]}"'] & & h_7\ ((f_2,g_3),\sigma_3,\mathcal{K}_2\cup\{b\},\emptyset) \arrow[d, "{[1,\{b\}^s_k,I]}"] & \\
((f_3,g_2),\sigma_2,\mathcal{K}_1\cup\{a,b\},\emptyset)\ h_4 \arrow[d, "{[1,b,I]}"'] & & h_8\ ((f_3,g_3),\sigma_4,\mathcal{K}_2\cup\{b\},\emptyset) \arrow[dl, "{[1,b,I]}"'] \arrow[dr, "{[1,b,I]}"] & \\
((f_6,g_2),\sigma_2,\mathcal{K}_1\cup\{a,b,c_2\},\emptyset)\ h_5 & h_9\ ((f_5,g_3),\sigma_4,\mathcal{K}_2\cup\{b,c_2\},\emptyset) & & h_{10}\ ((f_6,g_3),\sigma_4,\mathcal{K}_2\cup\{b,c_2\},\emptyset)
\end{tikzcd}$$

Figure 3.3: Strategy tree $\mathcal{T}_{ex}$ for P_{ex} with $\mathcal{K}_1 = \mathcal{K}_0 \cup \{\langle a,b\rangle\}$, $\mathcal{K}_2 = \mathcal{K}_0 \cup \{\langle b,b\rangle\}$, $\sigma_1 = \{x \mapsto a\}$, $\sigma_2 = \sigma_1 \cup \{y \mapsto b\}$, $\sigma_3 = \{x \mapsto b\}$, and $\sigma_4 = \sigma_3 \cup \{y \mapsto b\}$. Also, for brevity of notation, in the first component of the states we write, for instance, f_1 instead of $\Pi_1{\downarrow}f_1$. The strategy property we consider is $((C_{ex}, C'_{ex})) = ((\{c_2\},\{c_1\}))$.

at least one of the other two edges can be taken, which in any case yields c_2. Formally, such strategies are defined as trees. In our example, the strategy tree corresponding to the strategy informally explained above is depicted in Figure 3.3. Its construction is in accordance with the following definition of strategy trees.

Definition 3.3.1 (strategy tree) *Let P be a protocol with n principals and $q \in \mathcal{G}_P$. A* q-strategy tree *is a rooted tree $\mathcal{T}_q = (V, E, r, \ell_V, \ell_E)$ where every vertex $v \in V$ is labeled by a state, $\ell_V(v) \in \mathcal{G}_P$, and every edge $(v,v') \in E$ is labeled by a label of a transition, $\ell_E(v,v')$, such that the following conditions are satisfied:*

(T1) $\ell_V(r) = q$.

(T2) $\ell_V(v) \xrightarrow{\ell_E(v,v')} \ell_V(v') \in \mathcal{G}_P$ *for all* $(v,v') \in E$.

(T3) Whenever $\ell_V(v) = q'$ and $q' \xrightarrow{[j]} q'' \in \mathcal{G}_P$ for some $v \in V$, $q', q'' \in \mathcal{G}_P$, and $j \in \{1,\ldots,n\}$, then there exists $v'' \in V$ such that $(v,v'') \in E$, $\ell_V(v'') = q''$, and $\ell_E(v,v'') = [j]$.

(T4) Whenever $\ell_V(v) = q'$ and $q' \xrightarrow{[j,m,\mathsf{SC}]} q'' \in \mathcal{G}_P$ for some $v \in V$, $q', q'' \in$

$\mathcal{G}_P$, $j \in \{1, \ldots, n\}$, and $m \in \mathcal{M}$, then there exists $v'' \in V$ such that $(v, v'') \in E$, $\ell_V(v'') = q''$, and $\ell_E(v, v'') = [j, m, \mathsf{sc}]$.

(T5) Whenever $(v, v') \in E$, $\ell_E(v, v') = [j, m, I]$, and there exists $q' \neq \ell_V(v')$ with $\ell_V(v) \xrightarrow{[j,m,I]} q' \in \mathcal{G}_P$ for some $v \in V$, $q' \in \mathcal{G}_P$, $j \in \{1, \ldots, n\}$, and $m \in \mathcal{M}$, then there exists v'' with $(v, v'') \in E$, $\ell_E(v, v'') = [j, m, I]$ and $\ell_V(v'') = q'$.

(T2) guarantees that all edges in $\mathcal{T}_q$ correspond to transitions in $\mathcal{G}_P$. (T3) says that every ε-transition of the transition graph of P must be present in the strategy graph. This is because the intruder should not be able to prevent a principal from performing an ε-rule, since these rules do not depend on input from the intruder. (T4) is similar: The intruder should not be able to block secure channels. (T5) says that although the intruder can choose to send a particular message to a particular principal, he cannot decide which transition this principal uses (if the message matches the LHS of more than one rule).

Note that if $\ell_V(v) = q'$ in a q-strategy tree $\mathcal{T}_q$, then there exists a path from q to q' in $\mathcal{G}_P$. By Remark 3.1.2 this implies that $\mathcal{T}_q$ has finite depth.

Let $C, C' \subseteq \mathcal{A} \cup \mathbb{K} \cup \mathcal{N}$. We say that a q-strategy tree $\mathcal{T}_q$ satisfies (C, C') if in every leaf of $\mathcal{T}_q$ all elements from C_i can be derived by the intruder and no element from C' can. Formally, for every leaf v of $\mathcal{T}_q$ with $\ell_V(v) = ((\Pi_1, \ldots, \Pi_n), \sigma, \mathcal{K}, \mathcal{S})$ it is required that $C \subseteq d(\mathcal{K})$ and $C' \cap d(\mathcal{K}) = \emptyset$ hold.

Definition 3.3.2 (strategy property) *A* strategy property *is a tuple*

$$\mathcal{C} = ((C_1, C'_1), \ldots, (C_l, C'_l)) \tag{3.1}$$

where $C_i, C'_i \subseteq \mathcal{A} \cup \mathbb{K} \cup \mathcal{N}$ for all $i \in \{1, \ldots, l\}$. A state $q \in \mathcal{G}_P$ satisfies *$\mathcal{C}$ if there exist q-strategy trees $\mathcal{T}_1, \ldots, \mathcal{T}_l$ such that every $\mathcal{T}_i$ satisfies (C_i, C'_i).*

3.3.2 Strategy Problems

We can now define the decision problem we are interested in:

Definition 3.3.3 (STRATEGY) *The decision problem* STRATEGY *asks, given a protocol P and a strategy property $\mathcal{C}$, whether there exists a state $q \in \mathcal{G}_P$ that satisfies $\mathcal{C}$. When this is the case we write $(P, \mathcal{C}) \in$* STRATEGY.

Note that in a q-strategy tree $\mathcal{T}_q$ there may exist vertices $v' \neq v$ with $\ell_V(v') = \ell_V(v)$ such that the subtrees $\mathcal{T}_q{\downarrow}v$ and $\mathcal{T}_q{\downarrow}v'$ of $\mathcal{T}_q$ rooted at v and v', respectively, are not isomorphic. In other words, the intruder's strategy may depend on the path that leads to a state (i.e., the history) rather than on the state alone, as is the case for positional strategies.

In general, positional strategies can be very restrictive. Here, however, we work in a framework where the "games" are of finite depth (bounded number of steps) and the winning conditions are reachability conditions. In such situations, positional strategies are no restriction, as we will show in what follows for our framework.

A q-strategy tree $\mathcal{T}_q$ is *positional* if different subtrees rooted at vertices with the same vertex label are isomorphic. (In particular, one could define positional strategies as subgraphs of $\mathcal{G}_P$.) We define p-Strategy analogously to Strategy, but with positional intruder strategies instead of arbitrary intruder strategies, and use similar notation.

The above assertion can now be stated formally:

Proposition 3.3.4 *For every instance* $(P, \mathcal{C})$ *of* Strategy *(and* p-Strategy*),*

$$(P, \mathcal{C}) \in \textsc{Strategy} \qquad \textit{iff} \qquad (P, \mathcal{C}) \in \textsc{p-Strategy}\ .$$

Proof It suffices to show that for every $q \in \mathcal{G}_P$ and $(C, C') \subseteq \mathcal{A} \cup \mathbb{K} \cup \mathcal{N}$ there exists a q-strategy tree which satisfies (C, C') iff there exists a positional q-strategy tree which satisfies (C, C'). The direction from right to left is trivial. For the other direction, let $\mathcal{T}_q$ be a q-strategy tree which satisfies (C, C'). If there exist $v' \neq v$ in $\mathcal{T}_q$ with $\ell_V(v) = \ell_V(v') = q'$, then it is not hard to see that the tree obtained by replacing $\mathcal{T}_q{\downarrow}v'$ with $\mathcal{T}_q{\downarrow}v$ (and consistently renaming the vertices) is still a q-strategy tree satisfying (C, C'). Now, starting with the subtrees of maximum depth and iteratively replacing all subtrees rooted at vertices labeled with the same state by one of the subtrees among them, we obtain the desired positional q-strategy tree. □

3.4 Constraint Solving

In this section, we briefly recall the notion of constraint system and state the well-known fact that procedures for solving constraint systems exist (see, e.g.,

[MS01] for more details). In Section 3.5, we will then use such a procedure as a black box in our constraint-based algorithm.

A *constraint* is of the form $t\colon T$ where t is a plain term and T is a finite non-empty set of plain terms. (Since we will take care of secure channel terms outside of constraint solving, we do not need to consider them here.) A sequence

$$\mathbf{C} = t_1\colon T_1, \ldots, t_n\colon T_n$$

of constraints is called a *constraint system.*

Given a constraint system $\mathbf{C}$ as above, one would like to determine all ground substitutions σ that satisfy $t_i\sigma \in d(T_i\sigma)$ for every $i \in \{1, \ldots, n\}$ and where $\mathrm{dom}(\sigma)$ contains only variables occurring in $\mathbf{C}$. These substitutions are called *solutions* of $\mathbf{C}$ and one writes $\sigma \vdash \mathbf{C}$ for such a substitution. Since there is, in general, an infinite number of such solutions, all solutions are typically described by a finite number of so-called simple constraint systems, which have obvious solutions, as will be explained in what follows.

For our purposes, a *simple constraint system* is of the form

$$\mathbf{C}' = x_1\colon T'_1, \ldots, x_m\colon T'_m$$

where the x_j's are pairwise distinct variables. Such a system has obvious solutions, namely, every substitution σ where every variable is assigned a different intruder atom from $\mathcal{A}_I$. Such a solution will be called *solution associated with* $\mathbf{C}'$.

Given a constraint system $\mathbf{C}$, a constraint solver produces a finite set U of pairs $(\mathbf{C}', \tau)$ where $\mathbf{C}'$ is simple and τ is a substitution such that

- no variable from $\mathbf{C}'$ belongs to $\mathrm{dom}(\tau)$, and
- for every $x \in \mathrm{dom}(\tau)$, every variable occurring in $\tau(x)$ also occurs in $\mathbf{C}'$.

The important property of U is that for every $(\mathbf{C}', \tau) \in U$ and for every solution σ' of $\mathbf{C}'$, the substitution $\sigma' \circ \tau$ is a solution of $\mathbf{C}$, in particular, this is true for every solution associated with $\mathbf{C}'$ as described above. This is why U is called a *sound set* for $\mathbf{C}$. It is said to be a *sound and complete set* for $\mathbf{C}$ if, in addition, every solution of $\mathbf{C}$ can be obtained in the described way.

If σ' is a solution associated with $\mathbf{C}'$ the solution $\sigma' \circ \tau$ will be called *complete solution associated with* $(\mathbf{C}', \tau)$.

If a given constraint system **C** satisfies certain conditions, which will be described below, sound and complete sets for **C** can be computed.

For a given constraint system **C** as above and a variable x in **C**, define $\mathsf{occ}(x) \in \{1, \ldots, n\}$ to be the minimal index such that $x \in \mathcal{V}(t_{\mathsf{occ}(x)})$, i.e., $\mathsf{occ}(x)$ is the index of the first constraint in the sequence where x occurs on the left-hand side of this constraint. If such an index does not exist, $\mathsf{occ}(x)$ is undefined.

We call a constraint system **C** as above *valid* if it satisfies the following properties for every $i \in \{1, \ldots, n\}$:

1. *Origination:* $\mathcal{V}(T_i) \subseteq \mathcal{V}(\{t_1, \ldots, t_{i-1}\})$, which means that $\mathsf{occ}(x)$ is defined and $\mathsf{occ}(x) < i$ for every $x \in \mathcal{V}(T_i)$.
2. *Monotonicity:*
   (a) $T_1 \subseteq T_i$.
   (b) For every $x \in \mathcal{V}(T_i)$ we have that $T_{\mathsf{occ}(x)} \subseteq T_i$.

Intuitively, origination corresponds to well-formedness of principals and monotonicity captures that the intruder does not forget information. Our definition of monotonicity above differs from the standard definition (see e.g., [MS01]). We chose to present the definition in the above form because it is simpler (but slightly stronger) than the standard definition and because it is sufficient for our application: the constraint systems we consider fulfill our stronger definition. We note, however, that we could also have used the more complex standard definition.

The following fact is well-known (see, e.g., [CV01, MS01, BMV03] and references therein):

Fact 3.4.1 *There exists a procedure which given a valid constraint system* ***C*** *outputs a sound and complete set for* ***C****.*

In our model we deal with an infinite set $\mathcal{A}_I$ of intruder atoms. Constraint solving procedures do not handle these kind of sets of atoms. It is easy to see that the set output by a constraint solving procedure for a given constraint system **C** also forms a sound and complete set for **C** when one considers the set $\mathcal{A}_I$ of intruder atoms.

While different constraint solving procedures (and implementations thereof) may compute different sound and complete sets, our constraint-based algorithm to be introduced in Section 3.5 works with any of them. It is only important that the set computed is sound and complete. So we fix one of these procedures for the remainder of this chapter and refer to it as *constraint solver*. As already mentioned in the introduction we only need one element of a sound and complete set at a time. We therefore view the constraint solver as a non-deterministic algorithm which at the end of each run on a given input returns an element from a (fixed) sound and complete set. We require that for every element in the sound and complete set, there exists a run of the constraint solver that returns it. If the sound and complete set is empty, the constraint solver always returns no.

We note that while standard constraint solving procedures can deal with the cryptographic primitives considered here, these procedures might need to be extended when adding further cryptographic primitives. For example, this is the case for private contract signatures, which are used in some contract signing protocols [GJM99]; it should be straightforward to add such signatures to constraint solving procedures.

3.5 The Constraint-Based Algorithm

We now present our constraint-based algorithm, called SolveStrategy, for deciding the problem Strategy. We first describe its main steps, with details given in subsequent sections. (In particular, the notions set in italics will be explained later.) The input to SolveStrategy is a protocol P and a strategy property $\mathcal{C} = ((C_1, C'_1), \ldots, (C_l, C'_l))$.

SolveStrategy$(P, \mathcal{C})$: $\{$yes, no$\}$

(A1) Guess a *symbolic branching structure* $\mathbf{B}$ for P and $\mathcal{C}$.

That is, guess a *symbolic path* π^s from the initial state of P to a *symbolic state* q^s and, for every $i \in \{1, \ldots, l\}$, a *symbolic* q^s*-strategy tree* $\mathcal{T}^s_{i,q^s}$ (see Section 3.5.1 for details).

(A2) From the symbolic branching structure $\mathbf{B}$ and the strategy property $\mathcal{C}$, derive the *induced valid constraint system* $\mathbf{C_B}$ (see Section 3.5.2 for the definition).

(A3) Run the constraint solver on $\mathbf{C_B}$. If it returns no, then halt. Otherwise, let $(\mathbf{C}', \tau)$ be the output returned by the solver and ν a complete solution associated with it.

(A4) Perform certain tests on $\mathbf{B}$ and ν to check whether ν when applied to $\mathbf{B}$ yields a valid path in $\mathcal{G}_P$ from the initial state of P to a state q and the q-strategy trees $\mathcal{T}_{i,q}$ satisfying (C_i, C_i') for every $i \in \{1, \ldots, l\}$ (see Section 3.5.3 for details of these tests).

(A5) If any of the tests failed, output no, and otherwise output yes.

In the following three sections, (A1), (A2), and (A4) will be further explained. Our main result is the following theorem, with the proof presented in Section 3.6:

Theorem 3.5.1 *Algorithm SolveStrategy is a (non-deterministic) decision procedure for* STRATEGY*.*

Corollary 3.5.2 STRATEGY *is decidable.*

We conclude this subsection with some remarks.

Remark 3.5.3 *In (A3), it is not sufficient to consider just one simple constraint system returned by the constraint solver.*

Remark 3.5.4 *(A4) is indispensable.*

The two previous remarks will be illustrated in Section 3.5.3.

Remark 3.5.5 *If yes is output, then $\boldsymbol{B}$ with ν applied is a solution of $(P, \mathcal{C})$.*

The previous remark will become clear from the proof of Theorem 3.5.1.

Remark 3.5.6 *SolveStrategy is phrased as a non-deterministic algorithm only for simplicity of presentation; one can easily turn it into a deterministic one, simply by successively checking all symbolic branching structures (of which there are only finitely many) and all simple constraint systems returned by a deterministic constraint solver.*

Remark 3.5.7 *SolveStrategy works with* any *constraint solving procedure.*

3.5.1 Guessing the Symbolic Branching Structure

To describe the first step of SolveStrategy in more detail, we first define symbolic branching structures, which consist of symbolic paths and symbolic strategy trees. To define symbolic paths and strategy trees, we need to introduce several other notions. These notions will be illustrated by the example in Figure 3.2.

One technical problem we have to overcome is that a symbolic strategy tree will represent a set of runs of protocols arranged in trees. Clearly, when two principal rules sharing variables are applied in independent branches of such a tree, the shared variables that have not been instantiated during the common history of the two branches need not be instantiated with the same terms. This makes it necessary to apply a renaming scheme for the variables involved: When x is a variable from the original protocol and u is a vertex of a symbolic tree (to be defined), we will use x_u to denote a new variable. We speak of *non-indexed* and *indexed* variables.

A *symbolic state*

$$q^s = ((\Pi_1, \ldots, \Pi_n), \mathcal{K}, \mathcal{S}, \mathcal{X}) \tag{3.2}$$

consists of an n-tuple $(\Pi_1, \ldots, \Pi_n)$ of rule trees, a finite set $\mathcal{K} \subseteq \mathcal{T}$ of terms (which may contain variables), a finite multi-set $\mathcal{S} \subseteq \mathcal{T}_{\mathsf{sc}}$ of secure channel terms (which may contain variables), and a finite set $\mathcal{X}$ of variables, which are called *used variables* and are used for renaming variables correctly, as explained in the previous paragraph. Obviously, a symbolic state is defined just as a concrete state except that the substitution is omitted, the intruder knowledge $\mathcal{K}$ and the secure channel $\mathcal{S}$ may contain terms instead of messages, and a set of reference variables has been added.

A *symbolic tree* is of the form $(V, E, r, \ell_V, \ell_E)$ with finite vertex and edge set V and E, respectively, root $r \in V$, vertex labeling function ℓ_V, which maps vertices to symbolic states, and edge labeling function ℓ_E, which maps edges to labels of symbolic transitions. Let $(u, u') \in E$ be an edge and assume

$$\begin{aligned}
\ell_V(u) &= ((\Pi_1, \ldots, \Pi_n), \mathcal{K}, \mathcal{S}, \mathcal{X}) \ , \quad \Pi_i = (V_i, E_i, r_i, \ell_i) \text{ for } i \in \{1, \ldots, n\} \ , \\
\ell_V(u') &= ((\Pi'_1, \ldots, \Pi'_n), \mathcal{K}', \mathcal{S}', \mathcal{X}') \ , \quad \Pi'_i = (V'_i, E'_i, r'_i, \ell'_i) \text{ for } i \in \{1, \ldots, n\} \ , \\
\ell_E((u, u')) &= \tau \ .
\end{aligned}$$

Then one the following conditions must be satisfied.

1. *Symbolic intruder transition,* $\tau = [i, v, I]$: $v \in V_i$ with $(r_i, v) \in E_i$ and $\ell_i(r_i, v) = R \Rightarrow S$ for some $i \in \{1, \ldots, n\}$, R and S such that
   (a) for every $j \neq i$ we have $\Pi'_j = \Pi_j$ and Π'_i is obtained from $\Pi_i{\downarrow}v$ by replacing every occurrence of a variable $x \in \mathcal{V}(R) \setminus \mathcal{X}$ by the new variable $x_{u'}$ (see above);
   (b) $\mathcal{K}' = \mathcal{K} \cup \{t'\}$ where either $t = S \notin \mathcal{T}_{\mathsf{sc}}$ or $S = \mathsf{sc}(n, n', t)$ for some $n, n' \in \mathcal{N}$, and t' is obtained from t by replacing every variable $x \in \mathcal{V}(R) \setminus \mathcal{X}$ by the new indexed variable $x_{u'}$;
   (c) either $\mathcal{S}' = \mathcal{S}$ and $S \notin \mathcal{T}_{\mathsf{sc}}$ or $\mathcal{S}' = \mathcal{S} \cup \{S'\}$, where S' is obtained from S by replacing every variable $x \in \mathcal{V}(R) \setminus \mathcal{X}$ by $x_{u'}$.
2. *Symbolic secure channel transition,* $\tau = [i, v, R', \mathsf{sc}]$: $v \in V_i$ with $(r_i, v) \in E_i$ and $\ell_i(r_i, v) = R \Rightarrow S$ such that $R' \in \mathcal{S}$, (a) and (b) from (1), and either $\mathcal{S}' = \mathcal{S} \setminus \{R'\}$ if $S \notin \mathcal{T}_{\mathsf{sc}}$ or $\mathcal{S}' = (\mathcal{S} \setminus \{R'\}) \cup \{S'\}$ otherwise.
3. *Symbolic ε-transition,* $\tau = [i, v]$: $v \in V_i$ with $(r_i, v) \in E_i$ and $\ell_i(r_i, v) = \varepsilon \Rightarrow S$ with (a)–(c) from (1).

In addition, $\mathcal{X}' = \mathcal{X} \cup \{x_{u'} \mid x \in \mathcal{V}(R) \setminus \mathcal{X}\}$.

Just as with (concrete) transitions, in a situation like above, we speak of a *symbolic transition*, we call i the *principal associated with the transition*, $R \Rightarrow S$ (for the intruder and secure channel transitions) and $\varepsilon \Rightarrow S$ (for the ε-transitions) the *principal rule associated with the transition*, and v the *principal vertex associated with the transition.* We call $\ell_V(u) \xrightarrow{\ell_E(u,u')} \ell_V(u')$ the *symbolic transition corresponding to (or associated with)* (u, u').

Let q^s be a symbolic state. A *symbolic q^s-tree* is a symbolic tree where the root is labeled q^s. Let $\mathcal{X}_{\mathcal{T}^s_{q^s}} = \bigcup_{v \in V} \mathcal{X}_v$ be the *set of used variables in $\mathcal{T}^s_{q^s}$* where $\mathcal{X}_v$ is the set of used variables in state $\ell_V(v)$.

Figure 3.4 depicts a symbolic q^s_0-tree $\mathcal{T}_{ex}$ for P_{ex} (Figure 3.2) where $q^s_0 = (\{\Pi_1, \Pi_2\},\ \mathcal{K}_0, \emptyset)$ is the symbolic initial state of P_{ex} and the set of used variables associated with q^s_0 is empty. Note that copies of the variables x and y have been introduced, namely $x_{h_3}, x_{h_7}, y_{h_4}, y_{h_8}$, following the aforementioned renaming scheme.

We can now define symbolic paths and symbolic strategy trees as special cases of symbolic trees.

Let $P = ((\Pi_1, \ldots, \Pi_n), \mathcal{K})$ be a protocol as usual. A *symbolic path* π^s

$$\begin{tikzcd}
\mathcal{T}_{ex} : & h_1\ ((f_1,g_1),\mathcal{K}_0,\emptyset,\emptyset) \arrow[dl, "{[2,g_2]}"'] \arrow[dr, "{[2,g_3]}"] & & \\
((f_1,g_2),\mathcal{K}_1,\{\mathsf{sc}(2,1,\langle a,b\rangle)\},\emptyset)\ h_2 \arrow[d, "{[1,f_2,\mathsf{sc}(2,1,\langle a,b\rangle),\mathsf{sc}]}"'] & & h_6\ ((f_1,g_3),\mathcal{K}_2,\{\mathsf{sc}(2,1,\langle b,b\rangle)\},\emptyset) \arrow[d, "{[1,f_2,\mathsf{sc}(2,1,\langle b,b\rangle),\mathsf{sc}]}"] & \\
((f_2,g_2),\mathcal{K}_1\cup\{x_{h_3}\},\emptyset,\{x_{h_3}\})\ h_3 \arrow[d, "{[1,f_3,I]}"'] & & h_7\ ((f_2,g_3),\mathcal{K}_2\cup\{x_{h_7}\},\emptyset,\{x_{h_7}\}) \arrow[d, "{[1,f_3,I]}"] & \\
((f_3,g_2),\mathcal{K}_1\cup\{x_{h_3},y_{h_4}\},\emptyset,\mathcal{X}_1)\ h_4 \arrow[d, "{[1,f_6,I]}"'] & & h_8\ ((f_3,g_3),\mathcal{K}_2\cup\{x_{h_7},y_{h_8}\},\emptyset,\mathcal{X}_2) \arrow[dl, "{[1,f_5,I]}"'] \arrow[dr, "{[1,f_6,I]}"] & \\
((f_6,g_2),\mathcal{K}_1\cup\{x_{h_3},y_{h_4},c_2\},\emptyset,\mathcal{X}_1)\ h_5 & h_9 & & h_{10} \\
 & ((f_5,g_3),\mathcal{K}_2\cup\{x_{h_7},y_{h_8},c_2\},\emptyset,\mathcal{X}_2) & & ((f_6,g_3),\mathcal{K}_2\cup\{x_{h_7},y_{h_8},c_2\},\emptyset,\mathcal{X}_2)
\end{tikzcd}$$

Figure 3.4: Symbolic strategy tree $\mathcal{T}_{ex}$ for the protocol P_{ex} with $\mathcal{K}_1 = \mathcal{K}_0 \cup \{\langle a,b\rangle\}$, $\mathcal{K}_2 = \mathcal{K}_0 \cup \{\langle b,b\rangle\}$, $\mathcal{X}_1 = \{x_{h_3}, y_{h_4}\}$, and $\mathcal{X}_2 = \{x_{h_7}, y_{h_8}\}$. Note that, for sake of presentation, the first component of the symbolic states associated to vertices of $\mathcal{T}_{ex}$ only contain the roots of rule trees rather than the whole rule trees. The strategy property we consider is $((C_{ex}, C'_{ex})) = ((\{c_2\},\{c_1\}))$.

of P is a symbolic q^s-tree where $q^s = ((\Pi_1,\ldots,\Pi_n),\mathcal{K},\emptyset,\emptyset)$ is the *symbolic initial state of* P and every vertex has at most one successor.

If q^s is a symbolic state, then a *symbolic* q^s*-strategy tree* is a symbolic q^s-tree $\mathcal{T}^s_{q^s} = (V,E,r,\ell_V,\ell_E)$ which satisfies the following conditions:

(S1) For every $v \in V$ and every symbolic ε-transition from $\ell_V(v)$ with label $[i,f]$, there exists v' with $(v,v') \in E$ such that $\ell_E(v,v') = [i,f]$.

(S2) Whenever $(v,v'),(v,v'') \in E$ with $v' \neq v''$, $\ell_E(v,v') = [j',f']$, and $\ell_E(v,v'') = [j'',f'']$, then $(j',f') \neq (j'',f'')$.

(S3) Whenever $(v,v'),(v,v'') \in E$ with $v' \neq v''$, $\ell_E(v,v') = [j',f',R',\mathsf{sc}]$, and $\ell_E(v,v') = [j'',f'',R'',\mathsf{sc}]$, then $(j',f',R') \neq (j'',f'',R'')$.

(S4) Whenever $(v,v'),(v,v'') \in E$ with $v' \neq v''$, $\ell_E(v,v') = [j',f',I]$, and $\ell_E(v,v'') = [j'',f'',I]$, then $j' = j''$ and $f' \neq f''$.

(S1) corresponds to (T3) and says that all symbolic ε-transitions that can be taken are present in a symbolic strategy tree. (S2) and (S3) make sure symbolic strategy trees cannot be too large: (S2) prevents superfluous symbolic ε-transitions to be present, while (S3) prevents superfluous secure channel transitions to be present. (S4) makes sure the intruder may only send one message to one principal when following his strategy.

There are certain properties required of strategy trees, for instance (T4) and (T5), which are not reflected in the definition of symbolic strategy trees. These properties are taken care of by the test in (A4) of SolveStrategy, which will be explained in Section 3.5.3.

The symbolic tree $\mathcal{T}_{ex}$ depicted in Figure 3.4 is in fact a symbolic q_0^s-strategy tree where $q_0^s = (\{\Pi_1, \Pi_2\}, \mathcal{K}_0, \emptyset, \emptyset)$ is the symbolic initial state of P_{ex}.

Given a protocol P, we call $\mathbf{B} = (\pi^s, \mathcal{T}_1^s, \ldots, \mathcal{T}_l^s)$ a *symbolic branching structure of* P if the following conditions are satisfied.

1. π^s is a symbolic path of P.
2. Let v be the leaf of π^s and $q^s = \ell_V(v)$. For each $i \in \{1, \ldots, l\}$, $\mathcal{T}_i^s$ is a symbolic q^s-strategy tree with root v.
3. For $i \in \{1, \ldots, l\}$, let V_i be the set of vertices of $\mathcal{T}_i^s$. For $i, j \in \{1, \ldots, l\}$ with $i \neq j$, $V_i \cap V_j = \{v\}$.

   Observe that, by our naming convention, this implies $\mathcal{X}_{\mathcal{T}_i^s} \cap \mathcal{X}_{\mathcal{T}_j^s} = \mathcal{X}_{\pi^s}$.

Obviously, there is a non-deterministic exponential time algorithm which given P can guess all possible symbolic branching structures (up to renaming of variables).

Consider our running example and the strategy property $((C_{ex}, C'_{ex})) = ((\{c_2\}, \{c_1\}))$. Then $\mathcal{T}_{ex}$ (Figure 3.4) can be viewed as a symbolic branching structure $\mathbf{B}_{ex}$ of P_{ex} when the path π^s is considered empty (and $l = 1$).

3.5.2 Constructing and Solving the Induced Constraint System

In this subsection, we explain how the constraint system $\mathbf{C_B}$ is derived from the symbolic branching structure $\mathbf{B} = (\pi^s, \mathcal{T}_1^s, \ldots, \mathcal{T}_l^s)$ computed in (A1) of SolveStrategy and the given strategy property $\mathcal{C}$, that is, we give the details for (A2).

The constraint system $\mathbf{C_B}$ can be shown to be valid, and hence, by Fact 3.4.1, a constraint solver can be used to solve it. In other words, (A3) is well-defined.

Illustrating the Construction by an Example

Before we get to technical details, we give some informal explanations and illustrate the construction with our running example, see Figure 3.2.

We first explain informally how to encode in a constraint system communication involving the secure channel. The basic idea is that we write messages intended for the secure channel into the intruder's knowledge and let the intruder deliver these messages. The problem is that while every message in the secure channel can only be read once, the intruder could try to deliver the same message several times. To prevent this, every such message when written into the intruder's knowledge is encrypted with a *new* key not known to the intruder and this key is also (and only) used in the principal rule which according to the symbolic branching structure is supposed to read the message. This guarantees that the intruder cannot deliver the same message several times to unintended recipients or make use of these encrypted messages in other contexts. Here we use the feasibility condition on principals introduced in Section 3.1.3, namely that verification keys can be derived by a principal. As explained before, without this condition, a principal rule of the form $\{y\}_x^s \Rightarrow x$ would be allowed even if the principal does not know (i.e., cannot derive) x. Such a rule would equip a principal with the unrealistic ability to derive any secret key from a ciphertext. Hence, the intruder, using this principal as an oracle, could achieve this as well and could potentially obtain the new keys used to encrypt messages intended for the secure channel.

We now turn to our example and explain how the (valid) constraint system, which we call $\mathbf{C}_{ex}$ derived from $\mathbf{B}_{ex}$ and $((C_{ex}, C'_{ex}))$ looks like and how it is derived from $\mathbf{B}_{ex}$, where $\mathbf{B}_{ex}$, as explained above, is simply the symbolic strategy tree $\mathcal{T}_{ex}$ (Figure 3.4): $\mathbf{C}_{ex}$ is the following sequence of constraints where $k_1, k_2 \in \mathcal{A}$ are new atoms and we write $t_1, \ldots, t_n$ instead of $\{t_1, \ldots, t_n\}$.

$$
\begin{array}{rrcl}
1. & \{\langle x_{h_3}, b\rangle\}^s_{k_1} & : & \mathcal{K}_1, \{\langle a, b\rangle\}^s_{k_1} \\
2. & \{\langle x_{h_7}, b\rangle\}^s_{k_2} & : & \mathcal{K}_2, \{\langle b, b\rangle\}^s_{k_2} \\
3. & \{y_{h_4}\}^s_k & : & \mathcal{K}_1, \{\langle a, b\rangle\}^s_{k_1}, x_{h_3} \\
4. & \{y_{h_8}\}^s_k & : & \mathcal{K}_2, \{\langle b, b\rangle\}^s_{k_2}, x_{h_7} \\
5. & y_{h_4} & : & \mathcal{K}_1, \{\langle a, b\rangle\}^s_{k_1}, x_{h_3}, y_{h_4} \\
6. & x_{h_7} & : & \mathcal{K}_2, \{\langle b, b\rangle\}^s_{k_2}, x_{h_7}, y_{h_8} \\
7. & y_{h_8} & : & \mathcal{K}_2, \{\langle b, b\rangle\}^s_{k_2}, x_{h_7}, y_{h_8} \\
8. & c_2 & : & \mathcal{K}_1, \{\langle a, b\rangle\}^s_{k_1}, x_{h_3}, y_{h_4}, c_2 \\
9. & c_2 & : & \mathcal{K}_2, \{\langle b, b\rangle\}^s_{k_2}, x_{h_7}, y_{h_8}, c_2 \\
10. & c_2 & : & \mathcal{K}_2, \{\langle b, b\rangle\}^s_{k_2}, x_{h_7}, y_{h_8}, c_2
\end{array}
$$

This constraint system is obtained from $\mathbf{B}_{ex}$ as follows: We traverse the vertices of $\mathbf{B}_{ex}$ in a top-down breadth first manner. Every edge induces a constraint except those edges which correspond to symbolic ε-transitions. This is how the constraints 1.–7. come about where 1., 3., and 5. are derived from the left branch of $\mathbf{B}_{ex}$ and 2., 4., 6., and 7. from the right branch. Note that in 1. and 2. we encode the communication with the secure channel by encrypting the terms with new keys k_1 and k_2. The terms $\{\langle a, b\rangle\}^s_{k_1}$ and $\{\langle b, b\rangle\}^s_{k_2}$ are not removed anymore from the right-hand side of the constraints, i.e., from the intruder knowledge in order for $\mathbf{C}_{ex}$ to satisfy the monotonicity property of constraint systems. As explained above, since we use *new* keys and due to the feasibility condition on principals, this does not cause problems. The constraints 8.–10. are used to ensure that c_2 can be derived at every leaf of $\mathcal{T}_{ex}$, a requirement that comes from our example security property $((C_{ex}, C'_{ex}))$ where $C_{ex} = \{c_2\}$. In vertex h_8 of $\mathcal{T}_{ex}$ two symbolic intruder transitions leave the vertex, which, as explained above, means that the associated principal rules should both be able to read the message delivered by the intruder.

Let $\mathbf{C}_1$ and $\mathbf{C}_2$ be constraint systems with empty sequences of constraints and the substitution $\nu_1 = \{x_{h_3} \mapsto a, x_{h_7} \mapsto b, y_{h_4} \mapsto a, y_{h_8} \mapsto b\}$ and $\nu_2 = \{x_{h_3} \mapsto a, x_{h_7} \mapsto b, y_{h_4} \mapsto b, y_{h_8} \mapsto b\}$, respectively. It is easy to see that $\{\mathbf{C}_1, \mathbf{C}_2\}$ is a sound and complete solution set for $\mathbf{C}_{ex}$. Since $\mathbf{C}_{ex}$ is valid, such a set can be computed by the constraint solver (Fact 3.4.1).

Formal Definition

We now provide a formal construction of the constraint system $\mathbf{C_B}$ from the symbolic branching structure $\mathbf{B} = (\pi^s, \mathcal{T}_1^s, \ldots, \mathcal{T}_l^s)$ and the given strategy property $\mathcal{C}$.

The construction consists of three steps: In the first two steps, $\mathbf{B}$ is transformed, and in the last step the resulting branching structure is turned into the constraint system $\mathbf{C_B}$. In the first step, $\mathbf{B}$ is transformed to get rid of secure channel terms generated by the intruder. In the second step, we encode communication with the secure channel as explained above.

Encoding the derivation of secure channel terms An intruder does not have secure channel terms in its initial knowledge and he does not receive secure channel terms during the run of a protocol. Hence, to construct a term $\mathsf{sc}(n, n', m)$ he has to derive n and m (not necessarily n'). Since addresses such as n may only occur in the first or second component of a secure channel term and the intruder cannot read these components, the intruder cannot derive new addresses. Thus, the only addresses the intruder knows are those in his initial knowledge. Therefore, the intruder can construct $\mathsf{sc}(n, n', m)$ only if n belongs to his initial knowledge and he can derive m. This motivates the following transformation:

(A2.1) For every edge e of $\mathbf{B}$ with which a symbolic intruder transition $q \xrightarrow{\tau} q'$ is associated and where the LHS of the associated principal rule is of the form $\mathsf{sc}(n, n', R)$, do the following:
If n belongs to the initial intruder knowledge, then replace the LHS of the associated principal rule simply by R, else stop and output no (for all of $\mathsf{SolveStrategy}$).

We refer to the symbolic branching structure obtained by the transformation just described as $\hat{\mathbf{B}}$. Strictly speaking, the resulting structure is not a symbolic branching structure anymore as it does not have the required form. By abuse of terminology, we still call it symbolic branching structure.

Encoding secure channel communication As already explained above, we eliminate the secure channel component in symbolic states and instead let the intruder handle all communication.

In what follows, with "new key" we mean a constant in $\mathcal{A}$ that does not occur anywhere else and which, in particular, the intruder does not (and will not get to) know.

(A2.2) For every vertex v of $\hat{\mathbf{B}}$ and every successor v' of v in $\hat{\mathbf{B}}$ do the following (we traverse the vertices of $\hat{\mathbf{B}}$ in a top-down breadth first manner starting with the root of $\hat{\mathbf{B}}$): If in the transition associated with the edge (v, v') of $\hat{\mathbf{B}}$ a message of the form $\mathsf{sc}(n, n', R)$ is written into the secure channel, then generate a new key $k_{v'}$ and write $\{R\}^s_{k_{v'}}$ into the intruder knowledge at v' and all successors of v' in $\hat{\mathbf{B}}$. Also, do the following for every edge (u, u') which is reachable from v' over edges not labelled $[\cdot, \cdot, \mathsf{sc}(n, n', R), \mathsf{sc}]$: Replace the label of (u, u') by $\{R\}^s_{k_{v'}}$ and the LHS of the principal rule associated with (u, u') by $\{R'\}^s_{k_{v'}}$ where R' is such that $\mathsf{sc}(n, n', R')$ is the LHS of the principal associated with (u, u').

We refer to the symbolic branching structure resulting from the transformation just described by $\overline{\mathbf{B}}$.

We call $\{R\}^s_{k_{v'}}$ the *(secure channel) encoding term associated with $k_{v'}$*. If a ground substitution is applied to this term, we call it the *(secure channel) encoding message associated with $k_{v'}$*. We call the keys introduced in the step described above *secure channel keys* (sch-keys) and denote the set of secure channel keys by K_{sc}. We refer to the key generated for the edge (v, v') by $k_{v'}$ (if a key was generated).

Deriving the constraint system From $\overline{\mathbf{B}}$ we now derive the constraint system $\mathbf{C}_{\mathbf{B}}$:

(A2.3) Let $\mathbf{C}_{\mathbf{B}}$ be the empty constraint system.

(A2.3.a) Traverse the vertices of $\overline{\mathbf{B}}$ in a top-down breadth first manner starting with the root of $\overline{\mathbf{B}}$ (and hence, the root of π^s). For every vertex v of $\overline{\mathbf{B}}$ and every successor v' of v do the following:
If the transition corresponding to the edge (v, v') is a symbolic secure channel or intruder transition, $\mathcal{K}$ is the intruder knowledge in the state associated with v, and R is the LHS of the principal rule associated with the transition, then add $R\colon \mathcal{K}$ to $\mathbf{C}_{\mathbf{B}}$ at the end.
We call constraints of this kind *intruder constraints of $\mathbf{C}_{\mathbf{B}}$*.

(A2.3.b) For every $i \in \{1, \dots l\}$, leaf v of $\mathcal{T}_i^s$, and $a \in C_i$ do the following: If $\mathcal{K}$ is the intruder's knowledge in the state associated with v, then add $a\colon \mathcal{K}$ to $\mathbf{C_B}$ at the end.
We call constraints of this kind *strategy property constraints of* $\boldsymbol{C_B}$.

This completes the description of (A2) of SolveStrategy.

To be able to apply Fact 3.4.1, we need the following lemma. The proof of this lemma is straightforward and therefore omitted: To show monotonicity one has to make use of the fact that secure channel encoding messages are not removed anymore from the intruder's knowledge

Lemma 3.5.8 $\boldsymbol{C_B}$ *is a valid constraint system.*

As a consequence of this lemma and Fact 3.4.1, we can employ the constraint solver to solve $\mathbf{C_B}$.

3.5.3 Checking the Induced Substitutions

Let $\mathbf{B} = (\pi^s, \mathcal{T}_1^s, \dots, \mathcal{T}_l^s)$ be the symbolic branching structure obtained in (A1) of SolveStrategy and let $(\mathbf{C}', \tau)$ be the output returned by the constraint solver when applied to $\mathbf{C_B}$ in (A2). Let ν be the complete solution associated with $(\mathbf{C}', \tau)$ in (A3), see Section 3.4. We emphasize that for our algorithm to work, it is important that ν replaces the variables in $\mathbf{C}'$ by *new* intruder atoms from $\mathcal{A}_I$ not occurring in $\mathbf{B}$.

In (A4), we want to check whether when applying ν to $\mathbf{B}$, which yields $\nu(\mathbf{B}) = (\nu(\pi^s), \nu(\mathcal{T}_1^s), \dots, \nu(\mathcal{T}_l^s))$, we obtain a solution of the problem instance $(P, \mathcal{C})$. Hence, we need to check whether i) $\nu(\pi^s)$ corresponds to a path in $\mathcal{G}_P$ from the initial state of $\mathcal{G}_P$ to some state in $\mathcal{G}_P$, say q, and ii) $\nu(\mathcal{T}_i^s)$ corresponds to a q-strategy tree for (C_i, C_i') for every $i \in \{1, \dots, l\}$.

Since ν is a complete solution of $\mathbf{C_B}$, some of these conditions are satisfied by construction. In particular, $\nu(\pi^s)$ is guaranteed to be a path in $\mathcal{G}_P$ starting from the initial state. Also, (T1)–(T3), do not need to be checked. Moreover, we know that $\nu(\mathcal{T}_i^s)$ satisfies $(C_i, \emptyset)$. Hence, SolveStrategy only needs to make sure that $\nu(\mathcal{T}_i^s)$ fulfills $(\emptyset, C_i')$ and that (T4) and (T5) are satisfied for every tree $\nu(\mathcal{T}_i^s)$.

This leads to the following tests:

(A4) For every $i \in \{1, \ldots, l\}$ do:

(A4.1) *Check the strategy property:* For every leaf v of $\mathcal{T}_i^s$ check that $C_i' \cap d(\mathcal{K}\nu) = \emptyset$ where $\mathcal{K}$ is the intruder knowledge in the symbolic state associated with v.

(A4.2) For every vertex v in $\mathcal{T}_i^s$ perform the following tests. Let $(\{\Pi_h'\}_h, \mathcal{K}, \mathcal{S}, \mathcal{X})$ be the symbolic state associated with v in $\mathcal{T}_i^s$ and for every $h \in \{1, \ldots, n\}$ let $\Pi_h' = (V_h', E_h', r_h', \ell_h')$.

(A4.2.a) *Check (T4):* For all $m \in \mathcal{S}\nu$, $h \in \{1, \ldots, n\}$, $n_h' \in V_h'$, and R such that

A. $(r_h', n_h') \in E_h'$,

B. R is the LHS of $\ell_h'(r_h', n_h')$, and

C. $R\nu$ matches m,

check that there exists a successor v' of v in $\mathcal{T}_i^s$ such that (v, v') is labelled $[h, n_h', R', \mathsf{sc}]$ for some R' with $R'\nu = m$.

(A4.2.b) *Check (T5):* For every successor v' of v in $\mathcal{T}_i^s$, $h \in \{1, \ldots, n\}$, $n_h', n_h'' \in V_h'$ with $n_h' \neq n_h''$, R, and R' such that

A. $[h, n_h', I]$ is the label of (v, v'),

B. R is the LHS of $\ell_h'(r_h', n_h')$,

C. $(r_h', n_h'') \in E_h'$,

D. R' is the LHS of $\ell_h'(r_h', n_h'')$, and

E. $R'\nu$ matches with $\hat{R}\nu$ where $\hat{R}$ is obtained from R by replacing each non-indexed variable x in R by $x_{v'}$,

check that there exists a successor v'' of v in $\mathcal{T}_i^s$ such that (v, v'') is labelled $[h, n_h'', I]$.

It is clear that the tests in (A4.2) can easily be performed given ν and $\mathcal{T}_i^s$. In [CKRT03], it was shown that the derivation problem, i.e., the problem of deciding $m \in d(\mathcal{K})$ given a ground term m and a finite set of ground terms $\mathcal{K}$, can be decided in polynomial time. Hence, (A4.1) can also be checked efficiently in the size of the security property and $\mathcal{K}\nu$.

If the above checks are successful, we say that ν is *valid* for $\mathbf{B}$. In this case, SolveStrategy outputs yes.

In our example, the induced substitution for $\mathbf{C}_i$ is ν_i as $\mathbf{C}_i$ does not contain any variables. It can easily be verified that with $\mathbf{C}' = \mathbf{C}_2$ and the induced substitution ν_2, the above checks are all successful. However, (A4.2.b) fails for $\mathbf{C}' = \mathbf{C}_1$ and ν_1 because in h_4 the rule $a \Rightarrow c_1$ could also be applied but it is not present in $\mathbf{B}_{ex}$. In fact, $\mathbf{B}_{ex}\nu_1$ would not yield a solution of the instance $(P_{ex}, ((C_{ex}, C'_{ex})))$. This example illustrates that in SolveStrategy one cannot dispense with the last step, namely checking the substitutions, and that one has to try the different pairs $(\mathbf{C}', \tau)$ in the sound and complete solution set for $\mathbf{C}_\mathbf{B}$.

3.6 Proof of the Main Theorem

In this section, we prove Theorem 3.5.1. Obviously, SolveStrategy always terminates; we only need to prove soundness and completeness. This is done in Section 3.6.2 and 3.6.3, respectively. We start with some additional notation that will be used in the proof.

For all of this section, we fix a protocol P and a strategy property $\mathcal{C}$ as usual.

3.6.1 Further Terminology

Concrete Branching Structures and Naming Conventions

Let $q \in \mathcal{G}_P$. A *q-tree* is a q-strategy tree where, however, only (T1) and (T2) need to be satisfied (see Definition 3.3.1). A *concrete path* π for P is a q_0-tree where q_0 is the initial state of P and every vertex in π has at most one successor. A *concrete branching structure* for P is defined analogously to *symbolic branching structure* only that now the symbolic path is a concrete path π and the symbolic strategy trees $\mathcal{T}_i$ are concrete strategy trees for every $i \in \{1, \ldots, l\}$. Such a branching structure satisfies $\mathcal{C}$ if $\mathcal{T}_i$ satisfies (C_i, C'_i) for every $i \in \{1, \ldots, l\}$.

Remark 3.6.1 $(P, \mathcal{C}) \in$ STRATEGY *iff there exists a concrete branching structure for P which satisfies $\mathcal{C}$.*

Concrete branching structures for P will be denoted $\mathbf{B} = (\pi, \mathcal{T}_1, \ldots, \mathcal{T}_l)$ where $\pi = (V^\pi, E^\pi, r^\pi, \ell_V^\pi, \ell_E^\pi)$ and $\mathcal{T}_i = (V^i, E^i, r^i, \ell_V^i, \ell_E^i)$ for every $i \in \{1, \ldots, l\}$.

Let $\mathbf{B}$ be a concrete branching structure for P. Note that $\mathbf{B}$ can be viewed as a q_0-tree $(V, E, r, \ell_V, \ell_E)$ where $V = V^\pi \cup V^1 \cup \cdots \cup V^l$, $E = E^\pi \cup E^1 \cup \cdots \cup E^l$, $r = r^\pi$, and ℓ_V and ℓ_E coincide with the labeling functions of the components of $\mathbf{B}$ on the respective domains.

For a symbolic or concrete tree $\mathcal{T}$, we write $v \in \mathcal{T}$ to express that v is a vertex in $\mathcal{T}$. Analogously, we write $(v, v') \in \mathcal{T}$ if (v, v') is an edge in $\mathcal{T}$. Note that since branching structures are viewed as trees we may also write $v \in \mathbf{B}$ if $v \in V$ and $(v, v') \in \mathbf{B}$ if $(v, v') \in E$. For $v \in \mathbf{B}$ let $\ell_V(v) = ((\Pi_1^v, \ldots, \Pi_n^v), \sigma^v, \mathcal{K}^v, \mathcal{S}^v)$ and $\Pi_j^v = (V_j^v, E_j^v, r_j^v, \ell_j^v)$.

As usual, for an edge (v, v') in $\mathbf{B}$, we talk about the transition (in $\mathcal{G}_P$) associated with (v, v'). When i is the principal associated with this transition, we call i the principal associated with (v, v'), respectively. Similarly, we speak of the principal rule and the principal vertex associated with (v, v').

Our notation and naming conventions for symbolic branching structures will be analogous; we will, however, always add an "s" as a superscript. That is, a symbolic branching structure will be denoted by $\mathbf{B}^s = (\pi^s, \mathcal{T}_1^s, \ldots, \mathcal{T}_l^s)$, we will write $\pi^s = (V^{s,\pi}, E^{s,\pi}, r^{s,\pi}, \ell_V^{s,\pi}, \ell_E^{s,\pi})$ and $\Pi_j^{s,v} = (V_j^{s,v}, E_j^{s,v}, r_j^{s,v}, \ell_j^{s,v})$.

Recall that for a symbolic branching structure $\mathbf{B}^s$ we denote the branching structure constructed from $\mathbf{B}^s$ in (A2) of SolveStrategy by $\overline{\mathbf{B}}^s$. We refer to the components of $\overline{\mathbf{B}}^s$ by appropriate decorations of the components of $\mathbf{B}^s$, i.e., $\overline{\ell}_E^{s,\pi}$, $\overline{V}_j^{s,v}$ and so on.

Substitutions of Symbolic States and Trees

Let $q^s = ((\Pi_1, \ldots, \Pi_n), \mathcal{K}, \mathcal{S}, \mathcal{X})$ be a symbolic state reachable from the initial symbolic state of P. Recall from Section 3.5.1 that the variables in $\mathcal{X}$ are indexed with vertices, that is, they are of the form x_v. Consider the mapping $\delta \colon x_v \mapsto x$ which drops the respective index. Since q^s is reachable from the initial symbolic state of P, the function δ is one-to-one.

Let ν be a substitution of the variables in $\mathcal{X}$. We denote by $\nu(q^s)$ the (concrete) state obtained from q^s by substituting the variables in q^s according to ν. More precisely, let $\mathcal{X}'$ be the image of δ. Then $\delta \colon \mathcal{X} \to \mathcal{X}'$ is a bijection. We define $\nu(q^s)$ to be the tuple $\nu(q^s) = ((\Pi_1', \ldots, \Pi_n'), \nu'|_{\mathcal{X}'}, \mathcal{K}\nu, \mathcal{S}\nu)$ where Π_i' is obtained from Π_i by dropping the indices of variables in $\mathcal{X}$ and $\nu'|_{\mathcal{X}'}$ is a substitution with domain $\mathcal{X}'$ and $\nu'|_{\mathcal{X}'}(x) = \nu(\delta^{-1}(x))$ for every $x \in \mathcal{X}'$.

For a symbolic q^s-tree $\mathcal{T} = (V, E, r, \ell_V, \ell_E)$ and a substitution ν of the variables used in $\mathcal{T}$ we denote by $\nu(\mathcal{T})$ the instantiation obtained from $\mathcal{T}$ by substituting the variables in $\mathcal{T}$ according to ν. More precisely, we define $\nu(\mathcal{T})$ in such a way that it has the form of a (concrete) $\nu(q^s)$-tree. For this, we need to adjust the format of the labeling of edges and of course replace the symbolic states by concrete ones. However, we note that $\nu(\mathcal{T})$ is not necessarily a $\nu(q^s)$-tree in the sense defined above since $\nu(q^s)$ may not be a state of $\mathcal{G}_P$ and the transitions associated to the edges of this tree may not be transitions of $\mathcal{G}_P$. Formally, $\nu(\mathcal{T})$ is defined to be the the tuple $\nu(\mathcal{T}) = (V, E, r, \nu(\ell_V), \nu(\ell_E))$ with $\nu(\ell_V)(v) = \nu(\ell_V(v))$ and

$$\nu(\ell_E)(v,v') = \begin{cases} [j] & \text{if } \ell_E(v,v') = [j,f] \\ [j,\nu(R),I] & \text{if } \ell_E(v,v') = [j,f,I] \text{ and } \ell_j(r_j,f) = [R \Rightarrow S] \\ [j,\nu(R),\mathsf{sc}] & \text{if } \ell_E(v,v') = [j,f,R,\mathsf{sc}] \end{cases}$$

for every $(v,v') \in E$ where r_j denotes the root of the jth principal Π_j in the symbolic state $\ell_V(v)$ and ℓ_j denotes the labeling function of Π_j. Note that $\nu(\mathcal{T})$ is in fact a concrete $\nu(q^s)$-tree.

Solutions of Symbolic Branching Structures

For a symbolic branching structure $\mathbf{B}^s = (\pi^s, \mathcal{T}_1^s, \dots, \mathcal{T}_l^s)$ and a substitution ν, let $\nu(\mathbf{B}^s) = (\nu(\pi^s), \nu(\mathcal{T}_1^s), \dots, \nu(\mathcal{T}_l^s))$.

We say that ν *solves* $\boldsymbol{B}^s$ *(with respect to the strategy property* $\mathcal{C}$*)* if $\nu(\mathbf{B}^s)$ is a concrete branching structure (which satisfies $\mathcal{C}$).

A *root path* π' in $\mathbf{B}$ is a sequence $\pi' = v_1, \dots, v_n$ of vertices $v_i \in \mathbf{B}$ such that $(v_i, v_{i+1}) \in \mathbf{B}$ and v_1 is the root of $\mathbf{B}$ (and hence, of π). If $\rho_i = R_i \Rightarrow S_i$ denotes the principal rule associated with the edge $(v_i, v_{i+1}) \in \mathbf{B}$, then $\rho = \rho_{\pi'} = \rho_1, \dots, \rho_{n-1}$ is called the *pr-sequence associated with* π'. Given a set $\mathcal{K}$ of messages we call a ground substitution σ of the variables occurring in ρ, a *solution of* ρ *w.r.t.* $\mathcal{K}$ if $R_i\sigma \in d(\mathcal{K} \cup \{S_j\sigma \mid j < i\})$ for every i. We call σ a *solution of* π' *w.r.t.* $\mathcal{K}$ if σ is a solution of $\rho_{\pi'}$ w.r.t. $\mathcal{K}$.

3.6.2 Soundness of SolveStrategy

In this section, we show soundness of SolveStrategy. The main problem is that when translating the symbolic branching structure that was guessed in (A1) of SolveStrategy into a constraint system we model the communication via secure channels by introducing what we call secure channel encoding terms. These encoding terms are written into the knowledge of the intruder and the idea is that the intruder delivers these terms instead of the secure channel terms that would have been delivered by the secure channel in the model. The problem is that the intruder might use these messages also in other contexts. We need to show that one can eliminate these secure channel encoding messages from a solution of the constructed constraint system.

We start with some lemmas on substitution, especially in the context of encryption.

Preliminaries

For terms t, t', t'', we denote by $t_{|t' \to t''}$ the term obtained from t by simultaneously replacing every occurrence of t' by t''. This definition is extended to sets of terms and substitutions in the obvious way. We write $E_{|t' \to t''}$ and $\sigma_{|t' \to t''}$, respectively.

The following lemma can easily be shown by induction on the length of derivations using the same techniques as for example in [RT03].

Lemma 3.6.2 *Let E be a set of messages and m_1, m_2, m_3, m_4 messages with $m_1 = \{m_2\}^s_{m_3}$. If $m_1 \in E$ and $m_3 \notin d(E)$, then*

$$d(E)_{|m_1 \to m_4} \subseteq d(E_{|m_1 \to m_4}) \ . \tag{3.3}$$

Terms can be viewed as finite vertex-labeled ordered trees where the vertices are labeled with function symbols and the number of successors of a vertex is exactly the arity of the function symbol the vertex is labelled with. By convention, if a vertex is labeled by the encryption symbol $\{\cdot\}^s_\cdot$, then the right successor corresponds to the key. Similarly, for the $\{\cdot\}^a_\cdot$ and $\mathsf{sig}(\cdot, \cdot)$ symbols. For a term t, we write $v \in t$ to say that v is a vertex of t and $t{\downarrow}v$ to denote the subterm of t rooted at vertex v of t.

We say that a term t *only occurs in the context of* $\{t'\}^s_t$ *in the term* t'' if for all $v \in t''$ with $t''{\downarrow}v = t$ we have that $t''{\downarrow}v' = \{t'\}^s_t$ where v' is the predecessor of v in t''. For a set of terms E we say that t *only occurs in the context of* $\{t'\}^s_t$ *in* E if t only occurs in the context of $\{t'\}^s_t$ in all terms $t'' \in E$. Let σ be a substitution. We say that t *only occurs in the context of* $\{t'\}^s_t$ *in* σ if t only occurs in the context of $\{t'\}^s_t$ in $\sigma(x)$ for all $x \in \text{dom}(\sigma)$.

Lemma 3.6.3 *Let R be a term, m_1, m_2 be messages, $k \in \mathcal{A}$ be an atom, and σ be a ground substitution such that $k \notin Sub(R)$ and k only occurs in the context of $\{m_1\}^s_k$ in σ. Then*

$$(R\sigma)_{|\{m_1\}^s_k \to m_2} = R(\sigma_{|\{m_1\}^s_k \to m_2}) \ . \tag{3.4}$$

Proof First observe that there does not exist a subterm t of R such that t is not a variable and $t\sigma = \{m_1\}^s_k$. Otherwise, since $k \notin \text{Sub}(R)$, t would be of the form $\{t'\}^s_x$ for some t' and variable x, and hence, $\sigma(x) = k$ in contradiction to the assumption that k only occurs in the context of $\{m_1\}^s_k$ in σ. From this the claim of the lemma follows easily. □

The following lemma can be proved by a simple induction.

Lemma 3.6.4 *Let E be a set of messages and k, m be messages. If k only occurs in the context of $\{m\}^s_k$ in E, then k only occurs in the context of $\{m\}^s_k$ in $d(E)$.*

A vertex v of a term is a *key node* if it is a descendant of the right successor (inclusive) of a vertex labeled with $\{\cdot\}^s_\cdot$, $\{\cdot\}^a_\cdot$, or $\mathsf{sig}(\cdot,\cdot)$. A term t *occurs only in a key position* in term t' if every vertex v of t' such that the term corresponding to v is t is a key node. Otherwise, we say that t *occurs in a non-key position* in t'. For example, x only occurs in a key position in the term $\{\langle a, b\rangle\}^s_{\langle c,x\rangle}$ but not in the term $\{\langle x, b\rangle\}^s_c$.

From the above lemmas, we obtain:

Lemma 3.6.5 *Let $k_1, \ldots, k_n \in \mathcal{A}$ be pairwise distinct atoms, let $t_1, \ldots, t_n$ be terms, and let $\mathcal{K}_1$ and $\mathcal{K}_2$ be sets of terms such that $\{k_1, \ldots, k_n\} \cap Sub(\mathcal{K}_1) = \emptyset$, $\mathcal{K}_2 = \{\{t_1\}^s_{k_1}, \ldots, \{t_n\}^s_{k_n}\}$, and $\{k_1, \ldots, k_n\} \cap Sub(t_i) = \emptyset$ for $i \in \{1, \ldots, n\}$. Let σ be a ground substitution such that $\{k_1, \ldots, k_n\} \cap Sub(\sigma(x)) = \emptyset$ for every variable x. Then the following is true for every term R with $\{k_1, \ldots, k_n\} \cap Sub(R) = \emptyset$.*

1. *$R\sigma \in d(\mathcal{K}_1\sigma \cup \mathcal{K}_2\sigma)$ implies $R\sigma \in d(\mathcal{K}_1\sigma)$.*
2. *For every i, $\{R\}^s_{k_i}\sigma \in d(\mathcal{K}_1\sigma \cup \mathcal{K}_2\sigma)$ implies $R\sigma = t_i\sigma$.*

Proof To show (1), we first observe that $k_i \notin d(\mathcal{K}_1\sigma \cup \mathcal{K}_2\sigma)$. Now, iteratively applying Lemma 4.7.27 for every i with $m_1 = \{t_i\}^s_{k_i}\sigma$ and some intruder atom m_4 allows us to eliminate the messages in $\mathcal{K}_2\sigma$. (2) easily follows with Lemma 3.6.4. □

Proof of Soundness

Assume SolveStrategy outputs yes. We have to show $(P, \mathcal{C}) \in \textsc{Strategy}$.

Let $\mathbf{B}^s = (\pi^s, \mathcal{T}_1^s, \ldots, \mathcal{T}_l^s)$ be the symbolic branching structure guessed in (A1) of SolveStrategy. Let $\mathbf{C} = \mathbf{C}_{\mathbf{B}^s}$ be the corresponding constraint system constructed in (A2) and let $\mathbf{C}'$ be the simple constraint system returned by the constraint solver in (A3) such that the complete solution ν associated with $\mathbf{C}'$ passes the tests in (A4) of SolveStrategy.

We would like to show that ν solves $\mathbf{B}^s$ with respect to $\mathcal{C}$. Recall that this means that $\nu(\mathbf{B}^s)$ is a concrete branching structure which satisfies $\mathcal{C}$. The main problem is that ν might contain secure channel keys (see Section 3.5.2). With these keys, ν cannot solve $\mathbf{B}^s$ since they do not occur in $\mathbf{B}^s$. We will therefore iteratively eliminate these keys from ν. This will be done in such a way that before and after every step the current substitution satisfies certain conditions with respect to $\overline{\mathbf{B}}^s$ and $\mathbf{C}$ (recall the definition of $\overline{\mathbf{B}}^s$ and $\mathbf{C}$ from Section 3.5.2). At the end, we will have a ground substitution μ for which we then show that it solves $\mathbf{B}^s$ with respect to $\mathcal{C}$.

For a given substitution σ, the conditions are:

(P1) For every root path π of $\overline{\mathbf{B}}^s$ the substitution σ is a solution for π.

(P2) The substitution σ solves the strategy property constraints of $\mathbf{C}$.

(P3) The substitution σ passes the tests in (A4).

We first prove:

Lemma 3.6.6 *Let σ be a substitution satisfying (P1)–(P3) and let $k \in K_{\mathsf{SC}}$ be a secure channel key and $\{t\}_k^s$ be the corresponding encoding term.*

Then k only occurs in the context of $\{t\sigma\}_k^s$ in σ.

Proof Assume that there exists a variable z in the domain of σ such that k does not only occur in the context of $\{t\sigma\}_k^s$ in $\sigma(z)$. Note that by construction, z is of the form x_v where v is a vertex in $\mathbf{B}^s$ and x is a variable in P. Let z be minimal with this property. That is, there exists $v \in \overline{\mathbf{B}}^s$ such that $z \in \mathcal{X}_v$ (the set of used variables at vertex v) and for every proper ancestor $v' \in \overline{\mathbf{B}}^s$ of v with $z \notin \mathcal{X}_{v'}$ and every $z' \in \mathcal{X}_{v'}$ it holds that k only occurs in the context of $\{t\sigma\}_k^s$ in $\sigma(z')$. Let z and v be as above and let $\pi' = v_1, \ldots, v_n$ with $v_n = v$ be a root path in $\overline{\mathbf{B}}^s$ and $\rho = \rho_{\pi'}^{\overline{\mathbf{B}}} = \rho_1, \ldots, \rho_{n-1}$ be the pr-sequence associated with π' where $\rho_i = R_i \Rightarrow S_i$. Since σ satisfies condition (P1), we know that $R_n\sigma \in d(\overline{\mathcal{K}}^{s,v_{n-1}}\sigma)$ (recall the definition of $\overline{\mathcal{K}}^{s,v_{n-1}}$ from above). Clearly, k only occurs in the context of $\{t\sigma\}_k^s$ in $\overline{\mathcal{K}}^{s,v_{n-1}}$. Also, $\mathcal{V}(\overline{\mathcal{K}}^{s,v_{n-1}}) \subseteq \mathcal{X}_{v_{n-1}}$. And hence, k only occurs in the context of $\{t\sigma\}_k^s$ for every $z' \in \mathcal{V}(\overline{\mathcal{K}}^{s,v_{n-1}})$. Hence, k only occurs in the context of $\{t\sigma\}_k^s$ in $\overline{\mathcal{K}}^{s,v_{n-1}}\sigma$. Lemma 3.6.4 thus yields:

(*) k only occurs in the context of $\{t\sigma\}_k^s$ in $R_n\sigma$.

However, we also have that $z \in \mathcal{V}(R_n)$ and that k does not only occur in the context of $\{t\sigma\}_k^s$ in $\sigma(z)$. From this it follows that there exists a subterm of R_n of the form $\{t\}_z^s$ for some term t and that $\sigma(z) = k$. By the feasibility condition on principals we know that $z \in d(\mathcal{K} \cup \{R_1, \ldots, R_n\})$ where $\mathcal{K}$ is the union of the initial knowledge of the principals in P and we consider variables as constants. Because of the minimality, we know that z only occurs in R_n and it follows that z must occur in a non-key position in R_n. Hence, k also occurs in a non-key position in $R_n\sigma$, in contradiction to (*). □

Using Lemma 3.6.6, we obtain:

Lemma 3.6.7 *Let σ be a substitution satisfying (P1)–(P3) and let $k \in K_{\mathsf{SC}}$ be a secure channel key and $\{t\}_k^s$ be the corresponding encoding term.*

Let $\pi' = v_1, \ldots, v_n$ be a root path of $\overline{\boldsymbol{B}}^s$ and $\rho = \rho_1, \ldots, \rho_{n-1}$ with $\rho_i = R_i \Rightarrow S_i$ the pr-sequence associated with π'. In addition, let $\mathcal{K}_{v_i} = \overline{\mathcal{K}}^{s,v_i}$ be as in the proof of Lemma 3.6.6.

Then k only occurs in the context of $\{t\sigma\}_k^s$ in $\mathcal{K}_{v_i}\sigma$ and $R_i\sigma$ for every i.

Proof By the construction of $\overline{\mathbf{B}}^s$, we know that k only occurs in the context of $\{t\sigma\}_k^s$ in $\mathcal{K}_{v_i}$ and R_i for every i. From this together with Lemma 3.6.6, the claim follows immediately. □

We can now state:

Proposition 3.6.8 *Let σ be a substitution satisfying (P1)–(P3) and let $k \in K_{\mathsf{SC}}$ be a secure channel key and $\{t\}_k^s$ be the corresponding encoding term.*

Let $a \in \mathcal{A}_I$ be a new intruder atom and define $\sigma' = \sigma_{|m \to a}$ where $m = \{t\sigma\}_k^s$.

Then σ' satisfies (P1)–(P3).

Proof First note that due to Lemma 3.6.6 atom k does not occur in σ' anymore. We have to show that σ' satisfies (P1)–(P3).

(P1) Let $\pi' = v_1, \ldots, v_n$, ρ, ρ_i, R_i, S_i, and $\mathcal{K}_{v_i}$ be given as in Lemma 3.6.7. By assumption the substitution σ satisfies (P1), and hence, σ solves π'. That is, $R_i\sigma \in d(\mathcal{K}_{v_{i-1}}\sigma)$ for every i. We want to show that $R_i\sigma' \in d(\mathcal{K}_{v_{i-1}}\sigma')$ for every i. We proceed with a case distinction.

If k does not occur in $R_i\sigma$ and $\mathcal{K}_{v_{i-1}}\sigma$, then $R_i\sigma = R_i\sigma'$ and $\mathcal{K}_{v_{i-1}}\sigma = \mathcal{K}_{v_{i-1}}\sigma'$, and hence, $R_i\sigma' \in d(\mathcal{K}_{v_{i-1}}\sigma')$.

Otherwise, by construction of $\overline{\mathbf{B}}^s$, we have that $\{t\}^s_k \in \mathcal{K}_{v_{i-1}}$, and hence, $m \in \mathcal{K}_{v_{i-1}}\sigma$. By Lemma 3.6.7, we know:

(*) k only occurs in the context of $\{t\sigma\}^s_k$ in $\mathcal{K}_{v_i}\sigma$ and $R_i\sigma$.

Now, if $k \in \mathrm{Sub}(R_i)$, then by construction of $\overline{\mathbf{B}}^s$, we know that R_i is of the form $\{t'\}^s_k$ for some term t'. It follows that $R_i\sigma = m$. Obviously, we have that $m = \{t\}^s_k\sigma = \{t\}^s_k\sigma'$. Thus, $m \in \mathcal{K}_{v_{i-1}}\sigma'$, and therefore, $R_i\sigma' \in d(\mathcal{K}_{v_{i-1}}\sigma')$.

If $k \notin \mathrm{Sub}(R_i)$, we can apply Lemma 3.6.3 and obtain that $(R_i\sigma)_{|m\to a} = R_i\sigma'$. Moreover, using Lemma 3.6.4 and (*), we know that $k \notin d(\mathcal{K}_{v_{i-1}}\sigma)$. Now, we can apply Lemma 4.7.27 and obtain that $R_i\sigma' = (R_i\sigma)_{|m\to a} \in d((\mathcal{K}_{v_{i-1}}\sigma)_{|m\to a})$. For $S \in \mathcal{K}_{v_{i-1}}$ such that $k \notin S$, Lemma 3.6.3 implies $S\sigma' = (S\sigma)_{|m\to a}$. By construction of $\overline{\mathbf{B}}^s$, if $k \in \mathrm{Sub}(S)$, then $S = \{t\}^s_k$, and hence, $S\sigma = m$. Since $a \in \mathcal{A}_I \subseteq d(\mathcal{K}_{v_{i-1}}\sigma')$, we obtain that $d((\mathcal{K}_{v_{i-1}}\sigma)_{|m\to a}) \subseteq d(\mathcal{K}_{v_{i-1}}\sigma')$. Consequently, $R_i\sigma' \in d(\mathcal{K}_{v_{i-1}}\sigma')$.

(P2) We need to show that for every $c \in C_i$ and root path $\pi' = v_1, \ldots, v_n$ where v_n is a leaf in $\mathcal{T}^s_i$, we have that $c \in d(\overline{\mathcal{K}}^{s,v_n}\sigma')$. We know that σ solves $\mathbf{C}$, and hence, $c \in d(\overline{\mathcal{K}}^{s,v_n}\sigma)$. From this, using the same arguments as above, we obtain that $c \in d(\overline{\mathcal{K}}^{s,v_n}\sigma')$.

(P3) We have to show that σ' also passes (A4.1), (A4.2.b), and (A4.2.b), and we know that σ passes them. First, we show that σ' passes (A4.1). We know:

(**) $C'_i \cap d(\overline{\mathcal{K}}^{s,v}\sigma) = \emptyset$ for every leaf v of $\mathcal{T}^s_i$ and every i.

If there exists $c \in C'_i \cap d(\overline{\mathcal{K}}^{s,v}\sigma')$, then it is easy to see that, since a is a new intruder atom, from $(\overline{\mathcal{K}}^{v,s}\sigma')_{|a\to\{t\}^s_k\sigma}$ we can still derive c. Together with the fact that $(\overline{\mathcal{K}}^{s,v}\sigma')_{|a\to\{t\}^s_k\sigma} = \overline{\mathcal{K}}^{s,v}\sigma$, this is a contradiction to (**).

To see that σ' passes (A4.2.a) and (A4.2.b), it suffices to observe that if condition C. and conditions C.–E. in (A4.2.a) and (A4.2.b), respectively, are satisfied for σ', then also for σ. Here we again use that a is a new intruder atom and that we can replace a again by $\{t\}^s_k\sigma$. $\square$

We can now phrase what we were aiming for:

Proposition 3.6.9 *Let μ be the substitution obtained from ν by repeatedly applying the transformation described in Proposition 3.6.8 (from σ to σ') until no secure channel encoding terms are left.*

Then $\mu(\boldsymbol{B}^s)$ is a concrete branching structure for P which satisfies $\mathcal{C}$.

Note that since ν was returned by SolveStrategy, and hence, solves $\mathbf{C}$ and passes the tests in (A4), ν satisfies (P1)–(P3) and Proposition 3.6.8 can be applied repeatedly.

Note that by Remark 3.6.1 this proposition implies soundness of SolveStrategy.

Proof We first show that every edge in $\mu(\mathbf{B}^s)$ corresponds to a transition in $\mathcal{G}_P$ by induction on the length of root paths $\pi' = v_1, \ldots, v_h$ in $\mathbf{B}^s$. That is, we show by induction on h:

(i) $\mu(\ell_V^s)(v_1)$ is the initial state of P,

(ii) $\mu(\ell_V^s)(v_i) \in \mathcal{G}_P$ for $i \in \{1, \ldots, h\}$, and

(iii) $\mu(\ell_V^s)(v_i) \xrightarrow{\mu(\ell_E^s)(v_i,v_{i+1})} \mu(\ell_V^s)(v_{i+1}) \in \mathcal{G}_P$ for $i \in \{1, \ldots, h-1\}$.

For $h = 1$, we only need to show that $\mu(\ell_V^s)(v_1)$ is the initial state of P, which is obvious. For the induction step assume that we are given a root path $\pi' = v_1, \ldots, v_{h+1}$ in $\mathbf{B}^s$ (and thus, in $\overline{\mathbf{B}}^s$). Let $\rho^s = \rho_1^s, \ldots, \rho_h^s$ with $\rho_i^s = R_i^s \Rightarrow S_i^s$ be the pr-sequence associated with π' in $\mathbf{B}^s$ and let $\overline{\rho}^s = \overline{\rho}_1^s, \ldots, \overline{\rho}_h^s$ with $\overline{\rho}_i^s = \overline{R}_i^s \Rightarrow \overline{S}_i^s$ be the pr-sequence associated with π' in $\overline{\mathbf{B}}^s$. Moreover, let $\mu(\mathcal{K}^{s,v_i})$ be the knowledge of the intruder at v_i in $\mu(\mathbf{B}^s)$. Note that $\mu(\mathcal{K}^{s,v_i}) \subseteq \mathcal{K}_0 \cup \{S_1^s\mu, \ldots, S_{i-1}^s\mu\}$ where $\mathcal{K}_0$ is the initial intruder knowledge in P. This inclusion could be strict since some of the S_j^s may be secure channel terms, and hence, are not written into the intruder's knowledge. Let $\mu(\overline{\mathcal{K}}^{s,v_i}) = \mathcal{K}_0 \cup \{\overline{S}_1^s\mu, \ldots, \overline{S}_{i-1}^s\mu\}$ be the intruder's knowledge at v_i w.r.t. $\overline{\mathbf{B}}^s$. Recall that for $\overline{\mathbf{B}}^s$ all messages are written into the intruder's knowledge.

We next show that the statements (i), (ii), and (iii) from above hold for π'. Obviously, statement (i) still holds. By the induction hypothesis, we know that $\mu(\ell_V^s)(v_i) \in \mathcal{G}_P$ for $i \in \{1, \ldots, h\}$ and $\mu(\ell_V^s)(v_i) \xrightarrow{\mu(\ell_E^s)(v_i,v_{i+1})} \mu(\ell_V^s)(v_{i+1}) \in \mathcal{G}_P$ for every $i \in \{1, \ldots, h-1\}$. We have to show that

$\mu(\ell_V^s)(v_{h+1}) \in \mathcal{G}_P$ and $\mu(\ell_V^s)(v_h) \xrightarrow{\mu(\ell_E^s)(v_h,v_{h+1})} \mu(\ell_V^s)(v_{h+1}) \in \mathcal{G}_P$. We distinguish between the different types of the symbolic transitions. For symbolic ε-transitions this is obvious.

For symbolic intruder transitions, the main point to show is that we have $R_h^s\mu \in d(\mu(\mathcal{K}^{s,v_h}))$. First assume that R_h^s is not a secure channel term. It follows that $R_h^s = \overline{R}_h^s$. Since μ satisfies condition (P1) from above, we know that $\overline{R}_h^s\mu \in d(\mu(\overline{\mathcal{K}}^{s,v_h}))$. We know that $\mu(\mathcal{K}^{s,v_h}) \subseteq \mu(\overline{\mathcal{K}}^{s,v_h})$ and that the only difference between $\mu(\mathcal{K}^{s,v_h})$ and $\mu(\overline{\mathcal{K}}^{s,v_h})$ is that the latter set may contain secure channel encoding messages. Since μ, R_h^s, and the terms in $\mathcal{K}^{s,v_h}$ do not contain secure channel keys, we immediately obtain $R_h^s\mu \in d(\mu(\mathcal{K}^{s,v_h}))$ by Lemma 3.6.5. The argument in case R_h^s is a secure channel term is similar.

For symbolic secure channel transitions, assume that $\ell_E^s(v_h, v_{h+1})$ is of the form $[j, f, \mathsf{sc}(n, n', R'), \mathsf{sc}]$. We have to show that $\mu(\mathsf{sc}(n, n', R')) \in \mu(\mathcal{S}^{s,v_h})$ and $\mu(R_{v_h}^s) = \mu(\mathsf{sc}(n, n', R'))$ where $\mathcal{S}^{s,v_h}$ denotes the secure channel in the symbolic state $\ell_V^s(v_h)$. By definition of symbolic secure channel transitions, we know that $\mathsf{sc}(n, n', R') \in \mathcal{S}^{s,v_h}$, and thus, $\mu(\mathsf{sc}(n, n', R')) \in \mu(\mathcal{S}^{s,v_h})$. If $R_{v_h}^s$ is of the form $\mathsf{sc}(n, n', R)$, then, by construction of $\overline{\mathbf{B}}^s$, $\overline{R}_h^s$ is of the form $\{R\}_k^s$ for some secure channel key k such that $\{R'\}_k^s \in \overline{\mathcal{K}}^{s,v_h}$ and k does not occur in any other term in $\overline{\mathcal{K}}^{s,v_h}$. We know that $\overline{R}_h^s\mu \in d(\overline{\mathcal{K}}^{s,v_h}\mu)$. Since k does not occur in μ nor in R and R', with Lemma 3.6.5 we obtain that $R\mu = R'\mu$, and thus, $\mu(R_h^s) = \mu(\mathsf{sc}(n, n', R'))$. Thus, we have shown that the edges in $\mu(\mathbf{B}^s)$ correspond to transitions in $\mathcal{G}_P$.

The last step is to show that $\mu(\mathcal{T}_j^s)$ is a strategy tree for (C_j, C_j'). Properties (T1) and (T2) of Definition 3.3.1 are satisfied as we have just shown. (T3) follows directly from (S3). (T4) and (T5) follow directly from the fact that μ passes (A4.2.a) and (A4.2.b). $\square$

3.6.3 Completeness of SolveStrategy

In this section, we show completeness of SolveStrategy. To this end, let $(P, \mathcal{C})$ be an instance of STRATEGY and assume $(P, \mathcal{C}) \in$ STRATEGY. We have to show that there is a run of SolveStrategy in which yes is returned. From Remark 3.6.1, we conclude that there exists a branching structure $\mathbf{B}$ for P that satisfies $\mathcal{C}$.

Our proof proceeds in three steps:

(C1) We turn $\mathbf{B}$ into a symbolic branching structure $\mathbf{B}^s$ together with a substitution τ such that $\tau(\mathbf{B}^s) = \mathbf{B}$.

Note that SolveStrategy can guess $\mathbf{B}^s$ in step (A1).

(C2) We consider the constraint system $\mathbf{C}$ that is constructed in (A2), provided $\mathbf{B}^s$ was guessed in (A1), and show that τ is a solution of $\mathbf{C}$.

It follows that τ is also a solution of a simple constraint system, say $\mathbf{C}'$, in the sound and complete solution set of $\mathbf{C}$. Thus, there is a run of the constraint solver which outputs $\mathbf{C}'$.

(C3) We show that the substitution $\tau_{\mathbf{C}'}$ associated with $\mathbf{C}'$ passes the tests performed in the third step of SolveStrategy.

This means SolveStrategy outputs yes

The technical problem we have to overcome is that the above steps can only be carried out if $\mathbf{B}$ is what we call a minimal branching structure. So we also need to show that this is the case without loss of generality.

Roughly speaking, minimal branching structures satisfy two properties: First, they should not contain superfluous transitions. Second, the secure channel transitions are in some sense complete. We now define these structures formally.

Preliminaries

To define minimal branching structures, we first need to introduce sch-functions and sch-completeness. To motivate these notions, we sketch the first step of our completeness proof, (C1).

The symbolic branching structure $\mathbf{B}^s$ and the substitution τ are constructed in an inductive manner starting from the root of $\mathbf{B}$. Given a vertex v such that the symbolic state associated with v is already defined and the substitution τ is already defined for the variables in $\mathcal{X}_v$ (the set of variables used in vertex v), we will define for each vertex $v' \in \mathbf{B}$ with $(v, v') \in \mathbf{B}$ the symbolic transition label of the edge (v, v') and the symbolic state associated with v' in $\mathbf{B}^s$ together with the substitution of the variables used in v'. In order to define the symbolic secure channel components of the state associated with vertex $v \in \mathbf{B}^s$ we will define what we call valid sch-functions.

Informally speaking, a valid sch-function for a concrete branching structure $\mathbf{B}$ associates with each vertex $v \in \mathbf{B}$ a sequence of secure channel terms which corresponds to a possible symbolic secure channel state in $v \in \mathbf{B}^s$. More precisely, a valid sch-function ℓ_{sch} associates with the root r of $\mathbf{B}$ the empty sequence. This corresponds to the empty secure channel in the symbolic state associated with r in $\mathbf{B}^s$. Whenever a secure channel message m is written into the secure channel by a transition from vertex v to v' in $\mathbf{B}$, then the sequence of secure channel terms associated with v' is that of v extended by the right-hand side of the principal rule associated with the edge (v, v'). Whenever a secure channel message m' is read from the secure channel, a particular secure channel term which matches with m' is removed from the sequence $\ell_{\mathsf{sch}}(v)$ of secure channel terms associated with v.

We now turn to the formal definition of the notions explained above. Formally, we call a sequence $s = (S_1, \ldots, S_k)$ of sch-terms an *sch-sequence.* For a term S, a substitution σ of variables such that $\mathcal{V}(S_i) \subseteq \operatorname{dom}(\sigma)$ for some $i \in \{1, \ldots, k\}$, and a message m, we say that the sch-sequence s' is an *(S, m, σ)-successor* of s if

$$s' = \begin{cases} (S_1, \ldots, S_{i-1}, S_{i+1}, \ldots, S_k) & \text{if } S_i\sigma = m \text{ and } S_j \neq S_i \text{ for all } j < i \text{ and} \\ & S \text{ is not an } \mathsf{sch}\text{-term}, \\ (S_1, \ldots, S_{i-1}, S_{i+1}, \ldots, S_k, S) & \text{if } S_i\sigma = m \text{ and } S_j \neq S_i \text{ for all } j < i \text{ and} \\ & S \text{ is an } \mathsf{sch}\text{-term.} \end{cases}$$

We call the term S_i the *term removed from s.*

A function ℓ_{sch} which maps every vertex $v \in \mathbf{B}$ to an sch-sequence is called an *sch-function.* Such a function is *valid* if for every $v \in \mathbf{B}$ the following conditions are satisfied:

(F1) $\mathcal{S}^v = \sigma^v(\{S_1, \ldots, S_k\})$ with $\ell_{\mathsf{sch}}(v) = (S_1, \ldots, S_k)$. (Note that $\{S_1, \ldots, S_k\}$ should be treated as a multiset.)

(F2) If v' is a successor of v in $\mathbf{B}$ and $\ell_E(v, v')$ is of the form $[i]$ or $[i, m, I]$ (i.e., the transition associated with (v, v') is an ε- or intruder transition), then

$$\ell_{\mathsf{sch}}(v') = \begin{cases} \ell_{\mathsf{sch}}(v) & \text{if } S \text{ is not an } \mathsf{sch}\text{-term} \\ (S_1, \ldots, S_k, S) & \text{otherwise} \end{cases}$$

where S is the right-hand side of the principal rule associated with (v, v').

(F3) If v' is a successor of v in $\mathbf{B}$ and $\ell_E(v, v')$ is of the form $[i, m, \mathsf{sc}]$ (i.e., the transition associated with (v, v') is a secure channel transition), then

$\ell_{\mathsf{sch}}(v')$ is a (S, m, σ^v)-successor of $\ell_{\mathsf{sch}}(v)$ where S is the right-hand side of the principal rule associated with (v, v').

In (F3), if S' is the term removed from $\ell_{\mathsf{sch}}(v)$, we call S' the *sch-term removed in* (v, v'). For ε- and intruder transitions, there is no removed sch-term.

As mentioned above, minimal branching structures will be defined in such a way that they satisfy two properties: i) there are no superfluous transitions and ii) the secure channel transitions are in some sense complete. To motivate the latter condition, we need to sketch (C3): We show that $\tau_{\mathbf{C}'}$ passes the tests performed in (A4), in particular, (A4.2.a), which says that all secure channel messages in the secure channel which could be read by applying a principal rule are in fact read by applying this rule. For this condition to be satisfied by $\tau_{\mathbf{C}'}$, we have to impose some specific structure on the branching structure $\mathbf{B}$.

Suppose, for example, that the symbolic secure channel $\mathcal{S}^{s,v}$ associated with $v \in \mathbf{B}^s$ contains terms $\mathsf{sc}(n, n', x)$ and $\mathsf{sc}(n, n', y)$ and $\tau(x) = \tau(y)$ where $\mathbf{B}^s$ and τ are the symbolic branching structure and substitution constructed from $\mathbf{B}$, respectively. Furthermore, assume that there is a principal rule of the form $\mathsf{sc}(n, n', z) \Rightarrow S$ applicable at v in $\mathbf{B}$, then, by definition of strategy trees (Definition 3.3.1, 4.) there has to be a secure channel transition with $\mathsf{sc}(n, n', z) \Rightarrow S$ being the associated principal rule of this transition where the message $\tau(\mathsf{sc}(n, n', x)) = \tau(\mathsf{sc}(n, n', y))$ is read from the secure channel. In the corresponding symbolic secure channel transition one of the two terms $\mathsf{sc}(n, n', x)$ and $\mathsf{sc}(n, n', y)$ is removed from the symbolic secure channel. The substitution $\tau_{\mathbf{C}'}$ does not have to fulfill the condition $\tau_{\mathbf{C}'}(x) = \tau_{\mathbf{C}'}(y)$ anymore. So the messages $\tau_{\mathbf{C}'}(\mathsf{sc}(n, n', x))$ and $\tau_{\mathbf{C}'}(\mathsf{sc}(n, n', y))$ are not necessarily the same. Still, both could be read when applying $\mathsf{sc}(n, n', z) \Rightarrow S$, and hence, in the concrete branching structure induced by $\mathbf{B}^s$ and $\tau_{\mathbf{C}'}$ there must be secure channel transitions for both messages. Therefore, we want to make sure that these transitions already occur in $\mathbf{B}$. This is captured by the notion of sch-completeness defined next.

Recall the definition of V^j from the beginning of Section 3.6. We call $\mathbf{B}$ *sch-complete* if there is a valid sch-function ℓ_{sch} for $\mathbf{B}$ and for all j, $v \in V^j$, and successors v' of v in $\mathbf{B}$ such that

- the label of the transition associated with (v, v') is $[i, m, \mathsf{sc}]$ for some i

and $m \in \mathcal{S}^v$,

- the principal vertex associated with (v, v') is f for some vertex f in Π_i^v, and
- the right-hand side of the principal rule associated with (v, v') is S for some term S,

the following conditions are satisfied: For every (S, m, σ^v)-successor s of $\ell_{\mathsf{sch}}(v)$, there is a successor v'' of v such that $\ell_{\mathsf{sch}}(v'') = s$, the label of the transition associated with (v, v'') is $[i, m, \mathsf{sc}]$, and the principal vertex associated with (v, v'') is f.

We can now define minimal concrete branching structures.

Definition 3.6.10 (minimal branching structure) *A branching structure $\boldsymbol{B}$ for P is called* minimal *if it satisfies the following conditions:*

(M1) *$\boldsymbol{B}$ is sch-complete.*

(M2) *For all $i \in \{1, \ldots, l\}$ and $(v, v'), (v, v'') \in E^i$ where $v' \neq v''$ and the transitions associated with (v, v') and (v, v'') are ε-transitions:*

- *The principal associated with (v, v') differs from the principal associated with (v, v''), or*
- *the principal vertex associated with (v, v') differs from the principal vertex associated with (v, v'').*

(M3) *For all $i \in \{1, \ldots, l\}$ and $(v, v'), (v, v'') \in E^i$ where $v' \neq v''$ where the transitions associated with (v, v') and (v, v'') are secure channel transitions:*

- *The principal associated with (v, v') differs from the principal associated with (v, v''), or*
- *the principal vertex associated with (v, v') differs from the principal vertex associated with (v, v''), or*
- *the sch-term removed in (v, v') differs from the sch-term removed in (v, v'').*

(M4) *For all $i \in \{1, \ldots, l\}$ and $(v, v'), (v, v'') \in E^i$: If $\ell_V(v, v') = [j, m, I]$ for some j and m, then $\ell_V(v, v'') = [j, m, I]$.*

(M2) – (M4) say that there are no superfluous transitions in the strategy trees of a minimal branching structure. These conditions correspond to the conditions (S2) – (S4) for symbolic strategy trees. (M1) is the completeness condition explained above.

The following lemma is easy to show, simply by cutting down a given branching structure appropriately.

Lemma 3.6.11 *If there exists a branching structure for P which satisfies $\mathcal{C}$, then there exists a minimal branching structure with this property.*

Proof of Completeness

We can now carry out (C1), (C2), and (C3), assuming that the given branching structure $\mathbf{B}$ is minimal.

Step (C1) Our goal is to construct a symbolic branching structure $\mathbf{B}^s$ for P and a substitution τ such that $\tau(\mathbf{B}^s) = \mathbf{B}$. In particular, we define $\mathbf{B}^s$ in such a way that the set of vertices and edges of $\mathbf{B}^s$ coincides with the set of vertices and edges in $\mathbf{B}$, respectively. More precisely, define

- $V^{s,\pi} = V^\pi$ and $V^{s,i} = V^i$ for every i, and
- $E^{s,\pi} = E^\pi$ and $E^{s,i} = E^i$ for every i, and
- $r^{s,\pi} = r^\pi$ and $r^{s,i} = r^i$ for every i.

It remains to define the labeling functions ℓ^s_V and ℓ^s_E of $\mathbf{B}^s$ and the sets of used variables associated with each vertex $v \in V$. For the root r of π^s we set $\ell^s_V(r) = q^s_0$ where q^s_0 is the symbolic initial state of the protocol P. We define the set $\mathcal{X}_r$ of used variables to be empty.

Assume that for $v \in V$ the set of used variables $\mathcal{X}_v$ and the symbolic state $\ell^s_V(v)$ associated with v are already defined. Let $v' \in V$ with $(v, v') \in E$. We need to define $\ell^s_E(v, v')$ and $\ell^s(v')$.

We define the first component of $\ell_E(v, v')$ to be j. We know that $\ell_V(v) \xrightarrow{\ell_E(v,v')} \ell_V(v') \in \mathcal{G}_P$. Let f be the principal vertex associated with this transition. Let $\mathcal{X}' = \mathcal{V}R \setminus \mathrm{dom}(\sigma^v)$ where $\ell^v_j(r^v_j, f) = R \Rightarrow S$ is the principal rule associated with (v, v') in $\mathbf{B}$. (Note that $\mathcal{X}'$ is the set of new variables in R). Define $\Pi^{s,v'}_j = \widehat{\Pi}^{s,v}_j \downarrow f$ where $\widehat{\Pi}^{s,v}_j$ is $\Pi^{s,v}_j$ with variables $x \in \mathcal{X}'$ renamed by $x_{v'}$.

The set $\mathcal{X}_{v'}$ of used variables in v' is set to be $\mathcal{X}_{v'} = \mathcal{X}_v \cup \{x_{v'} \mid x \in \mathcal{X}'\}$.

In the following, we denote by $\hat{R}$ and $\hat{S}$ the terms R and S where the variables occurring in R and S, respectively, are renamed by there corresponding indexed version from $\mathcal{X}_{v'}$.

To define the intruder knowledge $\mathcal{K}^{s,v'}$, the secure channel component $\mathcal{S}^{s,v'}$ of $\ell^s_V(v')$ and the edge label $\ell^s_E(v,v')$, we distinguish between the different types of transition labels $\ell_E(v,v')$.

- $\ell_E(v,v') = [j]$ (ε-transition): We set
  (a) $\ell^s_E(v,v') = [j,f]$;
  (b) $\mathcal{K}^{s,v'} = \mathcal{K}^{s,v} \cup \{\hat{S}\}$ if $\hat{S}$ is not a secure channel term and $\mathcal{K}^{s,v'} = \mathcal{K}^{s,v} \cup \{R'\}$ if $\hat{S}$ is a secure channel term of the form $\hat{S} = \mathsf{sc}(\cdot,\cdot,R')$; and
  (c) $\mathcal{S}^{s,v'} = \mathcal{S}^{s,v}$ if $\hat{S}$ is not a secure channel term and $\mathcal{S}^{s,v'} = \mathcal{S}^{s,v} \cup \{\hat{S}\}$ if $\hat{S}$ is a secure channel term.
- $\ell_E(v,v') = [j,m,I]$ (intruder transition): We define $\ell^s_E(v,v') = [j,f,I]$ and the rest as in (b) and (c) above.
- $\ell_E(v,v') = [j,m,\mathsf{sc}]$ (secure channel transition): We define $\ell^s_E(v,v') = [j,f,S',\mathsf{sc}]$ where $\ell_{\mathsf{sch}}(v')$ is a $(\hat{S},m,\sigma^v)$-successor of $\ell_{\mathsf{sch}}(v)$ and S' is the term removed from $\ell_{\mathsf{sch}}(v)$. The intruder knowledge $\mathcal{K}^{s,v'}$ is updated as in (b) above. The secure channel component is updated to $\mathcal{S}^{s,v'} = \mathcal{S}^{s,v} \setminus \{S'\}$ if $\hat{S}$ is not a secure channel term and $\mathcal{S}^{s,v'} = \mathcal{S}^{s,v} \setminus \{S'\} \cup \{\hat{S}\}$ if $\hat{S}$ is a secure channel term.

Using the fact that $\mathbf{B}$ is a minimal (concrete) branching structure, it is easy to verify that $\mathbf{B}^s$ is a symbolic branching structure.

We now define the substitution τ. The domain of τ is the set $\mathcal{X} = \bigcup_{v\in V} \mathcal{X}_v$, i.e., the set of used variables in $\mathbf{B}^s$. For a variable $x_v \in \mathcal{X}$ we set $\tau(x_v) = \sigma^v(x)$. By induction on h, it is easy to see that for each root path $v_1,\ldots,v_h$ in $\mathbf{B}^s$ we have that $\tau(\ell^s_V(v_i)) = \ell_V(v_i)$ and $\tau(\ell^s_E(v_i,v_{i+1})) = \ell_E(v_i,v_{i+1})$ for every i. From this, it immediately follows that $\tau(\mathbf{B}^s) = \mathbf{B}$.

Step (C2) We have to show that τ is a solution of the constraint system $\mathbf{C}$ constructed from $\overline{\mathbf{B}}^s$. There are two types of constraints in $\mathbf{C}$: First, constraints that were introduced for an edge $(v,v') \in E$, the intruder constraints, and second, the strategy constraints.

Let $R\colon T$ be an intruder constraint of $\mathbf{C}$ that was introduced for the edge

$(v, v') \in E$. We have to show that $\tau(R) \in d(\tau(T))$. There are three different cases that we have to distinguish: (a) $\ell_E(v, v') = [j, m, I]$ and m is not a secure channel message, (b) $\ell_E(v, v') = [j, m, I]$ and m is a secure channel message, and (c) $\ell_E(v, v') = [j, m, \mathsf{sc}]$.

Case (a). We know that $\tau(R) = \sigma^{v'}(R) = m \in d(\mathcal{K}^v)$ because R is the left-hand side of the principal rule associated with the edge (v, v') in $\mathbf{B}$. By construction of $\overline{\mathbf{B}}^s$, we have that $\mathcal{K}^v \subseteq \tau(\overline{\mathcal{K}}^{s,v}) = \tau(T)$ where $\overline{\mathcal{K}}^{s,v}$ is the intruder's knowledge at vertex v in $\overline{\mathbf{B}}^s$. Thus, it follows that $m \in d(\tau(T))$.

Case (b). We know that m is of the form $m = \mathsf{sc}(n, n', m')$ with $m' = \tau(R)$. We have that $m \in d(\mathcal{K}^v)$. Since, by construction, $\mathcal{K}^v$ does not contain secure channel terms, it follows that $n \in \mathcal{K}^v$ and $m' \in d(\mathcal{K}^v)$. As in (a) we know that $\mathcal{K}^v \subseteq \tau(\overline{\mathcal{K}}^{s,v}) = \tau(T)$. Thus, $m' \in d(\tau(T))$.

Case (c). The term R is a term of the form $\{R'\}^s_{k_{w'}}$ where $k_{w'}$ is the sch-key introduced in the transition associated with an edge $(w, w') \in E$ where w is a predecessor of v. So the left-hand side of the principal rule associated with (v, v') in $\mathbf{B}^s$ has the form $\mathsf{sc}(n, n', R')$ and the right-hand side of the principal rule associated with (w, w') in $\mathbf{B}^s$ has the form $\mathsf{sc}(n, n', R'')$. We know that $\tau(\mathsf{sc}(n, n', R')) = m = \tau(\mathsf{sc}(n, n', R''))$. Hence, $\tau(\{R'\}^s_{k_{w'}}) = \tau(\{R''\}^s_{k_{w'}})$. Since $\{R''\}^s_{k_{v''}} \in T$, it follows that $\tau(\{R'\}^s_{k_{w'}}) \in d(\tau(T))$.

Since a strategy constraint is of the form $a : \overline{\mathcal{K}}^{s,v}$ for a leaf $v \in \mathcal{T}_i$ and $a \in C_i$ for some i and $\tau(\mathcal{T}_i^s) = \mathcal{T}_i$ fulfills (C_i, C_i') we know that τ fulfills this constraint.

Step (C3) We will first show that τ passes the tests that in (A4) of $\mathsf{SolveStrategy}$ and from this it will follow that $\tau_{\mathbf{C}'}$ passes these tests.

Test (A4.1). Let v be a leaf of some $\mathcal{T}_i$. We know that $C_i' \cap d(\mathcal{K}^v) = \emptyset$ and that $\tau(\ell^s_V(v)) = \ell_V(v)$. Consequently, τ passes the first check because $\mathcal{T}_i$ satisfies (C_i, C_i').

Test (A4.2). For every i and vertex $v \in V^i$, we have to show that the following conditions are satisfied.

Test (A4.2.a). For $m \in \mathcal{S}^{s,v}\tau$, h, f, R such that

A. $(r_h^{s,v}, f) \in E_h^{s,v}$,

B. R is the LHS of $\ell_h^{s,v}(r_h^{s,v}, f)$, and

C. $R\tau$ matches with m

there is an edge $(v, v') \in E$ such that the label $\tau(\ell^s_E(v, v')) = [h, m, \mathsf{sc}]$ and $\ell^s_E(v, v')$ has the form $[h, f, \cdot, \mathsf{sc}]$.

Test (A.4.2.b). If there exists an edge $(v, v') \in E$ and $f' \neq f$ such that

A. the transition corresponding to the edge (v, v') is labeled with $[h, f, I]$,

B. R is the LHS of $\ell^{s,v}_h(r^{s,v}, f)$,

C. $(r^{s,v}_h, f') \in E^{s,v}_h$,

D. R' is the LHS of $\ell^{s,v}_h(r^{s,v}_h, f')$, and

E. $R'\tau$ matches with $\hat{R}\tau$ where $\hat{R}$ is obtained from R by replacing every non-indexed variable x in R by $x_{v'}$,

then there exists v'' such that $(v, v'') \in E$ and the label of (v, v'') in $\mathbf{B}^s$ is $[h, f', I]$.

Let $v \in V^i$ for some i. To show (A4.2.a), let $m \in \mathcal{S}^{s,v}\tau$, h, and f satisfy A.–C. as above. We know that $\tau(\mathcal{T}^s_i) = \mathcal{T}_i$ for all i. Since $\mathcal{T}_i$ is a strategy tree, there is an edge $(v, v') \in E$ such that the label is $\tau(\ell^s_E(v, v')) = \ell_E(v, v') = [h, m, \mathsf{sc}]$. This is sufficient.

To show (A4.2.b), let $(v, v') \in E$ and h, f, f', R, and R' satisfy A.– C. as above. We know that $\tau(\mathcal{T}^s_i) = \mathcal{T}_i$ for all i. Since $\mathcal{T}_i$ is a strategy tree, there is a vertex v'' such that the label of (v, v'') is $[h, m, I]$ and the principal vertex associated with (v, v') in $\mathbf{B}$ is f. By construction of $\mathbf{B}^s$, we have that the label of (v, v'') is $[h, f', I]$ as desired.

So τ passes the tests in (A4) of $\mathsf{SolveStrategy}$. Now we show that $\tau_{\mathbf{C}'}$ passes these tests as well.

Test (A4.1). Let v be a leaf of some $\mathcal{T}^s_i$ and suppose that there is an atom $a \in C'_i$ such that $a \in d(\mathcal{K}^{s,v}\tau_{\mathbf{C}'})$. Then it is easy to see that $a \in d(\mathcal{K}^{s,v}\tau)$, which is a contradiction to (C1).

Test (A4.2.a). Let $m \in \mathcal{S}^{s,v}\tau_{\mathbf{C}'}$, h, f such that

A. $(r^{s,v}_h, f) \in E^{s,v}_h$,

B. R is the LHS of $\ell^{s,v}_h(r^{s,v}_h, f)$, and

C. $R\tau_{\mathbf{C}'}$ matches with m.

Let $R' \in \mathcal{S}^{s,v}$ such that $m = R'\tau_{\mathbf{C}'}$. Since $R\tau_{\mathbf{C}'}$ matches with $R'\tau_{\mathbf{C}'}$, it is easy to see that $R'\tau$ matches with $R\tau$ (since τ is obtained from $\tau_{\mathbf{C}'}$ by replacing intruder atoms by messages). By definition of ℓ_{sch}, there is a successor v' of v such that $r^{v'}_h = f$ and $\ell_{\mathsf{sch}}(v')$ is a valid (S, m, σ^v)-successor of $\ell_{\mathsf{sch}}(v)$

with the term R' removed. So by construction of $\mathbf{B}^s$, the label $\ell^s_E(v, v')$ is $\ell^s_E(v, v') = [h, f, R', \mathsf{sc}]$, and thus, $\tau_{\mathbf{C}'}(\ell^s_E(v, v')) = [h, m, \mathsf{sc}]$ as desired.

Test (A4.2.b). Let $(v, v') \in E, h, f, R, f'$ and R' satisfy A.–C. as above. We have to show that there is a successor v'' of v in $\mathbf{B}$ such that the label of (v, v'') in $\mathbf{B}^s$ is $[h, f', I]$. Since we know that $R'\tau_{\mathbf{C}'}$ matches with $\hat{R}\tau_{\mathbf{C}'}$, as above we obtain that $R'\tau$ matches with $\hat{R}\tau$. Since τ passes the test (A4.2.b), there exists a vertex v'' such that $(v, v'') \in E$ and the label of (v, v'') in $\mathbf{B}^s$ is $[h, f', I]$.

This completes the proof of completeness of SolveStrategy.

Chapter 4

AMC for Cryptographic Protocols

The alternating time temporal logic (ATL) has been used to specify and analyze security properties of cryptographic protocols, see [KR01, KR02]. In this chapter we introduce the alternating-time μ-calculus (AMC) for cryptographic protocols and show that a fragment, called $\mathcal{I}$-monotone, of AMC is decidable for cryptographic protocols when reasonable restrictions on the protocols are imposed.

In this chapter we formalize the possible executions of protocols along with the Dolev-Yao intruder in terms of a certain class of infinite-state concurrent game structures [AHK02], which we call *security-specific concurrent game structures.* These concurrent game structures have an *infinite* state space since at every execution step the Dolev-Yao intruder can choose messages to be sent to principals among an *infinite* set of possible messages. Similar to [KKW06], we model the realistic situation that (honest and dishonest) principals may take actions at the same time and may receive/write several messages from/to other principals at the same time. Since many cryptographic protocols with game-theoretic security requirements assume resilient channels (also called secure channels here), i.e., channels that, unlike the network, are not under the control of the Dolev-Yao intruder, our model comprises such channels. We distinguish between direct and scheduled secure channels: A direct secure channel is a direct link between principals. Messages sent on scheduled secure channels are first sent to a buffer before being delivered to the intended recipient. The buffer is a player in the security-specific concurrent game structure and may team up with (honest or dishonest) principals or other scheduled secure channels, as can be specified by an AMC-formula. Honest principals are specified by finite edge-labeled trees where an edge is labeled by a rule which describes a possible receive-send action of a principal at the current step. Vertices in these trees may have self-loops to allow a principal to stay in the current state.

Based on the security-specific concurrent game structures that we define, game-theoretic security requirements for protocols can conveniently be expressed in terms of AMC-formulas (or alternatively, ATL*-formulas). In order to decide whether a given protocol satisfies a given security property, expressed as AMC-formula, one has to decide the AMC-model checking problem over the security-specific concurrent game structures, where the input to the problem is the protocol (which together with the Dolev-Yao intruder induces the security-specific concurrent game structure) and the AMC-formula.

Our main technical results are as follows: We show that the above model checking problem is undecidable for a class of protocols in which honest principals may be what we call *non-greedy*, i.e., they may ignore received messages even though they conform to the protocol specification. The undecidability result holds for a relatively simple, fixed AMC-formula. Fortunately, in typical protocol specifications, honest principals are greedy, i.e., they do not ignore messages that conform to the protocol specification. Hence, requiring honest principals to be greedy is reasonable from a practical point of view. We also exhibit another source of undecidability, namely protocols that involve *scheduled* secure channels *from the Dolev-Yao intruder* (i.e., dishonest principals) to honest principals. This undecidability result holds for greedy principals and again a fixed, simple AMC-formula. Since we allow the Dolev-Yao intruder to send messages over direct secure channels to principals, disallowing scheduled secure channels from the Dolev-Yao intruder to honest principals does not limit the power of the intruder. These undecidability results show that to obtain decidability it is necessary to consider only protocols with greedy principals and without scheduled secure channels from the Dolev-Yao intruder to honest principals. For this class of protocols we indeed obtain decidability, more accurately (co-)NEXPTIME-completeness, of the model checking problem for an expressive fragment of AMC, consisting of what we call $\mathcal{I}$-positive ($\mathcal{I}$-negative) AMC-formulas, where $\mathcal{I}$ is the name of the Dolev-Yao intruder in the concurrent game structure. An AMC-formula φ is $\mathcal{I}$-positive if all subformulas of φ of the form $\langle\langle A\rangle\rangle\psi$ with $\mathcal{I} \in A$ fall under an even number of negations and all subformulas of φ of the form $\langle\langle A\rangle\rangle\psi$ with $\mathcal{I} \notin A$ fall under an odd number of negations; a formula is $\mathcal{I}$-negative if its negation is $\mathcal{I}$-positive. We subsume the set of $\mathcal{I}$-positive and $\mathcal{I}$-negative formulas under the notion $\mathcal{I}$-monotone formulas. The same terminology can be applied to ATL*-formulas. It is easy to see that the property of being $\mathcal{I}$-positive/-negative is invariant under the translation from ATL* to AMC as described in [AHK02]. Kremer and Raskin were the first to express game-theoretic security properties in terms of fair ATL [KR01, KR02]. It turns out that all the properties that they have formulated, including for instance various forms of fairness, timeliness, balance, and abuse-freeness, fall into the $\mathcal{I}$-monotone fragment of ATL*, and hence, the $\mathcal{I}$-monotone fragment of AMC, indicating that the $\mathcal{I}$-monotone fragment suffices for most properties of interest.

The complexity upper bound is proved by a novel combination of techniques from the theory of infinite games, such as parity games and memoryless strategies, and techniques from cryptographic protocol analysis for reachability properties.

4.1 AMC and Parity Games

Following [AHK97, AHK02], in this section we recall the definition of concurrent game structures and AMC. We also introduce parity games for AMC-model checking.

4.1.1 Concurrent Game Structures

Our definition of a concurrent game structure differs from the one in [AHK02] in two aspects: First, the structures that we consider may have an *infinite* state space and in one state players may have an *infinite* number of possible moves. Second, while in [AHK02] a move of a player is identified with a natural number, in our setting it is more convenient to allow arbitrary values; in the context of cryptographic protocol moves will be vertices of trees and terms.

We define concurrent game structures as follows. A *concurrent game structure (CGS)* is a tuple $S = \langle \Sigma, Q, \mathbb{P}, \pi, \Delta, \delta \rangle$ where

- Σ is a non-empty, finite set of *players*,
- Q is a (possibly infinite) set of *states*,
- $\mathbb{P}$ is a finite set of *propositional variables/propositions*,
- $\pi : Q \to 2^{\mathbb{P}}$ is a *labeling function* (which assigns every state to the set of propositions true in this state),
- Δ is a function which for each state $q \in Q$ and each player $a \in \Sigma$ returns a (possibly infinite) set $\Delta(q, a)$ of *moves* available at state q to player a.

  For $A \subseteq \Sigma$ and $q \in Q$, an (A, q)*-move* is a function c which maps every $a \in A$ to a move $c(a) \in \Delta(q, a)$. Given $A \subseteq \Sigma$ and a state q, we write $\Delta^A(q)$ for the set of (A, q)-moves. An (A, q)-move is called a *partial move* if $A \neq \Sigma$, and a *total move* if $A = \Sigma$.
- δ is a *transition function* which, for each state q and each total move $c \in \Delta^{\Sigma}(q)$, returns a state $\delta(q, c) \in Q$ (the state obtained when in state q all players simultaneously perform their moves according to c).

A *computation* of S is an infinite sequence $\lambda = q_0, q_1, \ldots$ of states such that for each $i \geq 0$, the state q_{i+1} is a *successor* of q_i, i.e., $q_{i+1} = \delta(q_i, c)$ for some total move $c \in \Delta^{\Sigma}(q_i)$. We call λ a q-computation if $q_0 = q$. We refer to the ith state q_i in λ by $\lambda[i]$, to the sequence $q_i, q_{i+1}, \ldots, q_j$ by $\lambda[i, j]$, and to the sequence $q_i, q_{i+1}, \ldots$ by $\lambda[i, \infty]$.

Let $c \in \Delta^A(q)$ and $c' \in \Delta^{A'}(q)$ for $A, A' \subseteq \Sigma$ and $q \in Q$ with $A \subseteq A'$. We write $c \sqsubseteq c'$ if $c(a) = c'(a)$ for every $a \in A$. For a state q, a set of players $A \subseteq \Sigma$, and an (A, q)-move $c \in \Delta^A(q)$, we say that a state $q' \in Q$ is a *c–successor of* q if there is a total move $c' \in \Delta^{\Sigma}(q)$ with $c \sqsubseteq c'$ and $q' = \delta(q, c')$.

4.1.2 AMC

Following [AHK97, AHK02], we now recall the definition of the alternating μ-calculus (AMC).

Syntax of AMC-Formulas

An AMC-formula over the set $\mathbb{P}$ of propositions, the set $\mathcal{V}$ of variables, and the set Σ of players is one of the following:

- $p \in \mathbb{P}$,
- $X \in \mathcal{V}$,
- $\neg\varphi$ if φ is an AMC-formula,
- $\varphi_1 \vee \varphi_2$ if φ_1 and φ_2 are AMC-formulas,
- $\langle\!\langle A \rangle\!\rangle\varphi$ if $A \subseteq \Sigma$ and φ is an AMC-formula,
- $\mu X.\varphi$ if φ is an AMC-formula and all free occurrences of X (i.e., those that do not occur in a subformula of φ starting with μX) fall under an even number of negations.

We use the following common abbreviations: $[\![A]\!]\varphi = \neg\langle\!\langle A \rangle\!\rangle\neg\varphi$, $\varphi \wedge \psi = \neg(\neg\varphi \vee \neg\psi)$, and $\nu X.\varphi = \neg\mu X.\neg\varphi[X/\neg X]$ where $\varphi[X/\neg X]$ is obtained from φ by replacing every free occurrence of X in φ by $\neg X$ and vice versa. Using these abbreviations we can write every AMC-formula in negation normal form (also called positive normal form). An AMC-formula is in *negation normal form* if every negation symbol only occurs immediately in front of a proposition or a variable.

An AMC-formula is a *sentence* if it does not contain free variables, i.e., all variables are bounded by a fixed-point operator.

The *size* of an AMC-formula φ, denoted $|\varphi|$, is defined inductively in the obvious way.

Semantics of AMC-Formulas

To define the semantics of AMC-formulas, we first need some definitions and notations. Given a game structure $S = \langle \Sigma, Q, \mathbb{P}, \pi, \Delta, \delta \rangle$, a valuation F is a function from the set of variables $\mathcal{V}$ to 2^Q, i.e., subsets of Q. For F, a variable X, and a set $M \subseteq Q$, we denote by $F[X := M]$ the valuation that maps X to M and agrees with F on all other variables.

An AMC-formula φ is interpreted as a mapping φ^S from valuations to state sets. Intuitively, $\varphi^S(F)$ denotes the set of states in which φ is satisfied under the valuation F in the structure S. The mapping φ^S is defined inductively as follows:

- $p^S(F) = \{q \in Q \mid p \in \pi(q)\}$ for $p \in \mathbb{P}$.
- $X^S(F) = F(X)$ for $X \in \mathcal{V}$.
- $(\neg\varphi)^S(F) = Q \setminus \varphi^S(F)$.
- $(\varphi_1 \vee \varphi_2)^S(F) = \varphi_1^S(F) \cup \varphi_2^S(F)$.
- $(\langle\!\langle A \rangle\!\rangle \varphi)^S(F) = \{q \in Q \mid$ there exists $c \in \Delta^A(q)$ such that for every $c' \in \Delta^\Sigma(q)$ with $c \sqsubseteq c'$ we have $\delta(q, c') \in \varphi^S(F)\}$.
- $(\mu X.\varphi)^S(F) = \bigcap\{M \subseteq Q | \varphi^S(F[X := M]) \subseteq M\}$, i.e., $(\mu X.\varphi)^S(F)$ is the least fixed-point of the function that maps $M \subseteq Q$ to $\varphi^S(F[X := M])$. (Note that this function is monotonic.)

Note that if φ is a sentence, then the interpretation of φ in the structure S is uniquely determined independently of a valuation function F. In fact, $\varphi^S(F) = \varphi^S(F')$ for all valuation functions F and F', i.e., φ^S is a constant mapping. We therefore simply write φ^S instead of $\varphi^S(F)$ for some F.

Given a state q of a CGS S and a sentence φ, we write

$$(S, q) \models \varphi$$

if $q \in \varphi^S$.

Deciding $(S, q) \models \varphi$ for a given finitely represented CGS S, a state q in S, and a sentence φ is an AMC-model checking problem. The main purpose of this chapter is to study this problem for a class of CGSs induced by cryptographic protocols.

We note that AMC is more expressive than ATL* (and hence, ATL).

Theorem 4.1.1 *[AHK02] AMC is more expressive than ATL*, and hence, provided a suitable set of propositional variables, also more expressive than fair ATL. The alternation-free fragment of AMC is more expressive than ATL.*

4.1.3 Parity Games and AMC-Model Checking

In this section, we first recall the definition of parity games and then, similar to the case of modal μ-calculus, associate with every CGS S, state q, and AMC-sentence φ a parity game in which player 0 has a winning strategy iff $(S, q) \models \varphi$.

Parity Games

Following [GTW02], we now recall the definition of parity games.

A *parity game* G is a tuple (V, V_0, V_1, E, v_I, l) where V is a (possibly infinite) set of vertices partitioned into sets V_0 and V_1 (i.e., $V_0 \cup V_1 = V$ and $V_0 \cap V_1 = \emptyset$), $v_I \in V$ (the initial vertex), $E \subseteq V \times V$ is a set of edges, and l is a coloring function from V into the set of colors $\{0, \dots, m\}$, for some natural number m, such that (V, E) is a directed, leafless graph.

The parity game G is played by two players, player 0 and 1. A play of G starts by putting a token on vertex v_I. Now, if in a play a token is put on a vertex v (initially $v = v_I$) with $v \in V_i$ for $i \in \{0, 1\}$, then player i chooses a successor v' of v, i.e., $(v, v') \in E$, and moves the token to v'. Then, if $v' \in V_j$, for $j \in \{0, 1\}$, it is player j's turn to move the token to a successor of v', and so on. This continues forever. (Note that by definition of parity games, every vertex has a successor.) Formally, a *play* p is an infinite sequence $v_0, v_1, v_2, \dots$, of vertices such that $v_0 = v_I$ and $(v_i, v_{i+1}) \in E$ for every $i \geq 0$. The play p is *winning for player 0* if the maximum color occurring infinitely often in p, i.e., the color $\max\{k \mid k$ occurs infinitely often in the sequence $l(v_0), l(v_1), \dots\}$, is even. Otherwise, the play is *winning for player 1*.

A *strategy* f of player i is a function that for every finite prefix of a play, ending in a vertex $v \in V_i$, selects a successor v' of v, i.e., $(v, v') \in E$. A play $v_0, v_1, \dots$ is *consistent with* f, if for each n such that $v_n \in V_i$, we have $v_{n+1} = f(v_0, v_1, \dots, v_n)$. A strategy of player i is *winning*, if each play consistent with this strategy is winning for player i.

A strategy is *memoryless* (or *positional*), if it depends only on the last vertex, i.e., if $v_0, v_1, \dots, v_n$ and $v'_0, v'_1, \dots, v'_{n'}$ are prefixes of plays with $v_n = v'_{n'}$, then $f(v_0, v_1, \dots, v_n) = f(v'_0, v'_1, \dots, v'_{n'})$. We therefore often represent a memoryless strategy of player i by a function from V_i to V such that, for each $v \in V_i$, if $f(v) = v'$, then $(v, v') \in E$. A memoryless strategy f of player i in a parity game G induces a subgraph of (V, E) where all outgoing edges of vertices $v \in V_i$ are deleted except for the edge to $f(v)$. We call this graph a *strategy graph of player i* or the *strategy graph of player i induced by f*. Obviously, f is a winning strategy for player i iff all infinite paths in the induced strategy graph starting from the initial vertex v_I are winning for player i. We will sometimes assume that strategy graphs contain only vertices reachable from the initial vertex. Obviously, if such a graph is winning for player i, then each vertex v in this graph is winning for player i in the sense that each infinite path in this graph starting in v is winning for player i.

We summarize well-known and fundamental facts about parity games.

Fact 4.1.2 *[Mar75, Mos91, EJ91] (see also [Zie98]) Parity games are determined, i.e., either player* 0 *or player* 1 *has a winning strategy. The player who has a winning strategy has a memoryless one.*

Parity Games for AMC-Model Checking

In this section, we associate with every CGS S, state q_0 of S, and AMC-sentence φ in negation normal form a parity game $G^{\varphi}_{(S,q_0)}$ such that player 0 wins $G^{\varphi}_{(S,q_0)}$ iff $(S, q_0) \models \varphi$. Our construction follows that of modal μ-calculus (see, e.g., [Wil01, GTW02]) and is similar to the one in [SF06]. However, instead of first turning φ into an equivalent alternating parity tree automaton and then using this tree automaton to obtain the parity game, we construct the parity game directly from φ and S.

Throughout the rest of this section, let $S = \langle \Sigma, Q, \mathbb{P}, \pi, \Delta, \delta \rangle$ be a concurrent game structure, $q_0 \in Q$ be a state in S, and φ be an AMC-sentence in negation normal form. We assume, w.l.o.g., that for each variable X in φ there is exactly one subformula of the form $\mu X.\psi$ or $\nu X.\psi$ in φ. (This can obviously be guaranteed by renaming variables.) We refer to this subformula by φ^X.

In what follows, we refer to subformulas of φ by *standard subformulas*. Now, given S, $q \in Q$, and φ, we define the set $\mathrm{Sub}^q_S(\varphi)$ to consist of the following (standard and non-standard) subformulas of φ:

(a) ψ for every standard subformula ψ of φ,

(b) $\Box_c\psi$ for every standard subformula $\langle\!\langle A\rangle\!\rangle\psi$ of φ and $c \in \Delta^A(q)$,

(c) $\Diamond_c\psi$ for every standard subformula $[\![A]\!]\psi$ of φ and $c \in \Delta^A(q)$.

We will call elements of $\mathrm{Sub}^q_S(\varphi)$ *subformulas of* φ where, as mentioned, the formulas in (a) are called *standard subformulas of* φ and those in (b) and (c) are called *nonstandard subformulas of* φ. Note that $\Diamond_c$ and $\Box_c$ occur only as top symbols of subformulas; they are not nested.

Now, the parity game $G^{\varphi}_{(S,q_0)} = (V, V_0, V_1, E, v_I, l)$ for S, q_0, and φ is defined as follows (see below for a brief discussion of the differences to the construction for the model μ-calculus): The set V of vertices consists of all tuples of the form (q, ψ) where $q \in Q$ and $\psi \in \mathrm{Sub}^q_S(\varphi)$. The initial vertex v_I is (q_0, φ). The set V_0 consists of vertices of the form (q, ψ) where $q \in Q$ and ψ is of one of the following forms:

$$\psi' \vee \psi'', \qquad \langle\!\langle A \rangle\!\rangle \psi', \qquad \Diamond_c \psi'.$$

All remaining vertices belong to V_1. The set E of edges is the smallest set satisfying the following conditions:

- $(\ (q,p)\ ,\ (q,p)\ ) \in E$ for every $(q,p) \in V$.
- $(\ (q,\neg p)\ ,\ (q,\neg p)\ ) \in E$ for every $(q,\neg p) \in V$.
- $(\ (q,X)\ ,\ (q,\varphi^X)\ ) \in E$ for every $(q,X) \in V$.
- $(\ (q,\mu X.\psi)\ ,\ (q,\psi)\ ) \in E$ for every $(q,\mu X.\psi) \in V$.
- $(\ (q,\nu X.\psi)\ ,\ (q,\psi)\ ) \in E$ for every $(q,\nu X.\psi) \in V$.
- $(\ (q,(\psi \vee \psi'))\ ,\ (q,\psi)\ ) \in E$ and $(\ (q,(\psi \vee \psi'))\ ,\ (q,\psi')\ ) \in E$ for every $(q,(\psi \vee \psi')) \in V$.
- $(\ (q,(\psi \wedge \psi'))\ ,\ (q,\psi)\ ) \in E$ and $(\ (q,(\psi \wedge \psi'))\ ,\ (q,\psi')\ ) \in E$ for every $(q,(\psi \wedge \psi')) \in V$.
- $(\ (q,\langle\!\langle A \rangle\!\rangle\psi)\ ,\ (q,\Box_c\psi)\ ) \in E$ for every $(q,\langle\!\langle A \rangle\!\rangle\psi) \in V$ and $c \in \Delta^A(q)$.
- $(\ (q,[\![A]\!]\psi)\ ,\ (q,\Diamond_c\psi)\ ) \in E$ for every $(q,[\![A]\!]\psi) \in V$ and $c \in \Delta^A(q)$.
- $(\ (q,\Box_c\psi)\ ,\ (q',\psi)\ ) \in E$ for every $(q,\Box_c\psi) \in V$ and c-successor q' of q.
- $(\ (q,\Diamond_c\psi)\ ,\ (q',\psi)\ ) \in E$ for every $(q,\Diamond_c\psi) \in V$ and c-successor q' of q.

Let $\|\varphi\|$ be the depth of φ when φ is viewed as a syntax tree. The coloring function l is defined as follows:[1] The color of a state $s = (q, \psi)$ is defined as follows:

- $l(s) = 1$, if $\psi = p$ and $p \notin \pi(q)$ or $\psi = \neg p$ and $p \in \pi(q)$,
- $l(s) = 2\|\psi\|$, if $\psi = \nu X.\psi'$ for some ψ',
- $l(s) = 2\|\psi\| + 1$, if $\psi = \mu X.\psi'$ for some ψ', and

[1]Using the alternation depth of formulas, one can obtain a coloring function that assigns smaller colors. This is useful to achieve more efficient algorithms. However, for the complexity results shown in this chapter the coloring function employed here is sufficient.

- $l(s) = 0$ otherwise.

The above definition of $G^{\varphi}_{(S,q_0)}$ is similar to the case of modal μ-calculus with the following difference: In case of the modal μ-calculus, when a play reaches a position (q, ψ) with ψ of the form $\Box\psi'$ or $\Diamond\psi'$, then one of the players chooses a successor q' of q and the play continues with (q', ψ). In case of AMC, we have a family of modal operators $\langle\langle A\rangle\rangle$ and $[\![A]\!]$ and when a play reaches a position $(q, \langle\langle A\rangle\rangle\psi')$ or $(q, [\![A]\!]\psi')$ then we use an intermediate state before a successor of q is chosen: first one of the players moves to a position of the form $(q, \Box_c\psi')$ or $(q, \Diamond_c\psi')$, and then the opponent chooses a successor q' of q and the play continues with (q', ψ').

Similar to the case of the modal μ-calculus (see, e.g., [Wil01, GTW02]), one shows the following proposition:

Proposition 4.1.3 *For S, q_0, and φ as above we have that $(S, q_0) \models \varphi$ iff player 0 has a (memoryless) winning strategy in the parity game $G^{\varphi}_{(S,q_0)}$.*

4.2 Our Protocol and Intruder Model

We now introduce our protocol and intruder model. Similar to [KKW06], we consider a real concurrent communication model in which principals (including the intruder) may take actions at the same time and may receive/send several messages at the same time from/to different principals. Principals are connected via different kinds of channels: network and resilient channels. Instead of the term "resilient channel", we often use the term "secure channel". While network channels are completely controlled by the intruder, secure channels are not. In particular, the intruder may not be able to delay or modify messages sent over such a channel. We will consider two types of secure channels. Those that directly link to principals (direct secure channels) and those that are buffered (scheduled secure channels). While messages sent over direct secure channels are immediately delivered, messages sent over scheduled secure channels are first written into a buffer. The buffer is an independent agent which can follow its own strategy in delivering messages; it can for example team up with an honest principal or the intruder. Whether and with whom such a buffer collaborates depends on the security property considered, as specified by an AMC-formula.

While conceptually the model presented here and the one presented in [KKW06] are quite similar, the presentation and level of detail varies in the following main points: First, in [KKW06], no specific formalism for describing honest protocol participants was presented. Such participants could be arbitrary I/O components (in particular, arbitrary interactive, non-deterministic Turing machines). However, since in the present work we are interested in decidability results, we need to be more precise about the representation and the kind of computation honest protocol participants are allowed to perform. As defined below, such participants will be modeled by certain edge-labeled trees. Second, in [KKW06] we considered a general communication model and described how a system of I/O components runs. Then, protocol runs and attacks were described in terms of such systems where every entity (honest protocol participants, the intruder, scheduled secure channels) was modeled as an I/O component. In the present work, we do not consider systems of I/O components but model protocol runs and attacks directly in terms of concurrent game structures.

In what follows, we define i) terms and messages, ii) how the intruder can derive new messages from a given set of messages, iii) principals and protocols, and iv) concurrent game structures which describe the run of a protocol along with the intruder.

4.2.1 Terms and Messages

The set $\mathcal{T}$ of *terms* is defined by the core grammar given in Chapter 2:

We define $\mathcal{T}_\circ = \mathcal{T} \cup \{\circ\}$ and $\mathcal{M}_\circ = \mathcal{M} \cup \{\circ\}$ where '$\circ$' is a new symbol which stands for 'no message'. This symbol will be used in case there is no message on a channel.

4.2.2 Derivation of Messages

Given a set $\mathcal{K}$ of messages, the (infinite) set $d(\mathcal{K})$ of messages the intruder can derive from $\mathcal{K}$ is the smallest set satisfying the following conditions with $m, m' \in \mathcal{M}$:

1. $\mathcal{K} \subseteq d(\mathcal{K})$.
2. $\circ \in d(\mathcal{K})$.

3. *Composition and decomposition*: If $m, m' \in d(\mathcal{K})$, then $\langle m, m' \rangle \in d(\mathcal{K})$. Conversely, if $\langle m, m' \rangle \in d(\mathcal{K})$, then $m \in d(\mathcal{K})$ and $m' \in d(\mathcal{K})$.
4. *Symmetric encryption and decryption*: If $m, m' \in d(\mathcal{K})$, then $\{m\}^s_{m'} \in d(\mathcal{K})$. Conversely, if $\{m\}^s_{m'} \in d(\mathcal{K})$ and $m' \in d(\mathcal{K})$, then $m \in d(\mathcal{K})$.
5. *Asymmetric encryption and decryption*: If $m \in d(\mathcal{K})$ and $k \in d(\mathcal{K}) \cap \mathbb{K}_{pub}$, then $\{m\}^a_k \in d(\mathcal{K})$. Conversely, if $\{m\}^a_k \in d(\mathcal{K})$ and $k^{-1} \in d(\mathcal{K}) \cap \mathbb{K}_{priv}$, then $m \in d(\mathcal{K})$.
6. *Hashing*: If $m \in d(\mathcal{K})$, then $\mathsf{hash}(m) \in d(\mathcal{K})$.
7. *Signing*: If $m \in d(\mathcal{K}), k^{-1} \in d(\mathcal{K}) \cap \mathbb{K}_{priv}$, then $\mathsf{sig}(k, m) \in \mathcal{K}$. (The signature contains the public key but can only be generated if the corresponding private key is known.)
8. *Generating fresh constants*: $\mathcal{A}_I \subseteq d(\mathcal{K})$.

4.2.3 Channels, Principals, and Protocols

We denote by $\mathcal{P}$ the finite set of all principals. This set is partitioned into the set $\mathcal{H}$ of *honest* and the set $\mathcal{D}$ of *dishonest* principals. All dishonest principals will be subsumed by the intruder. The behavior of honest principals will be specified by certain trees (see below). Protocols will basically be defined by a set of such trees, specifying the behavior of all honest principals participating in a protocol run. First, we have to define how principals are connected via channels.

Channels and Multi Terms

We consider three types of communication channels between principals (including the intruder): (1) *network channels*, (2) *direct secure channels*, and (3) *scheduled secure channels*. Network channels are controlled by the intruder, i.e., every message sent on a network channel by an honest principal is immediately delivered to the intruder and every message received from a network channel was sent by the intruder (who impersonates some honest or dishonest principal). A direct secure channel is a direct link between principals, i.e., every message sent on such a channel by some principal to another principal will immediately be delivered to the latter principal without intervention by the intruder. Messages sent via a scheduled secure channel will first be sent to a buffer before they are delivered to the intended recipient. Such a buffer is an independent player in the concurrent games structures that we consider and may be controlled or may team up with (honest or dishonest) principals or other scheduled secure channels. This will be specified by AMC-formulas.

A network channel from a principal a to a principal b such that $a \neq b$ and not both a and b are dishonest will be denoted by $\mathsf{net}(a, b)$. Similarly, we use $\mathsf{dir}(a, b)$ and $\mathsf{sch}(a, b)$ to refer to direct and scheduled secure channels from a to b, respectively. The set of all the channels will be denoted by $\mathcal{C}$.

For sets $A, B \subseteq \mathcal{P}$ of principals, we define $\mathsf{Net}(A, B) = \{\mathsf{net}(a, b) \mid a \in A, b \in B, a \neq b, \text{ and } (a \in \mathcal{H} \text{ or } b \in \mathcal{H})\}$. Similarly, we define $\mathsf{Dir}(A, B)$ and $\mathsf{Sch}(A, B)$ for direct and scheduled secure channels. We define $\mathcal{C}(A, B) = \mathsf{Net}(A, B) \cup \mathsf{Dir}(A, B) \cup \mathsf{Sch}(A, B)$. We will write, for example, $\mathsf{Net}(a, B)$ instead of $\mathsf{Net}(\{a\}, B)$.

For a set $C \subseteq \mathcal{C}$, we call a mapping $\mathbf{r} : C \to \mathcal{T}_\circ$ a *multi term* and a mapping $\mathbf{m} : C \to \mathcal{M}_\circ$ a *multi message*. We denote by $\mathsf{ch}(\mathbf{m})$ and $\mathsf{ch}(\mathbf{r})$ the domain C of $\mathbf{m}$ and $\mathbf{r}$, respectively, and by $\mathcal{V}(\mathbf{r})$ the set of variables occurring in the range of $\mathbf{r}$, i.e., in the set $\{t \mid \mathbf{r}(c) = t \text{ for some } c \in C\}$. If σ is a substitution, we denote by $\mathbf{r}\sigma$ the multi term obtained by substituting every variable $x \in \mathcal{V}(\mathbf{r})$ occurring in $\mathbf{r}$ by $\sigma(x)$, i.e., $\mathbf{r}\sigma(c) = \mathbf{r}(c)\sigma$ for every $c \in C$.

Let $\mathbf{m}$ be a multi message, $\mathbf{r}$ be a multi term, and σ be a substitution with domain $\mathcal{V}(\mathbf{r})$. We say that $\mathbf{m}$ *matches with* $\boldsymbol{r}$ *by* σ, if $\mathsf{ch}(\mathbf{r}) \subseteq \mathsf{ch}(\mathbf{m})$ and $\mathbf{m}(c) = \mathbf{r}(c)\sigma$ for each $c \in \mathsf{ch}(r)$. We say that $\mathbf{m}$ *matches with* $\boldsymbol{r}$, if there is a substitution σ such that $\mathbf{m}$ matches with $\mathbf{r}$ by σ.

Honest Principals

We now define honest principals; more precisely, we should say 'instances of honest principals' since a principal might run several copies of a protocol in possibly different roles. Informally speaking, an honest principal is defined by a finite edge-labeled tree which describes the behavior of this principal in a protocol run. Each edge of such a tree is labeled by a rule which describes the receive-send action that is performed when the principal takes this edge in a run of the protocol. As mentioned above, in *one* receive-send action a principal may receive/send several messages on different channels. The trees that we consider may have self-loops. These allow a principal to stay in the same state. When a principal carries out a protocol, it traverses its tree, starting at the root. In every node, the principal takes its current input (on all channels the principal has access to or wants to read), chooses one of the edges leaving the node such that the current inputs match with the left-hand side of the rule the edge is labeled with, sends out (possibly different) messages on (possibly different) channels as determined by the right-hand side of the rule, and moves to the node the chosen edge leads to. Edges have priorities which influence which edge may be taken in case several edges are applicable. However, if several edges with the same priority can be taken, one such edge is picked non-deterministically. Formally, principals are defined as follows:

For sets $C, D \subseteq \mathcal{C}$, we call $\mathbf{r} \Rightarrow \mathbf{s}$ with $\mathbf{r} : C \to \mathcal{T}_\circ$ and $\mathbf{s} : D \to \mathcal{T}_\circ$ a *(C, D)–rule*. For an honest principal $a \in \mathcal{H}$, an *a–rule* is a (C, D)–rule with $C \subseteq \mathcal{C}(\mathcal{P}, a)$ and $D \subseteq \mathcal{C}(a, \mathcal{P})$. If σ is a substitution and $R = (\mathbf{r} \Rightarrow \mathbf{s})$ is a rule, we write $R\sigma$ to denote the rule obtained by substituting every variable x occurring in R by $\sigma(x)$, i.e., $R\sigma = (\mathbf{r}\sigma \Rightarrow \mathbf{s}\sigma)$.

Let $a \in \mathcal{H}$ be an honest principal. Its behavior is specified by what we call an a-instance (or simply principal). An *a-instance* (*principal*) is defined by a finite tree $P = (V, E, r, \ell_p, \ell)$ where V is the set of vertices, E is the set of edges, $r \in V$ is the root of the tree, and ℓ_p maps every edge $e \in E$ of P to a natural number, the *priority of this edge*. The labeling function ℓ maps every edge $e = (v, v') \in E$ of P to an a-rule $\ell(e)$ in such a way that every variable occurring in $\mathcal{V}(\mathbf{s})$ with $\ell(e) = (\mathbf{r} \Rightarrow \mathbf{s})$ also occurs on the left-hand side of $\ell(e)$, i.e., in $\mathcal{V}(\mathbf{r})$, or on the left-hand side of a rule on the path from the root r to v. In other words, every variable occurring on the right-hand side of a rule also occurs on the left-hand side of this or a preceding rule. Nodes of P may have *self-loops*, i.e., P may contain edges of the form $e = (v, v)$ for $v \in V$. In that case, we require that for $\ell(e) = (\mathbf{r} \Rightarrow \mathbf{s})$ the domains of $\mathbf{r}$ and $\mathbf{s}$ are empty, i.e., $\mathsf{ch}(\mathbf{r}) = \emptyset$ and $\mathsf{ch}(\mathbf{s}) = \emptyset$. In other words, when performing a self-loop, a principal neither reads nor writes messages from/onto a channel.

For an a-instance P, we denote by $\mathsf{ch}(P)$ the set of all the channels used by P, i.e., $\mathsf{ch}(P)$ consists of those channels c for which there exists an edge in P labeled with a rule of the form $\mathbf{r} \Rightarrow \mathbf{s}$ such that $c \in \mathsf{ch}(\mathbf{r})$ or $c \in \mathsf{ch}(\mathbf{s})$.

Protocols

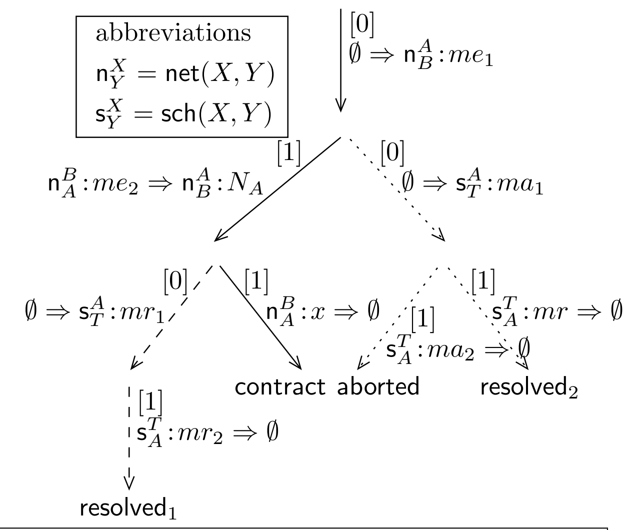
A *protocol* is a tuple $Pr = (\mathcal{H}, \mathcal{D}, \mathcal{K}, \{P_a\}_{a\in\mathcal{H}})$ where $\mathcal{H}$ and $\mathcal{D}$ are sets of honest and dishonest principals, respectively, P_a is an a-instance for each $a \in \mathcal{H}$, and $\mathcal{K}$ is the *initial intruder knowledge*, i.e., a finite set of messages. W.l.o.g., we assume that the set of vertices of the trees P_a, $a \in \mathcal{H}$, are pairwise disjoint. For a protocol Pr, we denote by $\mathsf{ch}(Pr)$ the set of channels used in Pr, i.e. the set of all channels c such that $c \in \mathsf{ch}(P_a)$, for some $a \in \mathcal{H}$. The size of Pr, denoted by $|Pr|$ is defined according to some standard representation of Pr.

4.2.4 Example: The ASW Protocol

Figure 4.1 and 4.2 present the formal specifications P_A and P_T of A and T, respectively. The specification of B can be defined similarly. The specification of the ASW protocol for the case that A and T are honest but B is dishonest (and hence, B's behavior is determined by the intruder) is the tuple $Pr_{ASW} = (\{A, T\}, \{B\}, \{A, B, T, k_A, k_B, k_B^{-1}, k_T\}, \{P_A, P_T\})$. In Figure 4.1 and 4.2 the communication between A and T is modeled by scheduled secure channels and the communication between B and T by direct secure channels. Note that allowing (dishonest) B direct communication with T increases his power. To check certain properties of this protocol, it is useful to add a "watch dog" W as another honest principal: W checks whether the intruder (B) has a standard or replacement contract (as defined in Chapter 2). More precisely, W waits to receive the standard or replacement contract (as defined in Chapter 2) on a network channel from B and if it receives such a contract, moves to a vertex called, say `Bhascont`; W ignores all other messages it receives.

4.2.5 The Concurrent Game Structure Induced by a Protocol

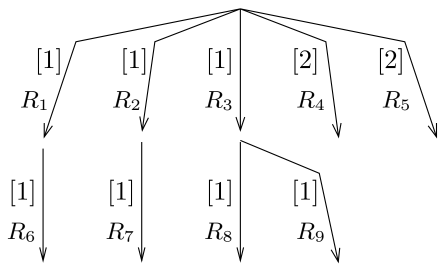
We now introduce the concurrent game structure induced by a protocol. The players involved are the honest principals, the scheduled secure channels, and the Dolev-Yao intruder (who subsumes the dishonest principals). The concurrent game structure describes what moves these players can take in


abbreviations
$\mathsf{n}_Y^X = \mathsf{net}(X,Y)$
$\mathsf{s}_Y^X = \mathsf{sch}(X,Y)$
[0]
$\emptyset \Rightarrow \mathsf{n}_B^A : me_1$
[1]
$\mathsf{n}_A^B : me_2 \Rightarrow \mathsf{n}_B^A : N_A$
[0]
$\emptyset \Rightarrow \mathsf{s}_T^A : ma_1$
[0]
$\emptyset \Rightarrow \mathsf{s}_T^A : mr_1$
[1]
$\mathsf{n}_A^B : x \Rightarrow \emptyset$
[1]
$\mathsf{s}_A^T : ma_2 \Rightarrow \emptyset$
[1]
$\mathsf{s}_A^T : mr \Rightarrow \emptyset$
contract
aborted
resolved$_2$
[1]
$\mathsf{s}_A^T : mr_2 \Rightarrow \emptyset$
resolved$_1$

me_1	$=$	$\mathsf{sig}[k_A, \langle k_A, k_B, k_T, \mathsf{contract}, \mathsf{hash}(N_A)\rangle]$
me_2	$=$	$\mathsf{sig}[k_B, \langle me_1, \mathsf{hash}(x)\rangle]$
ma_1	$=$	$\mathsf{sig}[k_A, \langle \mathsf{abort}, me_1\rangle]$
ma_2	$=$	$\mathsf{sig}[k_T, \langle \mathsf{abort}, ma_1\rangle]$
mr_1	$=$	$\langle me_1, me_2\rangle$
mr_2	$=$	$\mathsf{sig}[k_T, mr_1]$
mr	$=$	$\mathsf{sig}[k_T, \langle me_1, \mathsf{sig}[k_B, \langle me_1, y\rangle]\rangle]$

Figure 4.1: Honest Alice running the ASW protocol as initiator with Bob. Even though not drawn, every vertex in the tree has a self-loop with priority 0.

every state and what effect these moves have. Roughly speaking, the possible moves of an honest principal are those edges (receive-send actions) that leave the current vertex and that can be applied given the current input and the priority on the edges. As a result of taking such an edge, the principal writes output on (zero, one, or more) channels. A scheduled secure channel is represented by a sequence of messages, the messages on this channel. In a move it decides whether or not to deliver the first message in this sequence. (Alternatively, in case one would like to model secure channels that do not preserve the order of messages, one could allow secure channels to pick one of the messages in their sequence.) The intruder has an infinite number of possible moves. He can write a message on all channels that he controls, where for every such channel he can pick one of the (infinitely many possible) messages that he can derive in the current state. Direct secure channels will



rules:

R_1 :	$\mathsf{sch}(A,T) : ma^a$	$\Rightarrow$	$\mathsf{sch}(T,A) : mak^a$
R_2 :	$\mathsf{sch}(A,T) : mr^b$	$\Rightarrow$	$\mathsf{sch}(T,A) : mrc^b$
R_3 :	$\mathsf{dir}(B,T) : mr^c$	$\Rightarrow$	$\mathsf{dir}(T,B) : mrc^c$
R_4 :	$\mathsf{sch}(A,T) : mr^d$ $\mathsf{dir}(B,T) : mr^d$	$\Rightarrow$	$\mathsf{sch}(T,A) : mrc^d$ $\mathsf{dir}(T,B) : mrc^d$
R_5 :	$\mathsf{sch}(A,T) : ma^e$ $\mathsf{dir}(B,T) : mr^e$	$\Rightarrow$	$\mathsf{sch}(T,A) : mak^e$ $\mathsf{dir}(T,B) : mak^e$
R_6 :	$\mathsf{dir}(B,T) : mr^a$	$\Rightarrow$	$\mathsf{dir}(T,B) : mak^a$
R_7 :	$\mathsf{dir}(B,T) : mr^b$	$\Rightarrow$	$\mathsf{dir}(T,B) : mrc^b$
R_8 :	$\mathsf{sch}(A,T) : mr^c$	$\Rightarrow$	$\mathsf{sch}(T,A) : mrc^c$
R_9 :	$\mathsf{sch}(A,T) : ma^c$	$\Rightarrow$	$\mathsf{sch}(T,A) : mrc^c$

messages:

$mr^u = \langle me_1^u, me_2^u \rangle$
$me_1^u = \mathsf{sig}[k_A, \langle k_A, k_B, k_T, \mathsf{contract}, x_u \rangle]$
$me_2^u = \mathsf{sig}[k_B, \langle me_1^u, y_u \rangle]$
$ma^u = \mathsf{sig}[k_A, \langle \mathsf{abort}, me_1^u \rangle]$
$mak^u = \mathsf{sig}[k_T, \langle \mathsf{abort}, me_1^u \rangle]$
$mrc^u = \mathsf{sig}[k_T, \langle me_1^u, me_2^u \rangle]$

Figure 4.2: This is the tree model of participant TTP. Even though not drawn, every vertex in the tree has a self-loop with priority 0. The rules R_1 to R_5 are needed to distinguish between whether the TTP receives in one step (1) one abort message from A, (2) one resolve message from A, (3) one resolve message from B, (4) a resolve message from both A and B, or (5) an abort message from A and a resolve message from B.

always immediately deliver the message written to them, and therefore, they do not need to be modeled as players. Before providing the formal definition of the concurrent game structure, we need to introduce some notation.

For a principal P and a node v of P, we write $P{\downarrow}v$ to denote the subtree of P rooted at v. If σ is a substitution, we write $P\sigma$ for the principal obtained from P by applying σ to every rule occurring in P.

For a protocol Pr, let $C = \mathcal{C}(\mathcal{P}, a) \cap \mathsf{ch}(Pr)$ and $C' = \mathcal{C}(a, \mathcal{P}) \cap \mathsf{ch}(Pr)$. For $\mathbf{m} : C \to \mathcal{M}_\circ$, $\mathbf{m}' : C' \to \mathcal{M}_\circ$, an a-instance $P = (V, E, v_0, \ell_p, \ell)$, and an a-instance P', we write $(\mathbf{m}, P) \overset{v}{\mapsto} (\mathbf{m}', P')$ if $v \in V$, $(v_0, v) \in E$, $\ell(v_0, v)$ is of the form $\mathbf{r} \Rightarrow \mathbf{s}$, and there exists a substitution σ with $\mathrm{dom}(\sigma) = \mathcal{V}(\mathbf{r})$ such that

- $P' = (P \downarrow v)\sigma$,
- $\mathbf{m}$ matches with $\mathbf{r}$ by σ,

- for all $v' \in V$ such that $(v_0, v') \in E$, $\ell(v_0, v') = (\mathbf{r}' \Rightarrow \mathbf{s}')$ and $\mathbf{m}$ matches with $\mathbf{r}'$ we have that $\ell_p(v_0, v) \geq \ell_p(v_0, v')$, and
- $\mathbf{m}'$ matches with $\mathbf{s}$ by σ and $\mathbf{m}'(c) = \circ$ for each $c \in C' \setminus \mathsf{ch}(\mathbf{s})$.

In what follows, let $Pr = (\mathcal{H}, \mathcal{D}, \mathcal{K}_0, \{P_a^0\}_{a \in \mathcal{H}})$ be a protocol. Let $m \in \mathcal{M}_\circ$ and $t \in \mathcal{T}_\circ$. Let $\mathcal{IC} = (\mathsf{Net}(\mathcal{P}, \mathcal{H}) \cup \mathsf{Dir}(\mathcal{D}, \mathcal{H}) \cup \mathsf{Sch}(\mathcal{D}, \mathcal{H})) \cap \mathsf{ch}(Pr)$ be the set of all channels in Pr the intruder may write to and let $\mathcal{SC} = \mathsf{Sch}(\mathcal{P}, \mathcal{P}) \cap \mathsf{ch}(Pr)$ be the set of all scheduled secure channels in Pr.

We are now ready to define the concurrent game structure induced by Pr. The *concurrent game structure* $S = S_{Pr} = \langle \Sigma, Q, \mathbb{P}, \pi, \Delta, \delta \rangle$ *induced by* Pr is defined as follows:

- The set of players Σ is $\mathcal{H} \cup \mathcal{SC} \cup \{\mathcal{I}\}$.
- The set of states Q consists of tuples of the form $(\mathcal{K}, \overline{P}, \overline{\mathbf{m}}, \overline{s})$ where
  - $\mathcal{K}$ is a finite set of messages (the current intruder knowledge),
  - $\overline{P}$ is a family $\{P_a\}_{a \in \mathcal{H}}$ of a-instances P_a for every $a \in \mathcal{H}$,
  - $\overline{\mathbf{m}}$ is a family $\{\mathbf{m}_a\}_{a \in \mathcal{H}}$ of multi messages $\mathbf{m}_a : \mathcal{C}(\mathcal{P}, a) \cap \mathsf{ch}(Pr) \to \mathcal{M}_\circ$ for every $a \in \mathcal{H}$ (the current input to a).[2]
  - $\overline{s}$ is a family $\{(s_c, d_c)\}_{c \in \mathcal{SC}}$ of tuples (s_c, d_c) where $s_c \in \mathcal{M}^*$ is a sequence of messages, the messages on c, and $d_c \in \{\mathsf{delivered}, \overline{\mathsf{delivered}}\}$, indicating whether or not c delivered a message in the previous step, for every $c \in \mathcal{SC}$.
- The set $\mathbb{P}$ of propositional variables contains a propositional variable p_a for each constant $a \in \mathcal{A}$, a propositional variable p_v for each vertex v of a principal specified in Pr (recall that different principals have different sets of vertices), and propositional variables empty_c and $\mathsf{delivered}_c$ for every $c \in \mathcal{SC}$.
- The evaluation π of the propositional variables in a state $q =$

[2]Messages which are to be delivered to the intruder or scheduled secure channels will immediately be added to the intruder's knowledge and the secure channel buffer, respectively.

$(\mathcal{K}, \overline{P}, \overline{\mathbf{m}}, \overline{s})$ is defined as follows:

$$\begin{aligned}\pi(q) \;=\; & \{p_a \mid a \in d(\mathcal{K}) \cap \mathcal{A}\} \cup \\ & \{p_v \mid v \text{ is the root of } P_a \text{ for some } a \in \mathcal{H}\} \cup \\ & \{\mathsf{empty}_c \mid c \in \mathcal{SC}, s_c = \varepsilon\} \cup \\ & \{\mathsf{delivered}_c \mid c \in \mathcal{SC}, d_c = \mathsf{delivered}\},\end{aligned}$$

i.e., p_a is true in q if the intruder can derive a from its current knowledge, p_v is true in q if some honest principal is in vertex v, empty_c is true if c currently does not contain any messages, and $\mathsf{delivered}_c$ is true if $d_c = \mathsf{delivered}$.

- For $q = (\mathcal{K}, \overline{P}, \overline{\mathbf{m}}, \overline{s}) \in Q$ as above and a player $\alpha \in \Sigma$, we define $\Delta(q, \alpha)$ as follows:
  - If $\alpha \in \mathcal{H}$ with $P_\alpha = (V, E, v_0, \ell_p, \ell)$, the set $\Delta(q, \alpha)$ consists of all $v \in V$ such that $(\mathbf{m}_\alpha, P_\alpha) \overset{v}{\mapsto} (\mathbf{m}, P'_\alpha)$ for some $\mathbf{m}$ and P'_α, i.e., α can take one of the edges leaving the current vertex. If this set is empty, we define $\Delta(q, \alpha) = \{v_0\}$ (see below for an explanation).
  - If $\alpha = \mathcal{I}$, the set $\Delta(q, \alpha)$ consists of all $\mathbf{m} : \mathcal{IC} \to \mathcal{M}_\circ$ such that $\mathbf{m}(c) \in d(\mathcal{K})$ for every $c \in \mathcal{IC}$, i.e., the intruder can send messages on the channels that he controls, where the messages are derived from his current knowledge.
  - If $\alpha \in \mathcal{SC}$, then $\Delta(q, \alpha) = \{0\}$, in case $s_\alpha = \varepsilon$ (i.e., s_α is empty), and $\Delta(q, \alpha) = \{0, 1\}$, otherwise ($1 \mathrel{\widehat{=}}$ "α delivers the next message in the sequence" and $0 \mathrel{\widehat{=}}$ "α does not deliver a message").
- For $q = (\mathcal{K}, \overline{P}, \overline{\mathbf{m}}, \overline{s}) \in Q$ and a total move $\gamma \in \Delta_q^\Sigma$, we define the γ-successor $\delta(q, \gamma)$ of q to be the state $(\mathcal{K}', \overline{P}', \overline{\mathbf{m}}', \overline{s}')$ with $\overline{P}' = \{P'_a\}_{a \in \mathcal{H}}$, $\overline{\mathbf{m}}' = \{m'_a\}_{a \in \mathcal{H}}$, and $\overline{s}' = \{(s'_c, d'_c)\}_{c \in \mathcal{SC}}$ where:
  - $\mathcal{K}'$ is $\mathcal{K}$ with the following messages added: (1) the first message of s_c for every $c \in \mathsf{Sch}(\mathcal{H}, \mathcal{D}) \cap \mathsf{ch}(Pr)$ with $\gamma(c) = 1$, and (2) the message $\mathbf{m}(c)$ for every $c \in (\mathsf{Net}(a, \mathcal{P}) \cup \mathsf{Dir}(a, \mathcal{D})) \cap \mathsf{ch}(Pr)$ and every $a \in \mathcal{H}$ such that $\gamma(a) = v$ and $(\mathbf{m}_a, P_a) \overset{v}{\mapsto} (\mathbf{m}, P'_a)$, i.e., the intruder learns all messages sent by the scheduled secure channels to dishonest principals, by honest principals on the network (to honest or dishonest principals) or on direct secure channels to

dishonest principals.[3]

- P'_a is such that $(\mathbf{m}_a, P_a) \overset{v}{\mapsto} (\mathbf{m}, P'_a)$ for some $\mathbf{m}$, $v = \gamma(a)$, for every $a \in \mathcal{H}$, i.e., principal a takes edge v (and performs receive-send actions according to the rule the edge to v is labeled with).
- $\mathbf{m}'_a(c)$ for $a \in \mathcal{H}$ is equal to:
  - $\gamma(\mathcal{I})(c)$, if $c \in (\mathsf{Net}(\mathcal{P}, a) \cup \mathsf{Dir}(\mathcal{D}, a))$,
  - the first message of s_c, if $c \in \mathsf{Sch}(\mathcal{P}, a)$ and $\gamma(c) = 1$,
  - $\circ$, if $c \in \mathsf{Sch}(\mathcal{P}, a)$ and $\gamma(c) = 0$,
  - $\mathbf{m}(c)$, if $c = \mathsf{dir}(b, a)$ for some $b \in \mathcal{H}$ such that $(\mathbf{m}_b, P_b) \overset{v'}{\mapsto} (\mathbf{m}, P'_b)$ and $\gamma(b) = v'$, i.e., b has written $\mathbf{m}(c)$ on the direct secure channel c from b to a.
- s'_c for $c = \mathsf{sch}(a, b)$ is defined as follows: Let $s_c = m_1 \ldots m_n$. If $a \in \mathcal{H}$, then let $m = \mathbf{m}(c)$ where $(\mathbf{m}_a, P_a) \overset{v}{\mapsto} (\mathbf{m}, P'_a)$ for $v = \gamma(a)$. If $a \in \mathcal{D}$, then let $m = \mathbf{m}(c)$ with $\gamma(\mathcal{I}) = \mathbf{m}$. Now, if $\gamma(c) = 0$ and $m = \circ$, then $s'_c = s_c$. If $\gamma(c) = 1$ and $m = \circ$, then $s'_c = m_2 \cdots m_n$. If $\gamma(c) = 0$ and $m \neq \circ$, then $s'_c = m_1 \cdots m_n m$. If $\gamma(c) = 1$ and $m \neq \circ$, then $s'_c = m_2 \cdots m_n m$. (Note that if $\gamma(c) = 1$, then $n \geq 1$.)
- $d'_c = \mathsf{delivered}$ if $\gamma(c) = 1$, and $d'_c = \overline{\mathsf{delivered}}$ otherwise, for every $c = \mathsf{sch}(a, b) \in \mathsf{ch}(Pr)$.

We call the state $q^0 = (\mathcal{K}_0, \{P^0_a\}_{a \in \mathcal{H}}, \overline{\mathbf{m}}^0, \overline{s}^0)$ the *initial state of S_{Pr}* where $\overline{\mathbf{m}}^0 = \{\mathbf{m}^0_a\}_{a \in \mathcal{H}}$ with $\mathbf{m}^0_a(c) = \circ$ for every $c \in \mathcal{C}(\mathcal{P}, a)$ and $\overline{s}^0 = \{(s^0_c, d^0_c)\}_{c \in \mathcal{SC}}$ with $s^0_c = \varepsilon$ and $d^0_c = \overline{\mathsf{delivered}}$ for every $c \in \mathcal{SC}$.

We defined $\Delta(q, \alpha)$ for $\alpha \in \mathcal{H}$ in such a way that it is never empty. More precisely, if in the current vertex none of the outgoing edges can be taken, α will stay in the current state. This, in accordance with standard specifications, models that unexpected messages are ignored and guarantees that honest principals, just like all other agents in S_{Pr}, can always take an action, and hence, computations of the overall system will never be blocked. Note that one can explicitly add edges to the specification of an honest principal that lead to error states and are taken in case of unexpected messages. The

[3]In case secure channels are not supposed to be read protected, one would add all messages on direct and scheduled secure channels to the current intruder knowledge. However, here we model secure channels to be read protected, i.e., the intruder only gets to see the messages on secure channels to *dishonest* principals.

concrete examples that we consider are always complete in the sense that in every vertex there will be an edge (possibly a self-loop) that the principal can take.

4.3 Main Results

In this section, we summarize the main results of this chapter. First, we define the general protocol induced AMC-model checking problem and some sub cases. We then state our (un-)decidability and complexity-theoretic results.

Let $Pr = (\mathcal{H}, \mathcal{D}, \mathcal{K}_0, \{P_a^0\}_{a\in\mathcal{H}})$ be a protocol and $S_{Pr} = \langle \Sigma_{Pr}, Q_{Pr}, \mathbb{P}_{Pr}, \pi_{Pr}, \Delta_{Pr}, \delta_{Pr} \rangle$ the concurrent game structure induced by Pr.

We call

$$\textsc{Pamc} = \{(Pr, \varphi) \mid Pr \text{ a protocol and } \varphi \text{ an AMC-formula over } \Sigma_{Pr} \text{ and } \mathbb{P}_{Pr} \text{ such that } (S_{Pr}, q^0) \models \varphi \text{ where } q^0 \text{ is the initial state of } S_{Pr}\}$$

the *(general) protocol induced AMC-model checking problem.* The size of an instance (Pr, φ) of this problem is defined to be $|Pr| + |\varphi|$.

As we will see, this problem is undecidable. To identify the main sources of undecidability and to obtain decidable sub cases, we now introduce certain classes of protocols and define certain fragments of AMC.

We call a protocol Pr *dishonest scheduled secure channel free (dssc-free)* if no honest principal uses a scheduled secure channel from a dishonest principal as input channel, i.e., $\mathsf{ch}(Pr) \cap \mathsf{Sch}(\mathcal{D}, \mathcal{H}) = \emptyset$. Otherwise, we call a protocol *dssc-containing.* Note that dssc-free protocols allow honest principals to use direct secure channels from dishonest principals as input channel. Since these channels are completely controlled by the adversary, they provide the adversary with more power than scheduled secure channels. Hence, the exclusion of scheduled secure channels from dishonest principals is not a real restriction in terms of the power of the adversary.

We also consider what we call greedy protocols, which contains only greedy honest principals. Intuitively, an honest principal is greedy if it does not ignore messages in case they conform to the protocol specification. Formally, greedy protocols are defined as follows.

An a-rule $\mathbf{r} \Rightarrow \mathbf{s}$ is *consuming* if $\mathsf{ch}(\mathbf{r}) \neq \emptyset$. Intuitively, if principal a performs a consuming rule, then the form of the incoming messages matters.

An a-instance P is *greedy* if for all vertices v of P the outgoing edges of v labeled with consuming rules have priorities strictly higher than the priority of the self-loop of v (if any). Informally speaking, when a greedy principal can read a term using some consuming rule, then he has to apply such a rule, and hence, as a result moves to another vertex.

A protocol Pr is *greedy* if all of its a-instances, for $a \in \mathcal{H}$, are. Assuming a protocol to be greedy is a realistic assumption since in typical protocol specifications honest principals will not ignore messages if these messages conform to the messages they expect to receive.

We consider a fragment of AMC, called $\mathcal{I}$-monotone formulas where $\mathcal{I}$ denotes the intruder in the concurrent game structure induced by a protocol. Formally, an AMC-formula φ is *$\mathcal{I}$-positive* if all subformulas of φ of the form $\langle\langle A\rangle\rangle\psi$ with $\mathcal{I} \in A$ fall under an even number of negations and all subformulas of φ of the form $\langle\langle A\rangle\rangle\psi$ with $\mathcal{I} \notin A$ fall under an odd number of negations. An AMC-formula φ is *$\mathcal{I}$-negative* if $\neg\varphi$ is $\mathcal{I}$-positive. A formula φ is *$\mathcal{I}$-monotone* if it is either $\mathcal{I}$-positive or $\mathcal{I}$-negative.

As we mentioned above, each AMC-formula can be written in negation normal form using the abbreviation introduced in Section 4.1.2. It is easy to see that if φ is an $\mathcal{I}$-positive AMC-formula and φ' is the corresponding AMC-formula in negation normal form, then (i) for each subformula of φ' of the form $\langle\langle A\rangle\rangle\psi'$ we have that $\mathcal{I} \in A$ and (ii) for each subformula of φ' of the form $[\![A]\!]\psi'$ we have that $\mathcal{I} \notin A$.

$\mathcal{I}$-positive, -negative, and -monotone ATL- and ATL*-formulas are defined in the same way. As shown by Alur et al. [AHK02] (see also Theorem 4.1.1), every ATL and ATL*-formula can be translated into an equivalent AMC-formula. It is not hard to see that the translation preserves the property of being $\mathcal{I}$-positive/-negative, i.e., the translation of an $\mathcal{I}$-positive/-negative ATL*-formula yields an $\mathcal{I}$-positive/-negative AMC formula.

While the class of $\mathcal{I}$-monotone AMC-formulas is a proper fragment of the set of all AMC-formulas in terms of expressibility, all formulas (typically ATL or fair ATL) that we encountered in the literature for specifying security properties of cryptographic protocol are $\mathcal{I}$-monotone. Hence, the restriction to $\mathcal{I}$-monotone formulas does not seem to be a restriction from a practical point of view (see Section 4.4 for more details).

In what follows, we denote by

PAMC(greedy/non-greedy,dssc-containing/-free,$\mathcal{I}$-positive/-negative/-monotone)

the protocol induced AMC-model checking problem where the class of protocols is restricted to those that are i) greedy/non-greedy and ii) dssc-containing/-free and the AMC-formulas considered are $\mathcal{I}$-positive/-negative/-monotone, respectively.

Now, we can state our main results. The first theorem shows that PAMC is undecidability in case of non-greedy protocols.

Theorem 4.3.1 PAMC*(non-greedy,dssc-free,$\mathcal{I}$-positive) is undecidable, and hence, so are* PAMC*(non-greedy,dssc-free,$\mathcal{I}$-negative) and* PAMC*(non-greedy, dssc-free,$\mathcal{I}$-monotone).*

The proof of this theorem is presented in Section 4.5. It shows that the problem is undecidable even if no scheduled secure channels are used at all, i.e., neither from dishonest nor from honest principals, and if a very simple fixed $\mathcal{I}$-positive formula is used, namely, $\langle\langle\mathcal{I}\rangle\rangle\Diamond p_{\mathsf{ok}}$ where p_{ok} is a propositional variable.

The above theorem shows that to obtain decidability, one at least has to consider greedy protocols. The following theorem exhibits another source of undecidability, namely scheduled secure channels from dishonest parties.

Theorem 4.3.2 PAMC*(greedy, dssc-containing, $\mathcal{I}$-positive) is undecidable, and hence, so are* PAMC*(greedy, dssc-containing, $\mathcal{I}$-negative) and* PAMC*(greedy, dssc-containing, $\mathcal{I}$-monotone).*

The proof is presented in Section 4.6. Again, a fixed formula suffices for the proof, namely $\langle\langle\mathcal{I}, \mathsf{sch}(\mathsf{pcp}, \mathsf{test})\rangle\rangle\Diamond p_{\mathsf{ok}}$.

The two theorems above show that to obtain decidability, one has to restrict protocols to be greedy and dssc-free. The following theorem states that for this class of protocol and $\mathcal{I}$-monotone formulas we obtain decidability of the AMC-model checking problem.

Theorem 4.3.3 *The problem* PAMC(*greedy, dssc-free,$\mathcal{I}$-monotone*) *is decidable. More precisely, the problem* PAMC(*greedy, dssc-free,$\mathcal{I}$-positive*) *is NEXPTIME-complete, and hence,* PAMC(*greedy, dssc-free,$\mathcal{I}$-negative*) *is coNEXPTIME-complete.*

The only gap that the above theorem leaves is whether PAMC is also decidable for non $\mathcal{I}$-monotone formulas. As explained before, from a practical point of view the theorem seems to suffice since the formulas that we encountered in the literature fall into the class of $\mathcal{I}$-monotone formulas. While, as the undecidability results show, the restrictions to greedy and dssc-free protocol cannot be avoided, these restrictions are also not severe since typically protocols are greedy and requiring protocols to be dssc-free does not restrict the power of the adversary.

4.4 Example Properties

In this section, we illustrate the kind of properties that can be expressed in the $\mathcal{I}$-monotone fragment of AMC. Kremer and Raskin [KR02, KR01] were the first to formulate properties of fair exchange protocols, including contract-signing and non-repudiation protocols, in terms of fair ATL, a fragment of ATL* [AHK02], and hence, of AMC [AHK02] (see also Theorem 4.1.1). It turns out that all properties that Kremer and Raskin have formulated fall into the $\mathcal{I}$-monotone fragment of AMC, suggesting that the $\mathcal{I}$-monotone fragment of AMC suffices for most properties of interest. We demonstrate this fact by recalling some of these properties. Since, as mentioned, AMC is more expressive than ATL*, in what follows, for convenience we use ATL* as the specification language. As we will see, we will only need $\mathcal{I}$-monotone ATL* formulas. As mentioned in Section 4.3, $\mathcal{I}$-monotonicity is preserved under the translation to AMC as proposed in [AHK02], i.e., AMC formulas corresponding to $\mathcal{I}$-monotone ATL* formulas are $\mathcal{I}$-monotone.

The precise formulations of the properties stated by Kremer and Raskin typically depend on the specific protocol analyzed. For concreteness, we will therefore consider the specification of the ASW protocol Pr_{ASW}, with honest A and T and dishonest B, as presented in Section 4.2.4. Hence, formulas are stated w.r.t. the concurrent game structure $S_{Pr_{ASW}}$ induced by Pr_{ASW}.

As for example in [AHK02], we use $\Diamond\varphi$ (read "eventually φ") as abbreviation for the LTL-formula (true $\mathcal{U}$ φ) and $\Box\varphi$ (read "always φ") as abbreviation for $\neg\Diamond\neg\varphi$.

Fairness According to Kremer and Raskin, a protocol is *un*fair for honest A if dishonest B together with all scheduled secure channels has a strategy to obtain a signed contract from A such that A does not have a strategy to receive a signed contract from B, given that the secure scheduled channels between A and T are fair, i.e., messages on these channels are eventually delivered.

For the ASW protocol as specified in Section 4.2.4 this property can be formalized by the following $\mathcal{I}$-positive ATL* formula where $\mathcal{SC} = \mathsf{sch}(\{A, B, T\}, \{A, B, T\}) \cap \mathsf{ch}(\mathrm{Pr}_{ASW})$ is the set of all scheduled secure channels used in Pr_{ASW}:

$$\langle\!\langle \mathcal{I}, \mathcal{SC} \rangle\!\rangle \Diamond (p_{\mathtt{Bhascont}} \wedge \neg \langle\!\langle A \rangle\!\rangle (\varphi_{FairSch} \rightarrow \Diamond \varphi_{\mathtt{Ahascont}})) \tag{4.1}$$

where

$$\varphi_{FairSch} = \bigwedge_{c \in \mathsf{sch}(\{A,T\},\{A,T\})} \Box\Diamond\neg \mathsf{empty}_c \rightarrow \Box\Diamond \mathsf{delivered}_c \tag{4.2}$$

says that the scheduled secure channels between A and T are fair[4] and

$$\varphi_{\mathtt{Ahascont}} = p_{\mathsf{contract}} \vee p_{\mathsf{resolved}_1} \vee p_{\mathsf{resolved}_2} \tag{4.3}$$

says that A has a standard or replacement contract, according to the protocol specification.

The property formulated in (4.1) requires A to use the protocol in a "smart" way in order to get a signature from B. An alternative, stronger formulation of fairness would require that if A finished the protocol, and hence, if A cannot take any further action, either both A and B or neither of the two parties has a signed contract, provided that the scheduled secure channels between A and T are fair. In other words, the protocol is *un*fair, if there exists a state in the protocol run where i) A cannot take any further action, ii) A does not have a signature from B, but iii) B has a signature from A. Formally:

$$\langle\!\langle \mathcal{I}, A, T, \mathcal{SC} \rangle\!\rangle (\varphi_{FairSch} \wedge \Diamond (\varphi_{\mathtt{Afinished}} \wedge \neg \varphi_{\mathtt{Ahascont}} \wedge p_{\mathtt{Bhascont}})) \tag{4.4}$$

[4] Alternatively, one could require the other scheduled secure channels to be fair as well. The formulation of fairness for channels as presented is standard and, for example, also appears in [AHK02].

where

$$\varphi_{\mathtt{Afinished}} \quad = \quad p_{\mathsf{contract}} \vee p_{\mathsf{resolved}_1} \vee p_{\mathsf{resolved}_2} \vee p_{\mathsf{aborted}} \qquad (4.5)$$

says that A finished her protocol run.

Timeliness According to Kremer and Raskin, a protocol is timely for honest A if A has a strategy to finish the protocol while preserving fairness. Again, the scheduled secure channels (at least those between A and T) are required to be fair. Formally, timeliness for A is expressed by the following $\mathcal{I}$-negative ATL* formula:

$$\langle\langle A\rangle\rangle(\varphi_{FairSch} \rightarrow \Diamond(\varphi_{\mathtt{Afinished}} \wedge \\ (\neg\varphi_{\mathtt{Ahascont}} \rightarrow \neg\langle\langle \mathcal{I}, \mathcal{SC}\rangle\rangle(\varphi_{FairSch} \wedge \Diamond p_{\mathtt{Bhascont}})))). \qquad (4.6)$$

Balance and Abuse-freeness According to Kremer and Raskin, a protocol is *un*balanced for honest A if at some stage of the protocol run dishonest B has both a strategy to obtain a signature from A and a strategy to prevent A from obtaining a signature from B. For the protocol to be abusive, one additionally requires that B can convince an outside party C of this property. Whether or not B has this ability is indicated, in the model of Kremer and Raskin, by a propositional variable p_{prove2C}, which can as well be expressed in terms of propositional variables on vertices. Again, the scheduled secure channels between A and T are required to be fair. Unbalanced for A can be formulated as an $\mathcal{I}$-positive ATL* formula as follows:

$$\langle\langle \mathcal{I}, \mathcal{SC}, A, T\rangle\rangle\varphi_{FairSch} \wedge \Diamond(\varphi_{\mathsf{getcontract}} \wedge \varphi_{\mathsf{prevent}}) \qquad (4.7)$$

where

$$\varphi_{\mathsf{getcontract}} = \langle\langle \mathcal{I}, \mathcal{SC}''\rangle\rangle\varphi_{FairSch} \rightarrow \Diamond p_{\mathtt{Bhascont}}, \qquad (4.8)$$

$$\varphi_{\mathsf{prevent}} = \langle\langle \mathcal{I}, \mathcal{SC}''\rangle\rangle\varphi_{FairSch} \rightarrow \Diamond(\neg\langle\langle A\rangle\rangle(\varphi_{FairSch} \rightarrow \Diamond\varphi_{\mathtt{Ahascont}})) \qquad (4.9)$$

with $\mathcal{SC}'' = \mathsf{sch}(\{B,T\},\{B,T\}) \cap \mathsf{ch}(\mathrm{Pr}_{ASW})$.

Given p_{prove2C}, according to Kremer and Raskin abusiveness for A is formalized by the following ATL* formula:

$$\langle\langle \mathcal{I}, \mathcal{SC}, A, T\rangle\rangle\varphi_{FairSch} \wedge \Diamond(p_{\mathsf{prove2C}} \wedge \varphi_{\mathsf{getcontract}} \wedge \varphi_{\mathsf{prevent}}) \qquad (4.10)$$

We note that the more general, protocol-independent formulation of abuse-freeness proposed in [KKW06] is not captured by the formulation of Kremer and Raskin. The formulation in [KKW06] defines abuse-freeness in terms of certain tests. In order to formulate this property in ATL* or AMC, one would need to augment these logics by certain predicates reflecting the tests.

Very similar formulas as the ones presented above have been stated by Kremer and Raskin for non-repudiation protocols [KR01]. They are $\mathcal{I}$-monotone as well.

4.5 Proof of Theorem 4.3.1

We prove Theorem 4.3.1 by a reduction from Post's Corresponding Problem (PCP). Let us first recall the definition of PCP.

Given an alphabet A with $|A| \geq 2$, an instance Π of PCP over A is a sequence $(u_i, v_i)_{i=1}^n = (u_1, v_1), \ldots, (u_n, v_n)$ of pairs (u_i, v_i) of words $u_i, v_i \in A^*$. A solution of such an instance is a non-empty sequence $(k_i)_{i=1}^l = k_1, \ldots, k_l$ of indices $k_i \in \{1, \ldots, n\}$, $i \in \{1, \ldots, l\}$, for some l such that $u_{k_1} \cdots u_{k_l} = v_{k_1} \cdots v_{k_l}$. Now, given an instance of PCP (over A) the question is whether it has a solution. It is well-known that this problem is undecidable.

We now prove Theorem 4.3.1 by reduction from PCP. Let Π be an instance of PCP over A as above. We (effectively) associate a protocol Pr_Π and a formula φ_Π to Π such that Π has a solution iff $(S_{Pr_\Pi}, q^0) \models \varphi_\Pi$ where q^0 is the initial state of S_{Pr_Π}.

The set of atoms of Pr_Π is $\mathcal{A}_\Pi = A \cup \{\perp, 1, \ldots, n\}$. For a word $u \in \mathcal{A}_\Pi^*$ and a term t, we define $t \cdot u$ by induction on the length of u: $t \cdot u = t$ for $u = \varepsilon$ and $t \cdot u = \langle t, a \rangle \cdot v$ for $u = av$ and $a \in \mathcal{A}_\Pi$.

We encode a solution of Π as a sequence of terms over $\mathcal{A}_\Pi = A \cup \{\perp, 1, \ldots, n\}$ as follows: A sequence $t_0, \ldots, t_l$ of terms over $\mathcal{A}_\Pi$ is called a *solution sequence for* Π if the following three conditions are satisfied:

i) $t_0 = \langle \perp, \perp, \perp \rangle$,

ii) $t_l = \langle m_1, m_2, m_2 \rangle$ for terms m_1 and m_2 over $\mathcal{A}_\Pi$, and

iii) for all $i \in \{0, \ldots, l-1\}$, if $t_i = \langle s, s', s'' \rangle$, then $t_{i+1} = \langle s \cdot j, s' \cdot u_j, s'' \cdot v_j \rangle$ for some $j \in \{1, \ldots, n\}$.

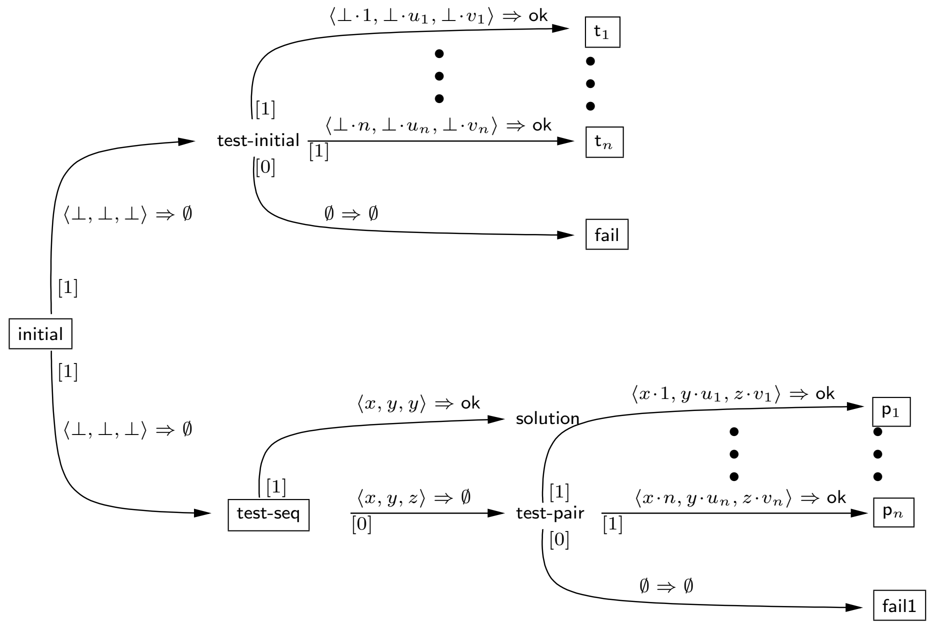
It is easy to see that Π has a solution iff there exists a solution sequence for Π. For a solution $(k_i)_{i=1}^l$ of Π we call the solution sequence $t_0, \ldots, t_l$ with $t_0 = \langle \bot, \bot, \bot \rangle$ and $t_{i+1} = \langle s_1 \cdot k_{i+1}, s_2 \cdot u_{k_{i+1}}, s_3 \cdot v_{k_{i+1}} \rangle$ for $0 \leq i < l$ and $t_i = \langle s_1, s_2, s_3 \rangle$ the *solution sequence associated with* $(k_i)_{i=1}^l$.

We define $\varphi_\Pi = \langle\!\langle \mathcal{I} \rangle\!\rangle \Diamond p_{\mathsf{ok}}$, i.e., this formula is true in those states where $\mathcal{I}$ has a strategy to obtain ok. (Note that φ_Π is presented as an ATL-formula, which by Theorem 4.1.1 can be turned into an AMC-formula).

The protocol Pr_Π is defined as follows: There is one honest principal, called test, and one dishonest principal, called pcp, in Pr_Π. The initial knowledge of the intruder is defined to be $\mathcal{K}_\Pi = \mathcal{A}_\Pi$. The honest principal test is specified by the test-instance P_{test} depicted in Figure 4.3 and explained next. Altogether, we define $Pr_\Pi = (\{\mathsf{test}\}, \{\mathsf{pcp}\}, \mathcal{K}_\Pi, \{P_{\mathsf{test}}\})$.

Principal test does not use any direct or scheduled secure channels. Hence, the only channel from which principal test reads is net(pcp, test) and the only channel to which test writes is net(test, pcp). Hence, the left-hand side of the test-rules depicted in Figure 4.3 are the messages test reads from net(pcp, test) and the messages on the right-hand side of these rules are the messages test writes to net(test, pcp). The labels [0] and [1] present the priorities of the edges. Vertices with boxes have self-loops with priority 0, those without do not have self-loops. Note that P_{test} is non-greedy; the greediness condition is violated in vertex test-seq.

The purpose of principal test is to test whether the intruder is able to send a solution sequence. More precisely, test guarantees that the intruder has a strategy to obtain ok iff the intruder is able to send a solution sequence to test, where the intruder is supposed to send in every step of a computation in S_{Pr_Π} one element of such a sequence. In initial, once $\langle \bot, \bot, \bot \rangle$ is received, test can decide to go to test-initial or test-seq. The purpose of going to test-initial is to check whether the successor term of $\langle \bot, \bot, \bot \rangle$ is in fact a successor term according to the definition of a solution sequence. (The out-going edge from test-initial with priority 0 guarantees that the intruder has to send a new term after the initial term $\langle \bot, \bot, \bot \rangle$.) The purpose of going to test-seq is to either stay there until a term of the form $\langle x, y, y \rangle$ is received, and hence, a valid last term of the sequence, or to check at some point of the sequence whether two consecutive terms are connected according to the definition of solution sequences (which can be done by moving to test-pair).



$\langle \bot\cdot 1, \bot\cdot u_1, \bot\cdot v_1\rangle \Rightarrow$ ok
t_1
[1]
$\langle \bot\cdot n, \bot\cdot u_n, \bot\cdot v_n\rangle \Rightarrow$ ok
test-initial
[1]
t_n
[0]
$\langle \bot, \bot, \bot\rangle \Rightarrow \emptyset$
$\emptyset \Rightarrow \emptyset$
fail
[1]
initial
[1]
$\langle x, y, y\rangle \Rightarrow$ ok
solution
$\langle x\cdot 1, y\cdot u_1, z\cdot v_1\rangle \Rightarrow$ ok
p_1
$\langle \bot, \bot, \bot\rangle \Rightarrow \emptyset$
[1]
[1]
$\langle x, y, z\rangle \Rightarrow \emptyset$
$\langle x\cdot n, y\cdot u_n, z\cdot v_n\rangle \Rightarrow$ ok
test-seq
[0]
test-pair
[1]
p_n
[0]
$\emptyset \Rightarrow \emptyset$
fail1

Figure 4.3: Honest instance $P_{\textsf{test}}$ in the proof of Theorem 4.3.1. A box around a node is an abbreviation for a self loop with priority 0.

Before we prove the correctness of our construction, we introduce some notation. By definition, see Section 4.2.5, a state of the concurrent game structure $\mathcal{S}_{Pr_\Pi}$ specified by the protocol Pr_Π is a tuple $p = (\mathcal{K}, \{P\}, \{\mathbf{m}\}, \bar{s})$ where $\mathcal{K}$ is the intruder knowledge, P is a test-instance, $\mathbf{m}$ is a mapping that assigns messages to the channels read by test, and $\bar{s}$ is a family of message sequences representing the states of the scheduled secure channels. Because there are no secure channels used in this protocol we will omit the last component of states when referring to them. The only channel from which participant test reads messages is net(pcp, test). Thus, $\mathbf{m}$ assigns messages to net(pcp, test) and we will only write the message $\mathbf{m}(\textsf{net}(\textsf{pcp}, \textsf{test}))$ when specifying a state. For ease of notation, we specify a test-instance in a state simply by its current root node. Thus, by these conventions the initial state of the concurrent game structure $\mathcal{S}_{Pr_\Pi}$ is given by $q_{\textsf{init}} = (\mathcal{K}_\Pi, \textsf{initial}, \circ)$. For a state $p = (\mathcal{K}, s, m)$ we denote the components $\mathcal{K}$, s, and m by $\mathcal{K}(p)$, $\textsf{state}_{\textsf{test}}(p)$, and $\textsf{net}(\textsf{pcp}, \textsf{test})(p)$, respectively. We call m the *value of channel net(pcp, test) in state p*.

We now show that Π has a solution iff $(S_{Pr_\Pi}, q^0) \models \varphi_\Pi$:

$\Rightarrow$: First, we show that if Π has a solution, then the intruder has a strategy to obtain ok. Intuitively, the strategy of the intruder is to send a solution sequence to instance test. More specifically, let $(k_i)_{i=1}^l$ be a solution of Π and let $t_0, \ldots, t_l$ be the solution sequence associated with $(k_i)_{i=1}^l$. We may assume, w.l.o.g., that t_l is the first among the terms in the sequence of the form $\langle m_1, m_2, m_2 \rangle$ for some m_1 and m_2.

The (positional) strategy σ_{ok} of the intruder to obtain ok only depends on the message on channel $\mathsf{net}(\mathsf{pcp}, \mathsf{test})$. We define σ_{ok} by the following table: (Note that the choice of the intruder is which message he sends to instance test, i.e., what the value of $\mathsf{net}_{\mathsf{test}}^{\mathsf{pcp}}$ in the next state of the concurrent game structure $\mathcal{S}_{Pr_\Pi}$ is.)

$\mathsf{net}(\mathsf{pcp}, \mathsf{test})(q)$	$\sigma_{\mathsf{ok}}(q)$
$\circ$	t_0
t_i	t_{i+1} for $i \in \{0, \ldots, l-1\}$

We have to show that if the intruder follows this strategy, then every computation in $\mathcal{S}_{Pr_\Pi}$ starting from the initial state will reach a state q such that $\mathsf{ok} \in \mathcal{K}(q)$. Let $\rho = q_0 q_1 \ldots$ be a computation consistent with σ_{ok} and such that $q_0 = q^0$. According to the specification of instance test, see Figure 4.3, we know that $\mathsf{state}_{\mathsf{test}}(q_2) \in \{\mathsf{test\text{-}initial}, \mathsf{test\text{-}seq}\}$. If $\mathsf{state}_{\mathsf{test}}(q_2) = \mathsf{test\text{-}initial}$, then $\mathsf{ok} \in \mathcal{K}(q_3)$ since we have that $\mathsf{net}(\mathsf{pcp}, \mathsf{test})(q_2) = t_1$. Thus, in this case we are done. If $\mathsf{state}_{\mathsf{test}}(q_2) = \mathsf{test\text{-}seq}$, then there is a minimal $i > 2$ such that $\mathsf{state}_{\mathsf{test}}(q_i) \neq \mathsf{test\text{-}seq}$ (note that if $\mathsf{state}_{\mathsf{test}}(q_j) = \mathsf{test\text{-}seq}$ and $\mathsf{net}(\mathsf{pcp}, \mathsf{test})(q_j) = t_n$, then according to the specification of instance test, $\mathsf{ok} \in \mathcal{K}(q_{j+1})$ and $\mathsf{state}_{\mathsf{test}}(q_{j+1}) = \mathsf{solution}$). If $\mathsf{state}_{\mathsf{test}}(q_i) = \mathsf{solution}$, then we are obviously done. If $\mathsf{state}_{\mathsf{test}}(q_i) = \mathsf{test\text{-}pair}$, then we know that $\mathsf{net}(\mathsf{pcp}, \mathsf{test})(q_{i-1}) \neq t_n$ (note that if the intruder has already sent t_n in q_{i-1} then $\mathsf{state}_{\mathsf{test}}(q_i)$ would be $\mathsf{solution}$). By the specification of instance test and by the definition of σ_{ok} we know that $\mathsf{state}_{\mathsf{test}}(q_{i+1}) \in \{\mathsf{p}_1, \ldots, \mathsf{p}_n\}$ and thus, we are done.

$\Leftarrow$: Now, we prove the other direction, i.e., if the intruder has a strategy to obtain ok, then Π has a solution. It suffices to show that there is a solution sequence for instance Π. Let σ_{ok} be a strategy of the intruder such that in each computation of $\mathcal{S}_{Pr_\Pi}$ starting from the initial state and consistent with σ_{ok} the intruder obtains ok. We want to show that the intruder has to send a solution sequence $t_0, t_1, \ldots, t_l$ for Π to instance test. More specifically, we want to show that there is a computation $\rho = q_0 q_1 \ldots$ of $\mathcal{S}_{Pr_\Pi}$, $q_0 = q^0$, and an index i such that $\sigma_{\mathsf{ok}}(q_0), \sigma_{\mathsf{ok}}(q_0 q_1), \ldots, \sigma_{\mathsf{ok}}(q_0 \cdots q_i)$ is a solution sequence for Π.

It is easy to see that it is w.l.o.g. to assume that (*) $\sigma_{\mathsf{ok}}(q^0) = \langle \bot, \bot, \bot \rangle$. The successor state q_1 of q^0 is $(\mathcal{K}_\Pi, \mathsf{initial}, \langle \bot, \bot, \bot \rangle)$. Since one possible choice of instance test in state q_1 is to proceed to state $\mathsf{test\text{-}initial}$ one possible successor state of q_1 is $(\mathcal{K}_\Pi, \mathsf{test\text{-}initial}, \circ)$, see Figure 4.3. Thus, if $\sigma_{\mathsf{ok}}(q_1) \neq \langle \bot \cdot j, \bot \cdot u_j, \bot \cdot v_j \rangle$ for all $j = 1, \ldots, n$, then instance test will proceed to state fail and the intruder will not obtain ok. Since σ_{ok} is a strategy for the intruder to obtain ok we can conclude that (**) $\sigma_{\mathsf{ok}}(q_1)$ is of the form $\langle \bot \cdot j, \bot \cdot u_j, \bot \cdot v_j \rangle$ for some $j \in \{1, \ldots, n\}$.

If the choice of instance test in state q_1 is to proceed to state $\mathsf{test\text{-}seq}$, then the successor state of q_1 is $(\mathcal{K}_\Pi, \mathsf{test\text{-}seq}, \sigma_{\mathsf{ok}}(q_1))$. By an inductive argument it is easy to see that for a run $\rho = q_0, q_1, q_2, \ldots$ of $\mathcal{S}_{Pr_\Pi}$ that is consistent with σ_{ok} we have that: (***) if $\mathsf{state}_{\mathsf{test}}(q_i) = \mathsf{test\text{-}seq}$, $\mathsf{net}(\mathsf{pcp}, \mathsf{test})(q_i) = \langle m_1, m_2, m_3 \rangle$, where $m_2 \neq m_3$, then $\sigma_{\mathsf{ok}}(q_i) = \langle m_1 \cdot j, m_2 \cdot u_j, m_3 \cdot v_j \rangle$ for some $j \in \{1, \ldots, n\}$.

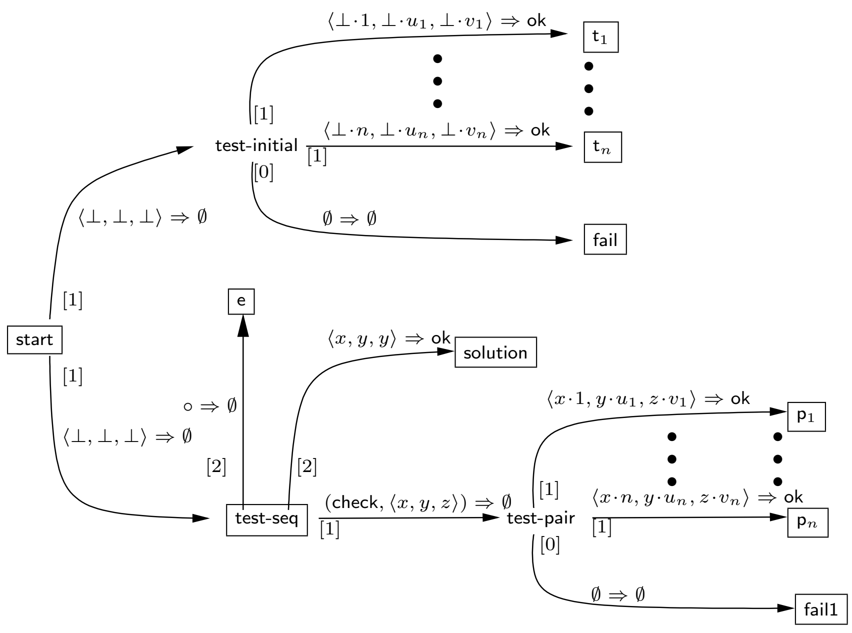
Since σ_{ok} is a strategy for the intruder to obtain ok there is no computation $\rho = q_0, q_1, \ldots$ of $\mathcal{S}_{Pr_\Pi}$ that is consistent with σ_{ok} such that $\mathsf{state}_{\mathsf{test}}(q_j) = \mathsf{test\text{-}seq}$ from some point on, i.e., $\mathsf{state}_{\mathsf{test}}(q_j) = \mathsf{test\text{-}seq}$ for all $j > i$ for some $i > 0$. Since instance test can decide to stay in state $\mathsf{test\text{-}seq}$ if $\mathsf{net}(\mathsf{pcp}, \mathsf{test})$ is not of the form $\langle m_1, m_2, m_2 \rangle$ for some terms m_1 and m_2 we can conclude that there is a computation $\rho' = q'_0, q'_1, \ldots$ that is consistent with σ_{ok} such that there is some minimal i such that $\mathsf{state}_{\mathsf{test}}(q'_i) = \mathsf{solution}$. Thus, $\mathsf{net}(\mathsf{pcp}, \mathsf{test})(q'_{i-1}) = \langle m_1, m_2, m_2 \rangle$ for some terms m_1, m_2. Together with (*), (**), and (***) we can conclude that the sequence $\sigma_{\mathsf{ok}}(q'_0), \sigma_{\mathsf{ok}}(q'_0 q'_1), \ldots, \sigma_{\mathsf{ok}}(q'_0 \cdots q'_{i-2})$ is a solution sequence for Π. $\square$

4.6 Proof of Theorem 4.3.2

The proof of Theorem 4.3.2 is very similar to the one of Theorem 4.3.1.

In Section 4.5, we used the non-greediness of instance P_{test} in node $\mathsf{test\text{-}seq}$ to check property iii) of solution sequences: From node $\mathsf{test\text{-}seq}$ of instance P_{test} there is a self-loop with priority 0 and a consuming edge with priority 0 (see Figure 4.3). Applying the self-loop means "do not check" and applying the outgoing edge means "check" whether the next two messages fulfill property iii) of solution sequences. In other words, in node test-seq, P_{test} can non-deterministically decide when to check property iii) of solution sequences for two consecutive messages. However, now we are not allowed to use non-greediness anymore. Nevertheless, we can simulate this non-deterministic behavior by introducing a new instance, which we will call P_{check}. Basically, this instance will send a "check" message to P_{test} in order to tell P_{test} when to check. More precisely, in the very first step of the protocol run P_{check} will send the "check" message on a scheduled secure channel $\mathsf{sch}(\mathsf{check}, \mathsf{test})$ to P_{test}. Then, this scheduled secure channel will non-deterministically decide when it delivers the "check" message. (Alternatively, P_{check} could be defined to be non-deterministic and decide when to send the "check" message to P_{test}, now on a direct secure channel. However, this instance seems to be less natural than the first one.) The only problem is that if in one step of the protocol run the $\mathsf{sch}(\mathsf{check}, \mathsf{test})$ decides to deliver the "check" message, then in the next state the intruder knows that a check will be performed in the next step. Hence, he could produce a message that together with the previous message sent passes the test, even though the rest of the sequence that the intruder is sending does not satisfy the conditions on solution sequences. In other words, the intruder knows one step in advance, i.e., before sending the second message of the two messages to be checked, when a check is going to be performed. (Note that in the proof of Theorem 4.3.1 this was not the case since by the time P_{test} changed its internal state to perform the check, the intruder must have sent the second message already.) We therefore let the intruder communicate with P_{test} only over a scheduled secure channel $\mathsf{sch}(\mathsf{pcp}, \mathsf{test})$. Now, by the time the intruder gets to know that a check is going to be performed, he must already have sent the second message of the two messages to be checked to $\mathsf{sch}(\mathsf{pcp}, \mathsf{test})$. In other words, he must already have committed to the second message, and hence, cannot change it anymore. Thus, this second message must have been valid in the first place.

We now present the reduction formally and prove its correctness. Let $\Pi = (u_i, v_i)_{i=1}^n$ be an instance of PCP over the alphabet A. As in Section 4.5,



$\langle \bot\cdot 1, \bot\cdot u_1, \bot\cdot v_1\rangle \Rightarrow$ ok
t$_1$
[1]
$\langle \bot\cdot n, \bot\cdot u_n, \bot\cdot v_n\rangle \Rightarrow$ ok
test-initial
[1]
t$_n$
[0]
$\langle \bot, \bot, \bot\rangle \Rightarrow \emptyset$
$\emptyset \Rightarrow \emptyset$
fail
[1]
e
start
$\langle x, y, y\rangle \Rightarrow$ ok
solution
[1]
$\circ \Rightarrow \emptyset$
$\langle x\cdot 1, y\cdot u_1, z\cdot v_1\rangle \Rightarrow$ ok
p$_1$
$\langle \bot, \bot, \bot\rangle \Rightarrow \emptyset$
[2]
[2]
[1]
(check, $\langle x, y, z\rangle$) $\Rightarrow \emptyset$
$\langle x\cdot n, y\cdot u_n, z\cdot v_n\rangle \Rightarrow$ ok
test-seq
[1]
test-pair
[1]
p$_n$
[0]
$\emptyset \Rightarrow \emptyset$
fail1

Figure 4.4: Honest instance P_{test} in the proof of Theorem 4.3.2. A box around a node is an abbreviation for a self-loop with priority 0.

we define a protocol Pr_Π and an ATL-formula φ_Π such that Π has a solution iff $(S_{Pr_\Pi}, q^0) \models \varphi_\Pi$ where q^0 is the initial state of S_{Pr_Π}.

Protocol Pr_Π contains two honest principals, P_{test} and P_{check} and one dishonest principal pcp. The initial knowledge of the intruder is defined by $\mathcal{K}_\Pi = A$. Altogether, $P_\Pi = (\{\mathsf{test}, P_{\mathsf{check}}\}, \{\mathsf{pcp}\}, \mathcal{K}_\Pi, \{P_{\mathsf{test}}, P_{P_{\mathsf{check}}}\})$ where the test-instance P_{test} is specified by the tree given in Figure 4.4, explained below. The instance P_{check} consists of only two edges: one edge (from the root to another node, say v) is labeled with a check-rule of the form $\emptyset \Rightarrow \mathsf{check}$ where check is sent via sch(check, test) to P_{test}. The second edge is a self-loop at v. Hence, the only action that P_{check} takes is to send, in the first protocol step, check to sch(check, test).

The definition of P_{test} (see Figure 4.4) is similar to the one of P_{test} in the proof of Theorem 4.3.1. However, now P_{test} may receive messages from two scheduled secure channels, sch(check, test) and sch(pcp, test). The convention in Figure 4.4 is that if the left-hand side of the rule consists only of one message, then this message comes from sch(pcp, test). If the left-hand side is a tuple with two components (this is only the case for the rule the edge from test-seq to test-pair is labeled with), then the first component is the message coming from $\mathsf{sch}(\mathsf{check},\mathsf{test})$ and the other one from $\mathsf{sch}(\mathsf{pcp},\mathsf{test})$.

The intuition behind P_{test} as presented in Figure 4.4 is the same as the one in Figure 4.3. The edge from test-seq to e is needed to guarantee that $\mathsf{sch}(\mathsf{pcp},\mathsf{test})$ always delivers messages. The priority of the edge from test-seq to test-pair is now 1, instead of 0, and hence, test-seq satisfies the greediness condition. However, because of the message check coming from $\mathsf{sch}(\mathsf{check},\mathsf{test})$, this edge will only be applied if $\mathsf{sch}(\mathsf{check},\mathsf{test})$ has delivered check.

The formula φ_Π is defined as $\varphi_\Pi = \langle\!\langle \mathcal{I}, \mathsf{sch}(\mathsf{pcp},\mathsf{test})\rangle\!\rangle F p_{\mathsf{ok}}$. (Note that, as in Section 4.5, we define, w.l.o.g., φ_Π as an ATL-formula.)

We now prove that our construction is correct, i.e., we prove that Π has a solution iff $(S_{Pr_\Pi}, q^0) \models \varphi_\Pi$:

$\Rightarrow$: First we show that if Π has a solution, then $\mathcal{I}$ together with the scheduled secure channel $\mathsf{sch}(\mathsf{pcp},\mathsf{test})$ has a strategy such that $\mathcal{I}$ obtains ok. Similar to the proof of Theorem 4.3.1, the strategy of $\mathcal{I}$ is to send a solution sequence to test via $\mathsf{sch}(\mathsf{pcp},\mathsf{test})$ and the strategy of $\mathsf{sch}(\mathsf{pcp},\mathsf{test})$ is to deliver a message whenever possible. More precisely, let $(k_i)_{i=1}^l$ be a solution of Π and let $t_0,\ldots,t_l$ be the solution sequence associated with $(k_i)_{i=1}^l$ (as defined in Section 4.5).

It is easy to see that if $\mathcal{I}$ and $\mathsf{sch}(\mathsf{pcp},\mathsf{test})$ follow their strategy, then after two steps participant test is in state start and message $\langle\bot,\bot,\bot\rangle$ is on channel $\mathsf{sch}(\mathsf{pcp},\mathsf{test})$ ready to be read by test. Also, P_{check} has written check on $\mathsf{sch}(\mathsf{check},\mathsf{test})$, which in turn may or may not have delivered this message. At this point, participant test has two alternatives to proceed: 1) advance to test-initial or 2) advance to test-seq (see Figure 4.4). If test advances to test-initial, then in the next step test advances to one of the states $\mathsf{t}_1,\ldots,\mathsf{t}_n$ and the intruder will obtain ok. If test advances to test-seq, we have to distinguish between two cases: 2a) message check is not delivered to test by $\mathsf{sch}(\mathsf{check},\mathsf{test})$ before the last message of the solution sequence sent by the intruder is delivered to test by $\mathsf{sch}(\mathsf{pcp},\mathsf{test})$ and 2b) message check is delivered to test before the last message of the solution sequence sent by the intruder is delivered to test. In case 2a) participant test advances to solution and the intruder will obtain ok. In case 2b) participant test advances to test-pair and since the intruder has sent a solution sequence to test and $\mathsf{sch}(\mathsf{pcp},\mathsf{test})$ immediately delivers all messages, test advances to one of the states $\mathsf{p}_1, \ldots, \mathsf{p}_n$ and the intruder obtains ok.

$\Leftarrow$: Now, we have to show that if the intruder together with the scheduled secure channel $\mathsf{sch}(\mathsf{pcp}, \mathsf{test})$ has a strategy σ_{ok} such that the intruder obtains ok, then the PCP-instance Π has a solution. It is easy to see when playing according to strategy σ_{ok} that the intruder at some point must send $\langle \perp, \perp, \perp \rangle$ to test over the scheduled secure channel $\mathsf{sch}(\mathsf{pcp}, \mathsf{test})$ and that at some point $\mathsf{sch}(\mathsf{pcp}, \mathsf{test})$ delivers this message. Thus, at some point participant test is in state start and $\langle \perp, \perp, \perp \rangle$ is ready to be read on channel $\mathsf{sch}(\mathsf{pcp}, \mathsf{test})$. There are two possible ways of how participant test may proceed: 1) advancing to state $\mathsf{test\text{-}initial}$ and 2) advancing to state $\mathsf{test\text{-}seq}$. If 1) participant test advances to $\mathsf{test\text{-}initial}$, then we have that (*) there has to be a message of the form $\langle \perp \cdot i, \perp \cdot u_i, \perp \cdot v_i \rangle$ stored on channel $\mathsf{sch}(\mathsf{pcp}, \mathsf{test})$ for some $i \in \{1, \ldots, n\}$ since otherwise in the next step test would advance to state fail and the intruder would not obtain ok, in contradiction to the assumption. If 2) participant test advances to state $\mathsf{test\text{-}seq}$, then because of the edge from $\mathsf{test\text{-}seq}$ to e, we know that in each step $\mathsf{sch}(\mathsf{pcp}, \mathsf{test})$ has to deliver a message to test, since otherwise participant test would advance to e and the intruder could not obtain ok anymore. We now distinguish between two cases: 2a) message check is never delivered to test by the scheduled secure channel $\mathsf{sch}(\mathsf{check}, \mathsf{test})$ and 2b) message check is delivered to test eventually. In case of 2a), we have that (**) at some point $\mathsf{sch}(\mathsf{pcp}, \mathsf{test})$ must deliver a message of the form $\langle m_1, m_2, m_2 \rangle$ to test since this is the only way for the intruder to obtain ok. In case of 2b), similar to the proof of Theorem 4.3.1, we can conclude that (***) the sequence of messages delivered by $\mathsf{sch}(\mathsf{pcp}, \mathsf{test})$ to test satisfy property iii) of the properties of solution sequences. At this point we use that the intruder sends messages to test via a scheduled secure channel. This guarantees that the intruder must have sent the next message in a sequence to $\mathsf{sch}(\mathsf{pcp}, \mathsf{test})$ before he knows that a check is going to be performed. Now, from (*), (**), and (***) we can conclude that the intruder has to send a solution sequence to test, and thus, Π has a solution. $\square$

4.7 Proof of Theorem 4.3.3

To prove Theorem 4.3.3, it obviously suffices to show that PAMC(greedy, dssc-containing, $\mathcal{I}$-positive) is NEXPTIME-complete. In the following subsections, this statement is proved. The proof of the complexity upper bound is presented in Section 4.7.1, with the proofs of key lemmas postponed to Section 4.7.2 to 4.7.6. The complexity lower bound is shown in Section 4.7.7.

4.7.1 Poof of Theorem 4.3.3 Using Key Lemmas

In this section, we prove Theorem 4.3.3 using three key lemmas. The proofs of these lemmas are postponed to subsequent sections.

Let $Pr = (\mathcal{H}, \mathcal{D}, \mathcal{K}, \{P_a\}_{a \in \mathcal{H}})$ be a greedy and dssc-free protocol and φ be an $\mathcal{I}$-positive AMC-formula over Σ_{Pr} and $\mathbb{P}_{Pr}$. Since every AMC-formula can (in polynomial-time) be turned into an AMC-formula in negation normal form, we may, w.l.o.g., assume that φ is in negation normal form. Let $S = S_{Pr} = \langle \Sigma, Q_S, \mathbb{P}, \pi, \Delta, \delta \rangle$ be the concurrent game structure induced by Pr.

We define an equivalence relation $\sim$ on Q_S as follows. For $q, q' \in Q_S$, we write $q \sim q'$ if q and q' are equal up to the messages on input ports of honest participants, i.e., $\mathcal{K}_q = \mathcal{K}_{q'}$, $\overline{P}_q = \overline{P}_{q'}$, and $\overline{s}_q = \overline{s}_{q'}$. We will write $q \prec q'$, if $q \not\sim q'$ and q' is a descendant of q in S, i.e., there exists $q_1, \ldots, q_n$ such that $q_1 = q$, $q_n = q'$, and q_{i+1} is a successor of q_i for every $i = 1, \ldots, n-1$. We extend these relations to states of the parity game $G^{\varphi}_{(S,q^0)}$ where q^0 is the initial state of S: We write $(q, \psi) \sim (q', \psi')$, if $q \sim q'$, and we write $(q, \psi) \prec (q', \psi')$, if $q \prec q'$.

We call a state q of S consuming, if on the input port of some honest participant a there is a message which can be read by some consuming rule. Since, by assumption a is greedy, this implies that a's state will change in the next step, i.e., a moves to a new vertex. Formally, a state $q = (\mathcal{K}, \overline{P}, \overline{\mathbf{m}}, \overline{s})$ is *consuming*, if there exists $a \in \mathcal{H}$ with $P_a = (V, E, v_0, \ell_p, \ell)$ and if there exists $v \in V$ such that $\ell(v_0, v) = (\mathbf{r} \Rightarrow \mathbf{s})$ is consuming and $\mathbf{m}_a$ matches with $\mathbf{r}$. Otherwise, q is called *non-consuming.* Note that in non-consuming states, honest principals can only take edges with non-consuming rules (including self-loops). In particular, any two equivalent, non-consuming states have the same set of successors. We call a vertex $v = (q, \psi)$ in $G^{\varphi}_{(S,q^0)}$ *non-consuming* if q is non-consuming.

The following definition says that a strategy is $\sim$-uniform if the intruder chooses the same messages whenever he is in certain non-consuming, equivalent states.

Definition 4.7.1 Consider the game $G^{\varphi}_{(S,q^0)}$ as above. A strategy f for Player 0 is *$\sim$-uniform*, if it is memoryless and moreover, for all non-consuming states v, v' with $v \sim v'$ we have that:

(a) If $v = (q, \langle\!\langle A \rangle\!\rangle \psi)$, $v' = (q', \langle\!\langle A \rangle\!\rangle \psi)$ with $f(v) = (q, \Box_c \psi)$ and $f(v') = (q', \Box_{c'} \psi)$, then $c = c'$.

(b) If $v = (q, \Diamond_c \psi)$ and $v' = (q', \Diamond_c \psi)$ then $f(v) = f(v')$.

The following lemma says that it suffices to consider $\sim$-uniform strategies.

Lemma 4.7.2 *If there exists a winning strategy of Player* 0 *in* $G^{\varphi}_{(S,q^0)}$*, then there exists a $\sim$-uniform winning strategy for this player.*

The proof of this lemma is provided in Section 4.7.4. We call a strategy graph induced by a $\sim$-uniform strategy a *$\sim$-uniform strategy graph.*

Lemma 4.7.3 *If F is a $\sim$-uniform strategy graph for Player* 0 *in* $G^{\varphi}_{(S,q^0)}$*, then the length of every path in F starting from the initial vertex and without repetitions has length polynomially bounded in $|Pr| + |\varphi|$. Also, the number of reachable vertices of F is exponentially bounded in $|Pr| + |\varphi|$.*

The proof of this lemma is provided in the Section 4.7.5. Lemma 4.7.2 and 4.7.3 imply that if Player 0 wins the game $G^{\varphi}_{(S,q^0)}$, then one can witness this fact by a strategy graph F with an exponentially bounded number of vertices. However, this does not necessarily mean that the representation of F is bounded exponentially in $|Pr| + |\varphi|$ since the size of states in F, in particular the size of messages in such states, might be big. Fortunately, it is possible to show that the overall size of F can be bounded exponentially, where the size of a strategy graph is defined in the obvious way according to some standard representation, where the set of all messages occurring in F are represented by a single DAG.

Lemma 4.7.4 *If F is a winning strategy graph for Player* 0 *in* $G^{\varphi}_{(S,q^0)}$ *as described in Lemma 4.7.3, then there exists a winning strategy graph F' of (overall) size exponentially bounded in $|Pr| + |\varphi|$.*

The proof of this lemma is provided in Section 4.7.6.

Using the three lemmas just stated, it immediately follows that PAMC(greedy, dssc-containing, $\mathcal{I}$-positive) is in NEXPTIME: By the lemmas, we know that Player 0 wins $G^{\varphi}_{(S,q^0)}$ iff there exists a winning strategy graph for Player 0 of size exponentially bounded in $|Pr| + |\varphi|$. So, we first guess such a graph and then check whether it represents a winning strategy graph for Player 0. The last step can be checked in polynomial time in the size of the graph (see, e.g., [GTW02]). Hence, we have a non-deterministic exponential-time algorithm for the problem PAMC(greedy, dssc-containing, $\mathcal{I}$-positive).

In the following sections, we present the proofs of Lemma 4.7.2 to 4.7.4. However, first, in Section 4.7.2 and 4.7.3 we summarize some useful properties of concurrent game structures and parity games induced by protocols.

4.7.2 Properties of Concurrent Game Structures for Protocols

The following lemma says that $\prec$ as defined above is transitive, where for an instance P we write $\mathsf{root}(P)$ to denote the root of P.

Lemma 4.7.5 *Let S be defined as above and let q, q', q'' be states of S. If $q \prec q'$ and $q' \prec q''$, then $q \prec q''$.*

Proof Let $q = (\mathcal{K}, \overline{P}, \overline{\mathbf{m}}, \overline{s}), q' = (\mathcal{K}', \overline{P}', \overline{\mathbf{m}}', \overline{s}'), q'' = (\mathcal{K}'', \overline{P}'', \overline{\mathbf{m}}'', \overline{s}'')$ be states of $S = \langle \Sigma, Q, \mathbb{P}, \pi, \Delta, \delta \rangle$ such that $q \prec q'$ and $q' \prec q''$. We have to show that $q \prec q''$, i.e., $q \not\sim q''$ and q'' is a descendant of q. Obviously, q'' is a descendant of q. Since $q \not\sim q'$ and $q' \not\sim q''$, one of the following cases must occur. From every case we can conclude that $q \not\sim q''$:

- $P_a \neq P'_a$ or $P'_a \neq P''_a$ for some $a \in \mathcal{H}$: By definition of S and since q' is a descendant of q and q'' is a descendant of q', it follows that $\mathsf{root}(P''_a)$ is a (proper) descendant of $\mathsf{root}(P_a)$ in P_a. In particular, $\mathsf{root}(P_a) \neq \mathsf{root}(P''_a)$. Hence, $q \not\sim q''$.
- $\mathcal{K} \neq \mathcal{K}'$ or $\mathcal{K}' \neq \mathcal{K}''$: Since q' is a descendant of q and q'' is a descendant of q' we know that $\mathcal{K} \subseteq \mathcal{K}' \subseteq \mathcal{K}''$. Thus, we can conclude that $\mathcal{K}$ is a strict subset of $\mathcal{K}''$. Thus, $q \not\sim q''$.
- $(s_c, d_c) \neq (s'_c, d'_c)$ or $(s'_c, d'_c) \neq (s''_c, d''_c)$ for some $c \in \mathsf{Sch}(\mathcal{H}, \mathcal{P}) \cap \mathsf{ch}(Pr)$ (note that $\mathsf{ch}(Pr) \cap \mathsf{Sch}(\mathcal{D}, \mathcal{P}) = \emptyset$ since protocols are dssc-free) and

$P_a = P'_a = P''_a$ for all $a \in \mathcal{H}$: Since $c = \mathsf{sch}(a,b)$ for some honest $a \in \mathcal{H}$ and some $b \in \mathcal{P}$, and $P_a = P'_a = P''_a$ for all $a \in \mathcal{H}$, we know that nothing is written to any scheduled secure channel by a, and hence, $|s_c| \geq |s'_c| \geq |s''_c|$. If $|s_c| > |s''_c|$, we immediately obtain that $q \not\prec q''$. Otherwise, we have $|s_c = s''_c|$ which implies that $d'_c = d''_c = \overline{\mathsf{delivered}}$ and thus $d_c = \mathsf{delivered}$. So, $d_c \neq d''_c$. Thus, $q \not\prec q''$. □

While a computation in the concurrent game structure for a protocol can be infinite, "real progress", which is captured by relation $\prec$, can only be made a bounded number of times during such a computation. This fact is formally stated in the following lemma.

Lemma 4.7.6 *If $q_1 \prec \cdots \prec q_n$, then n is bounded polynomially w.r.t. the size of Pr.*

Proof If $q \prec q'$, then, by definition, one of the following cases holds: q and q' differ on the state of (a) some honest participant, (b) a scheduled secure channel in $\mathsf{Sch}(\mathcal{H},\mathcal{P})$, or (c) the intruder knowledge.

The lemma follows from the following observations: Each honest participant can change his state at most n times, where n is the size of the protocol description, so case (a) can happen only n^2 times. Moreover, each scheduled secure channel in $\mathsf{Sch}(\mathcal{H},\mathcal{P})$ receives only n messages (from honest participants) during the course of a protocol execution, so its state can be changed at most $2n$ times. It means that case (b) can happen at most $2n^2$ times. Now, whenever a state of the intruder is changed, the state of some honest participant or some secure channel has to be changed as well. So, if (c) happens, then so must (a) or (b). □

The following lemma follows immediately from the definition of a consuming state and greedy principals (recall that principals considered here are greedy).

Lemma 4.7.7 *If a state q is consuming and q' is a successor of q, then $q' \prec q$.*

The next lemma formalizes the already mentioned intuition behind non-consuming states (see Section 4.7.1): When in a non-consuming state, an instance ignores incoming messages, and hence, two equivalent, non-consuming states have the same set of successors.

Lemma 4.7.8 *Let q_1, q_2 be non-consuming states with $q_1 \sim q_2$. If a state q is a c-successor of q_1, for $c \in \Delta^A(q_1)$ and some $A \subseteq \Sigma$, then $c \in \Delta^A(q_2)$ and q is also a c-successor of q_2.*

Proof Since $q_1 \sim q_2$, we have $q_1 = (\mathcal{K}, \overline{P}, \overline{\mathbf{m}}_1, \overline{s})$ and $q_2 = (\mathcal{K}, \overline{P}, \overline{\mathbf{m}}_2, \overline{s})$. It is enough to show that if $c \in \Delta^\Sigma(q_1)$, then $c \in \Delta^\Sigma(q_2)$ and $\delta(q_1, c) = \delta(q_2, c)$.

The set of moves of $\mathcal{I}$ depends only on the intruder knowledge ($\mathcal{K}$) which is the same in q_1 and q_2 and the result of applying these moves has the same consequences.

The set of moves of the secure channels depends only on $\overline{s}$ which, again, is the same in q_1 and q_2. Thus, these players have the same moves in q_1 and q_2, and the result of applying these moves has again the same consequences.

The states of an honest participant in q_1 and q_2 are the same, and, because the states are non-consuming, only non-consuming rules (and exactly those) can be applied. But the set of these rules is the same in q_1 and q_2 and the application of these rules does not depend on the current input. Hence, the result of applying these moves has the same consequences. $\square$

4.7.3 Some Properties of Parity Games for Protocols

Given Pr, $S = S_{Pr}$, q^0 (the initial state of S), and an $\mathcal{I}$-positive AMC-formula φ in negation normal form as above, in this section we study the induced parity game $G^\varphi_{(S,q^0)}$. We also state some general properties of parity games. We first note:

Remark 4.7.9 *For each subformula $\langle\langle A\rangle\rangle\psi$ of φ, we have that $\mathcal{I} \in A$ and for each subformula $[\![A]\!]\psi$ of φ, we have that $\mathcal{I} \notin A$. Also, for each subformula $\Box_c\psi$ of φ, the domain of c contains $\mathcal{I}$ and for each subformula $\Diamond_c\psi$ of φ, the domain of c does not contain $\mathcal{I}$.*

The following lemma is a consequence of Lemma 4.7.8.

Lemma 4.7.10 *If $(q_1, \Diamond_c\psi)$ and $(q_2, \Diamond_c\psi)$ are vertices of $G^\varphi_{(S,q^0)}$ where q_1 and q_2 are non-consuming, $q_1 \sim q_2$, and such that (q, ψ) is a successor of $(q_1, \Diamond_c\psi)$, then (q, ψ) is also a successor of $(q_2, \Diamond_c\psi)$. Similarly, if $(q_1, \Box_c\psi)$ and $(q_2, \Box_c\psi)$ are vertices of $G^\varphi_{(S,q^0)}$ with non-consuming states q_1 and q_2, $q_1 \sim q_2$, and (q, ψ) is a successor of $(q_1, \Box_c\psi)$, then (q, ψ) is also a successor of $(q_2, \Box_c\psi)$.*

From Lemma 4.7.5 and 4.7.6, we obtain:

Lemma 4.7.11 *If λ is a play in $G^{\varphi}_{(S,q^0)}$, then λ can be written as a concatenation $\lambda_1 \dots \lambda_n$ where*

- *λ_j for $j = 1, \dots, n-1$ is a finite sequence of states of $G^{\varphi}_{(S,q^0)}$ and λ_n is an infinite sequence of states of $G^{\varphi}_{(S,q^0)}$,*
- *n can be polynomially bounded in the size of Pr,*
- *for all i and states v, u in λ_i we have that $v \sim u$, and*
- *for all $i < j$, u in λ_i, and v in λ_j, we have $u \prec v$.*

Proof First observe that if $(q_1, \psi_1), (q_2, \psi_2), \dots$ is a play, then, for $i < j$, the state q_j is a descendant of q_i, and so, either $q_i \sim q_j$ or $q_i \prec q_j$. Now, the lemma easily follows from Lemma 4.7.5 and 4.7.6. □

We now summarize some useful properties of parity games, independent of the particular game $G^{\varphi}_{(S,q^0)}$.

Lemma 4.7.12 *Let λ be an infinite play in some parity game G. Assume that λ is the concatenation $\lambda_1\lambda_2 \dots$ where each λ_i is a non-empty sequence of vertices such that, for each index i, the maximal color occurring in λ_i is even (odd). Then the maximal color occurring in λ infinitely often is even (odd).*

Proof Let a be the maximal color occurring infinitely often in λ. Because the set of colors is finite, there is an index i_0 such that, for each $i > i_0$, no color $a' > a$ occurs in λ_i. Clearly, there is $j > i_0$ such that a occurs in λ_j. So, a is the maximal color occurring in λ_j which, by assumption, is even (odd). □

Before we proceed, we need to introduce some terminology.

Let $G = (V, V_0, V_1, E, v_I, l)$ be a parity game. A *(finite) path* π in G is a finite sequence of the form $v_1, \dots, v_n$ such that $(v_i, v_{i+1}) \in E$ for every $i = 1, \dots, n-1$. We say that a vertex $v \in V$ is *reachable* in G if there exists a path from v_I to v in G. For $U \subseteq V$, we call $\pi = v_1, \dots, v_n$ a *(finite) U-path* if π is a path and $v_i \in U$ for every $i = 1, \dots, n$. An *infinite path* λ in G is a finite sequence of the form $v_1, v_2, \dots$ such that $(v_i, v_{i+1}) \in E$ for every $i \geq 1$. We call λ winning for Player 0 if the maximum color occurring infinitely often in λ is even; otherwise λ is winning for Player 1. We call λ an *(infinite) U-path* if λ is an infinite path and $v_i \in U$ for every $i \geq 1$.

Let $U \subseteq V$ and let $f : U \to V$ be a function. We call f *consistent* with a path $v_1, \ldots, v_n$ in G if $v_{i+1} = f(v_i)$ for every $i = 1, \ldots, n-1$ with $v_i \in U$; analogously for infinite paths.

Definition 4.7.13 *Consider a parity game (V, V_0, V_1, E, v_I, l). For a set $U \subseteq V$, a U-*strategy for Player 0 *is a function $f : U \cap V_0 \to V$ such that if $f(v) = v'$, then $(v, v') \in E$, and each infinite U-path consistent with f and starting with a state reachable from v_I is winning for Player 0.*

Definition 4.7.14 *Let f be a U-strategy for Player 0, $U \subseteq V$, and $D \subseteq$* dom(f)*. We say that f* gives a good choice for D in a vertex $v \in D$*, if for each U-path $\pi = v_1, \ldots, v_n$ in G consistent with f such that $v_1 = f(v)$, $v_n \in D$, and $v_i \in U \setminus D$, for every $1 \leq i < n$, the maximal color occurring in π is even.*

Lemma 4.7.15 *Let (V, V_0, V_1, E, v_I, l) be a parity game, $U \subseteq V$, f be a U-strategy for Player 0, and D be a non-empty subset of* dom(f)*. Then, there exists a vertex $v \in D$ such that f gives a good choice for D in v.*

Proof For the sake of contradiction, suppose that there is no vertex $v \in D$ such that f gives a good choice for D in v. Let v_0 be some element of D (recall that D is non-empty).

Because f does not give a good choice in v_0, there is a U-path π_1 consistent with f, starting in $f(v_0)$, and ending in some vertex $v_1 \in D$ such that the maximal color occurring in π_1 is odd. Because f does not give a good choice in v_1 we can repeat the argument and obtain a U-path π_2 consistent with f, starting with $f(v_1)$, ending in some vertex $v_2 \in D$ such that the maximal color occurring in π_2 is odd, and so on. In this way, we obtain an infinite path $\lambda = \pi_1\pi_2 \cdots \in U^\omega$ consistent with f such that the maximal color on λ_i is odd, for every i. By Lemma 4.7.12, it follows that λ is winning for Player 1, in contradiction to the assumption that f is a U-strategy for Player 0. □

4.7.4 Proof of Lemma 4.7.2

Assume that there exists a winning strategy f in $G^{\varphi}_{(S,q^0)}$ for Player 0. By Fact 4.1.2, we may assume that this winning strategy is memoryless. Starting with f, we will construct a $\sim$-uniform winning strategy for Player 0.

For $v \in V$, let $[v] = [v]_\sim = \{v' \in V \mid v \sim v'\}$ denote the equivalence class of v (w.r.t. $\sim$). We define $V/_\sim = \{[v]_\sim \mid v \in V\}$ to be the set of equivalence classes of V. For each $\rho \in V/_\sim$, let $f_\rho : \rho \cap V_0 \to V$ be the restriction of f to ρ, i.e., $f_\rho(v) = f(v)$, for each $v \in \mathrm{dom}(f_\rho) = \rho \cap V_0$. (Note that f_ρ is a ρ-strategy for Player 0.)

The outline of the proof is as follows: For each $\rho \in V/_\sim$, we will construct $f'_\rho : \rho \cap V_0 \to V$ such that the function f' defined by $f'(v) = f'_{[v]}(v)$, for every $v \in V_0$, is a $\sim$-uniform winning strategy for Player 0.

For a (possibly non-standard) subformula ψ of φ, we define the set $D_\rho(\psi) = \{v \in \rho \mid v = (q, \psi) \text{ for some non-consuming } q\} \subseteq \rho$.

We say that a function $g_\rho : \rho \cap V_0 \to V$ is $\sim$-uniform w.r.t. a set $D \subseteq \rho \cap V_0$, if for each $v, v' \in D$, (a) and (b) of Definition 4.7.1 hold true for g_ρ.

Now, we construct f'_ρ from f_ρ by iteratively performing the two steps described below, until this is not possible anymore. More precisely, let $f^0_\rho = f_\rho$. Given f^i_ρ, we obtain $f^{i+1}_\rho : \rho \cap V_0 \to V$ either by applying step A or step B below. In the construction we use the fact that each f^i_ρ is a ρ-strategy for Player 0 (see Definition 4.7.13), which will be proven later.

A. Pick a subformula of φ of the form $\Diamond_c \psi$ such that f^i_ρ is not $\sim$–uniform w.r.t. $D = D_\rho(\Diamond_c \psi)$. Note that f^i_ρ is an ρ-strategy for Player 0. Note also that D is a non-empty subset of $\mathrm{dom}(f^i_\rho)$. Thus, by Lemma 4.7.15, there exists a vertex $\tilde{v} \in D$ such that f^i_ρ gives a good choice for D in $\tilde{v}$. We define f^{i+1}_ρ as follows: $f^{i+1}_\rho(v) = f^i_\rho(\tilde{v})$, if $v \in D$, and $f^{i+1}_\rho(v) = f^i_\rho(v)$, otherwise.

B. Pick a subformula of φ of the form $\langle\langle A \rangle\rangle \psi$ such that f^i_ρ is not $\sim$–uniform w.r.t. $D = D_\rho(\langle\langle A \rangle\rangle \psi)$. Note that f^i_ρ is an ρ-strategy for Player 0. Note also that D is a non-empty subset of $\mathrm{dom}(f^i_\rho)$. Thus, by Lemma 4.7.15, there exists a node $\tilde{v} = (\tilde{q}, \langle\langle A \rangle\rangle \psi) \in D$ such that f^i_ρ gives a good choice for D in $\tilde{v}$. We define f^{i+1}_ρ as follows: Let $(\tilde{q}, \Box_{\tilde{c}} \psi) = f^i_\rho(\tilde{v})$. If $v = (q, \langle\langle A \rangle\rangle \psi) \in D$, we set $f^{i+1}_\rho(v) = (q, \Box_{\tilde{c}} \psi)$, and if $v \notin D$, we set $f^{i+1}_\rho(v) = f^i_\rho(v)$.

It is easy to show that if f^i_ρ is $\sim$–uniform w.r.t. $D_\rho(\langle\langle A \rangle\rangle \psi)$ (or $D_\rho(\Diamond_c \psi)$), then f^{i+1}_ρ is also $\sim$-uniform w.r.t. this set. Moreover, if f^{i+1}_ρ is obtained by step B, for some $\langle\langle A \rangle\rangle \psi$, then f^{i+1}_ρ is $\sim$–uniform w.r.t. the set $D_\rho(\langle\langle A \rangle\rangle \psi)$. Hence, because the number of distinct subformulas of φ of this form is bounded by

$|\varphi|$, this step can be done at most $|\varphi|$ times. Similarly, if f_ρ^{i+1} is obtained by step A, for some $\diamondsuit_c\psi$, then f_ρ^{i+1} is $\sim$–uniform w.r.t. the set $D_\rho(\diamondsuit_c\psi)$. The number of subformulas of φ of the form $\diamondsuit_c\psi$ is bounded by $O(|\varphi|\cdot 2^n)$, where n is the size of Pr. Here we use that φ is $\mathcal{I}$-positive, and hence, $\mathcal{I} \notin \mathrm{dom}(c)$ (see Remark 4.7.9), and each participant, except for I, has a bounded number of available moves in each state. Consequently, this step can also only be performed a bounded number of times. Thus, after a bounded number of steps, say k steps, we obtain f_ρ^k which is $\sim$–uniform w.r.t $D_\rho(\psi)$, for each ψ of the form $\langle\!\langle A\rangle\!\rangle\psi'$ or $\diamondsuit_c\psi'$. We define $f'_\rho = f_\rho^k$.

Now, we prove that each f_ρ^i is a ρ-strategy for Player 0. We proceed by induction on i. In case $i = 0$, we have $f_\rho^0 = f_\rho$ and from the definition of f_ρ it follows that f_ρ^0 is a ρ-strategy for Player 0. So, suppose that f_ρ^i is a ρ-strategy for Player 0. We will show that f_ρ^{i+1} is a ρ-strategy for Player 0 as well.

First, it is easy to show that if $f^{i+1}(v) = v'$, then $(v, v') \in E$ (if f^{i+1} is obtained by Step B, then it follows from Lemma 4.7.8 and the definition of $G^{\varphi}_{(S,q^0)}$; if f^{i+1} is obtained by Step A, then we use Lemma 4.7.10).

Now, suppose that λ is an infinite ρ-path consistent with f_ρ^{i+1} and starting with a vertex reachable from the initial state of $G^{\varphi}_{(S,q^0)}$. We consider two cases: (1) There is a suffix λ' of λ such that λ' does not contain vertices in D. In this case, λ' is consistent with f_ρ^i. Thus λ' it is winning for Player 0 and so is λ. (2) λ contains an infinite number of elements in D. In this case we can split λ into $\lambda_0\lambda_1\ldots$ such that the last element of λ_i is the only one in λ_i belonging to D. Let $k > 0$. Let $\lambda_k = v_1, \ldots, v_n$ and v_0 be the last element of λ_{k-1}. So, $v_0, v_n \in D$ and $v_i \notin D$, for $0 < i < n$. Now we consider two cases, depending on whether f_ρ^{i+1} was obtain by step A or B:

- If f_ρ^{i+1} was obtained by step A, then $v_1 = f_\rho^i(\tilde{v})$ is the successor of $\tilde{v}$ for which f_ρ^i gives a good choice. By definition of a good choice for D, the maximal color occurring in $\lambda_k = v_1, \ldots, v_n$ is even.
- If f_ρ^{i+1} was obtained by step B, then v_1 is of the form $(q, \Box_{\tilde{c}}\psi)$, where $(\tilde{q}, \Box_{\tilde{c}}\psi) = f_\rho^i(\tilde{v})$ is the successor of $\tilde{v}$ for which f_ρ^i gives a good choice for D. Because v_2 is a successor of $(q, \Box_{\tilde{c}}\psi)$ and $q, \tilde{q} \in \rho$ are non-consuming, by Lemma 4.7.10, we have that v_2 is also a successor of $(\tilde{q}, \Box_{\tilde{c}}\psi)$. Hence, the path $\tilde{v}, (\tilde{q}, \Box_{\tilde{c}}\psi), v_2, \ldots, v_n$ is consistent with f_ρ^i. Note also that $(\tilde{q}, \Box_{\tilde{c}}\psi)) \notin D$. So, by the definition of a good choice, the maximal

color on $(\tilde{q}, \Box_{\tilde{c}}\psi), v_2, \ldots, v_n$ is even. Because the colors of $(\tilde{q}, \Box_{\tilde{c}}\psi)$ and $(q, \Box_{\tilde{c}}\psi)$ are the same, the maximal color on $\lambda_k = v_1, \ldots, v_n$ is also even.

So, in the both cases, we have proven that the maximal color on λ_k is even. Thus, by Lemma 4.7.12, the path $\lambda_1\lambda_2\ldots$ is winning for Player 0 and so is λ.

This shows that f_ρ^{i+1} is a ρ-strategy for Player 0. In particular, f'_ρ is a ρ-strategy for Player 0. We define $f'(v) = f'_{[v]}(v)$ for every $v \in V_0$. It remains to show that f' is a $\sim$-uniform winning strategy for Player 0.

We know that f'_ρ is $\sim$-uniform w.r.t $D_\rho(\psi)$, for each ψ of the form $\langle\!\langle A \rangle\!\rangle\psi'$ or $\Diamond_c\psi'$. From this it is easy to conclude that f' is $\sim$-uniform.

To prove that f' is winning for Player 0, let us consider some play λ consistent with f'. By Lemma 4.7.11, there is a suffix λ' of λ such that λ' is an infinite ρ-path for some equivalence class ρ. Because λ' is consistent with f' and, for each $v \in \rho$, we have that $f'(v) = f'_\rho(v)$, the infinite path λ' is consistent with f'_ρ as well. Since f'_ρ is an ρ-strategy, we can conclude that λ' is winning for Player 0, and hence, so is λ. $\Box$

4.7.5 Proof of Lemma 4.7.3

Given Pr, $S = S_{Pr}$, q^0 (the initial state of S), and an $\mathcal{I}$-positive AMC-formula in negation normal form as above, and $G^\varphi_{(S,q^0)}$ in this section we proof Lemma 4.7.3.

We first show that the branching degree of $G^\varphi_{(S,q^0)}$ is bounded exponentially. This is true independent of whether or not the underlying strategy is $\sim$-uniform.

Lemma 4.7.16 *The branching degree of a strategy graph of Player* 0 *of* $G^\varphi_{(S,q^0)}$ *is exponentially bounded* $|Pr|$.

Proof By the definition of $G^\varphi_{(S,q^0)}$, the only vertices of $G^\varphi_{(S,q^0)}$ which can have more than two successors are of the form $(q, \langle\!\langle A \rangle\!\rangle\psi)$, $(q, [\![A]\!]\psi)$, $(q, \Diamond_c\psi)$, and $(q, \Box_c\psi)$. Vertices of the form $(q, \langle\!\langle A \rangle\!\rangle\psi)$ and $(q, \Diamond_c\psi)$ belong to V_0, so in any strategy graph for Player 0 these vertices have exactly one successor. Hence, it suffices to check that the number of successors in $G^\varphi_{(S,q^0)}$ of vertices of the form $(q, [\![A]\!]\psi)$ and $(q, \Box_c\psi)$ is exponentially bounded in $|Pr|$.

Let us first consider the case of $v = (q, [\![A]\!]\psi)$. Because φ is $\mathcal{I}$-positive, we know that $\mathcal{I} \notin A$. Hence, A contains only some honest principals and some secure channels. Each successor of v corresponds to some combination of moves of players in A, so the number of successors of v is $|\Delta_q^A|$. Since the number of moves available to each honest principal and to each secure channel is linear in $|Pr|$, $|\Delta_q^A|$ is exponentially bounded in $|Pr|$.

Now, let us consider the case of $v = (q, \Box_c\psi)$. Because φ is $\mathcal{I}$-positive, we have that $\mathcal{I} \in \mathrm{dom}(c)$. Each successor of v corresponds to some c-successor of q, i.e. to some combination of moves of players in $B = \Sigma \setminus \mathrm{dom}(c)$. Hence, one can identify every such successor with exactly one element of Δ_q^B. From the previous case, we know already that $|\Delta_q^B|$ is exponentially bounded in $|Pr|$. □

The following lemmas states that in a path in $G^{\varphi}_{(S,q^0)}$ one cannot stay long in a state with the same first component, i.e., the same state of S, without repeating the second component, i.e., the subformula of φ.

Lemma 4.7.17 *For every path $(q, \psi_1), \ldots, (q, \psi_n)$ in $G^{\varphi}_{(S,q^0)}$ with $\psi_i \neq \psi_j$, for every $i, j \in \{1, \ldots, n\}$, $i \neq j$, it holds that $n \leq 2|\varphi| + 1$.*

Proof If ψ_i is non-standard, then ψ_{i+1} is standard (by the definition of $\mathrm{Sub}^q_S(\varphi)$, symbols $\Box_c$ and $\Diamond_c$ are not nested). Thus, at least $\llcorner n/2 \lrcorner$ formulas amongst $\psi_1 \ldots, \psi_n$ are standard. Because there is at most $|\varphi|$ standard subformulas of φ, we can conclude that $n \leq 2|\varphi| + 1$. □

The following lemma states properties of paths consisting of equivalent states.

Lemma 4.7.18 *Let $v_1, \ldots, v_n$ be a path in $G^{\varphi}_{(S,q^0)}$ such that $v_i \sim v_j$, for every $i, j \in \{1, \ldots, n\}$. Then:*

1. *For each $i, j \in \{1, \ldots, n-1\}$, if $v_i = (q_i, \Box_c\psi)$ and $v_j = (q_j, \Box_c\psi)$, then $v_{i+1} = v_{j+1}$.*
2. *For each $i, j \in \{1, \ldots, n-2\}$, if $v_i = (q_i, [\![A]\!]\psi)$ and $v_j = (q_j, [\![A]\!]\psi)$, then $v_{i+1} = (q_i, \Diamond_c\psi)$ and $v_{j+1} = (q_j, \Diamond_c\psi)$, for some c.*

Proof In both cases, the choices made in v_i and v_j represent choices of some honest principals and some scheduled secure channels, but not choices of the intruder. These choices cannot change the state of these agents: in case 1, this is because $v_i \sim v_{i+1}$ and $v_j \sim v_{j+1}$, and in case 2, it is because $v_i \sim v_{i+2}$ and $v_j \sim v_{j+2}$. So, the choices are uniquely determined and correspond to the choice of staying in the same state for honest players, and to the choice of not delivering any message by the scheduled secure channels (recall that since the protocol is assumed to be dssc-free, these scheduled secure channels only get their messages from honest principals). $\square$

In what follows, we call a vertex v of $G^{\varphi}_{(S,q^0)}$ *modal* if it is of the form $(q, \langle\!\langle A \rangle\!\rangle\psi)$ or $(q, [\![A]\!]\psi)$.

Lemma 4.7.19 *Let $v_1, \ldots, v_n$ be a path in a $\sim$-uniform strategy graph for Player 0 in the game $G^{\varphi}_{(S,q^0)}$ such that $v_i \sim v_j$, for every $i, j \in \{1, \ldots, n\}$, and the vertices $v_1, \ldots, v_n$ are non-consuming. Let, for some $i, j \in \{1, \ldots, n-2\}$, the vertices v_i and v_j be modal and of the form $v_i = (q_i, \psi)$ and $v_j = (q_j, \psi)$. Then, we have $v_{i+2} = v_{j+2}$.*

Proof We consider two cases:

Case 1: $\psi = \langle\!\langle A \rangle\!\rangle\psi'$. By the definition of $\sim$-uniform strategy, there exists c such that $v_{i+1} = (q_i, \Box_c\psi)$ and $v_{j+1} = (q_j, \Box_c\psi)$. Thus, by Lemma 4.7.18, we obtain $v_{i+2} = v_{j+2}$.

Case 2: $\psi = [\![A]\!]\psi'$. By Lemma 4.7.18, there exists c such that $v_{i+1} = (q_i, \Diamond_c\psi)$ and $v_{j+1} = (q_j, \Diamond_c\psi)$. Thus, we obtain $v_{i+2} = v_{j+2}$ by the definition of $\sim$-uniform strategy. $\square$

Let H be a $\sim$-uniform strategy graph of Player 0 in $G^{\varphi}_{(S,q^0)}$. A path $v_1, \ldots, v_n$ in H is *conservative*, if, for all $i \neq j$ we have that $v_i \neq v_j$ and $v_i \sim v_j$.

Lemma 4.7.20 *Let $\pi = v_1, \ldots, v_n$ be a conservative path in a $\sim$–uniform strategy graph of Player 0 in the game $G^{\varphi}_{(S,q^0)}$. If v_1 is consuming, then $n \leq 2|\varphi| + 1$.*

Proof Assume that v_i is of the form (q_i, ψ_i) for all $i = 1, \ldots, n$. Let k be maximal such that $q_i = q_1$ for all $i \leq k$. By Lemma 4.7.17, $k \leq 2|\varphi| + 1$. We will show that $n = k$, which gives $n \leq 2|\varphi| + 1$. By the sake of contradiction, suppose that $k < n$. So, $q_{k+1} \neq q_k$, and thus q_{k+1} must be a successor of q_k. By the assumption, the state $q_k = q_1$ is consuming, so by Lemma 4.7.7, we have $q_k \prec q_{k+1}$, which contradicts the assumption that π is conservative. $\square$

Lemma 4.7.21 *Let $v_1, \ldots, v_n$ be a conservative path in a $\sim$–uniform strategy graph for Player 0 in the game $G^{\varphi}_{(S,q^0)}$. If $v_1, \ldots, v_n$ are non-consuming, then the number of modal vertices in $v_1, \ldots, v_n$ is bounded by $|\varphi| + 2$.*

Proof We will show that a modal subformula of φ cannot occur twice in the path $v_1, \ldots, v_{n-2}$, which means that the number of modal vertices in $v_1, \ldots, v_{n-2}$ is bounded by the number of distinct modal subformulas of φ, and hence, is $\leq |\varphi|$. Consequently, the number of modal vertices in $v_1, \ldots, v_n$ is bounded by $|\varphi| + 2$.

Suppose that, for $i, j \leq n-2$ we have $v_i = (q_i, \psi)$ and $v_j = (q_j, \psi)$, for a modal formula ψ. By Lemma 4.7.19, $v_{i+2} = v_{j+2}$, which contradicts the assumption that $v_1, \ldots, v_n$ is conservative. □

Lemma 4.7.22 *Let $\pi = v_1, \ldots, v_n$ be a conservative path in a $\sim$–uniform strategy graph of Player 0 in the game $G^{\varphi}_{(S,q^0)}$. If $v_1, \ldots, v_n$ are non-consuming, then $n \leq p(|\varphi|)$ for some fixed polynomial p in $|\varphi|$.*

Proof We split π into $\pi_0, \ldots, \pi_m$ such that for each u, v in π_i the first components of u and v are the same and, if $v = (q_v, \psi_v)$ is the last element of π_i and $u = (q_u, \psi_u)$ is the first element of π_{i+1}, then $q_v \neq q_u$.

For $0 \leq i < m$, the second component of the last element of π_i has to be a non-standard formula. Moreover, for $0 < i \leq m$, the second component of the first element of π_i is standard. Hence, for $0 < i < m$, π_i has at least two elements. A predecessor of a vertex with a non-standard formula is modal, so each π_i, for $0 < i < m$, contains a modal element. Hence, by Lemma 4.7.21, we obtain $m \leq |\varphi| + 4$. By Lemma 4.7.17, the length of each π_i is bounded by $2|\varphi| + 1$, so we conclude that $n \leq (2|\varphi| + 1)(|\varphi| + 4) =: p(|\varphi|)$. □

Now, we are ready to prove Lemma 4.7.3. First, by Lemma 4.7.16, the branching degree of a strategy graph of Player 0 is exponentially bounded. Below, we show that every path in $G^{\varphi}_{(S,q^0)}$ starting from the initial state of $G^{\varphi}_{(S,q^0)}$ without repetitions has length polynomially bounded in $|Pr| + |\varphi|$. This shows the first part of Lemma 4.7.3 and from this, together with the bounded branching degree, the lemma follows.

Let π be a path without repetitions in a $\sim$–uniform strategy graph of Player 0. Let $v_1, \ldots, v_n$ be a subsequence of π such that $v_i \sim v_j$, for $1 \leq$

$i, j \leq n$. By Lemma 4.7.11, the number of maximal subsequences of π of this type is at most polynomial in $|Pr|$. Thus, to prove a polynomial bound on the length of π, it is enough to show that n is polynomially bounded in $|Pr| + |\varphi|$.

Observe that $v_1, \dots, v_n$ is conservative. Thus, if $v_1, \dots, v_n$ does not contain any consuming vertex, then, by Lemma 4.7.22, n is polynomially bounded. Otherwise, let k be the smallest index such that v_k is consuming. By Lemma 4.7.22, k is polynomially bounded in $|\varphi|$ and, by Lemma 4.7.20, $n - k$ is polynomially bounded in $|\varphi|$ as well. Hence, we obtain a polynomial bound on n as desired. $\square$

4.7.6 Proof of Lemma 4.7.4

Given Pr, $S = S_{Pr}$, q^0 (the initial state of S), and an $\mathcal{I}$-positive AMC-formula φ in negation normal form as above, and $G^{\varphi}_{(S,q^0)}$ in this section we proof Lemma 4.7.4. To do so, we need to bound the size of messages used in a winning strategy graph. The main idea is to (iteratively) replace certain (unnecessarily big) messages by new atoms in such a way that the resulting graph is still a winning strategy graph. For this purpose, we first characterize the set $d(E)$ of terms derivable from E in terms of what we call intruder rules and study how the replacement of terms by other terms effects the derivability of terms.

For a term t the set $\mathrm{Sub}(t)$ of *subterms* of t is defined as usual. We extend $\mathrm{Sub}(\cdot)$ to sets of terms, multi terms, a-rules and a-instance for $a \in \mathcal{H}$, and protocols as expected.

The intruder rules that we use include those introduced in [RT03]. In addition, we need rules for hashing, signatures, and generating new atoms. In what follows, we often write E, m and m, m' instead of $E \cup \{m\}$ and $\{m, m'\}$, respectively.

An intruder rule L is of the form $E \to m$ where E is a finite set of messages and m is a message. A rule of this form is also called m-rule since m is generated. Given a set E', L can be *applied to* E' if $E \subseteq E'$. The rule L induces a binary relation $\to_L$ on finite sets of messages: $\to_L = \{(E', E' \cup \{m\}) \mid L \text{ can be applied to } E'\}$. If $\mathcal{L}$ is a set of intruder rules, then $\to_{\mathcal{L}} = \bigcup_{L \in \mathcal{L}} \to_L$. For a binary relation $\to$ we write $E \to E'$ instead of $\to (E, E')$. The reflexive and transitive closure of $\to$ is denoted by $\to^*$.

To characterize $d(E)$, we consider the following set of intruder rules. In what follows, the notion "intruder rule" will always refer to the rules introduced below. This set is partitioned into decomposition and composition rules. Accordingly, we call a rule decomposition and composition rule, respectively.

Decomposition rules are of one of the following forms, where m and m' are messages and $k \in \mathbb{K}$ (and thus, $k^{-1} \in \mathbb{K}$):

1. $\langle m, m' \rangle \rightarrow m$ and $\langle m, m' \rangle \rightarrow m'$.
2. $\{m\}^s_{m'}, m' \rightarrow m$.
3. $\{m\}^a_k, k^{-1} \rightarrow m$.

Composition rules are of one of the following forms, where m, m' are some messages, $k, k_0, k_1, k_2 \in \mathbb{K}$, and $a_I \in \mathcal{A}_I$:

1. $m, m' \rightarrow \langle m, m' \rangle$.
2. $m, m' \rightarrow \{m\}^s_{m'}$.
3. $m, k \rightarrow \{m\}^a_k$.
4. $m \rightarrow \mathsf{hash}(m)$.
5. $m, k^{-1} \rightarrow \mathsf{sig}(k, m)$.
6. $\rightarrow a_I$.

Let $\mathcal{L}$ denote the set of all (composition and decomposition) rules. It is easy to see that

$$d(E) = \bigcup \{E' \mid E \rightarrow^*_{\mathcal{L}} E'\}.$$

A *derivation* is of the form $E_0 \rightarrow_{L_0} E_1 \rightarrow_{L_1} E_2 \rightarrow_{L_2} \cdots \rightarrow_{L_{n-1}} E_n$ where $E_i \rightarrow_{L_i} E_{i+1}$ for every i. We call n the length of the derivation. We know that for every $m \in d(E)$ there exists n, intruder rules $L_0, \ldots, L_{n-1}$, and sets $E_0, \ldots, E_n$ such that $E_0 = E$, $m \in E_n$, and there is a derivation from E_0 to E_n as above. We call such a derivation a derivation for m of length n. The derivation is *minimal* if no step can be removed such that the resulting sequence is still a derivation for m. Clearly, for every $m \in d(E)$ there exists a minimal derivation. We write $m \in d^c(E)$ if there exists a minimal derivation of m where the last rule is a composition rule. The following fact is well-known (see, e.g., [RT03]).

Lemma 4.7.23 *Let $m \in d(E)$ and let D be a minimal derivation of m from E such that D ends with a decomposition rule. Then, $m \in Sub(E)$.*

From this lemma, we obtain:

Lemma 4.7.24 *Let E be a set of messages and let τ be a message such that $\tau \notin Sub(E)$ and $\tau \notin d^c(E)$. Then for all $m \in d(E)$ we have that $\tau \notin Sub(m)$.*

Proof Let $D = E_0 \to_{L_1} E_1 \cdots \to_{L_n} E_n$ be a derivation of m from $E_0 = E$. Assume, for the purpose of contradiction, that $\tau \in \mathrm{Sub}(m)$. Then, there exists a minimal $i \neq 0$ such that $\tau \in \mathrm{Sub}(E_i)$ since τ is a subterm of E_n. Assume that L_i is an s-rule for some s. Then, τ is a subterm of s. If τ is a proper subterm of s, by the definition of intruder rules, it follows that τ is a subterm of E_{i-1}, in contradiction to the minimality of i. Thus, $\tau = s$ and therefore, $\tau \in d(E)$. Since $\tau \notin d^c(E)$ it follows that all derivations of τ end with a decomposition rule. Hence, by Lemma 4.7.23, $\tau \in \mathrm{Sub}(E)$, in contradiction to the assumption that $\tau \notin \mathrm{Sub}(E)$. Hence, $\tau \notin \mathrm{Sub}(m)$. □

Let

$$\textsc{Derive} = \{(E, m) \mid m \in d(E)\}$$

where E and m are given as DAGs be the *derivation problem*. The following is well-known (see, e.g., [CKRT03]):

Lemma 4.7.25 Derive *can be decided in polynomial time.*

We now study which messages can be derived from a set of messages if certain terms are replaced by other terms.

Definition 4.7.26 *Let t, t' and t'' be terms. By $t_{|t' \to t''}$ we denote the term obtained from t by simultaneously replacing any occurrence of t' in t by t''.*

For a set T of terms we define $T_{|t'\to t''} = \{t_{|t'\to t''} \mid t \in T\}$. For a sequence $s = t_1 \cdots t_n$ of terms the sequence $s_{|t'\to t''}$ is defined by $s_{|t'\to t''} = t_{1|t'\to t''} \cdots t_{n|t'\to t''}$. For a substitution σ we denote by $\sigma_{|t'\to t''}$ the substitution σ' with the same domain as σ and $\sigma'(x) = \sigma(x)_{|t'\to t''}$ for all $x \in \mathrm{dom}(\sigma)$. For a multi message $\mathbf{m} : A \to \mathcal{M}_\circ$ for some $A \subseteq \mathcal{C}$, we denote by $\mathbf{m}_{|t'\to t''}$ the multi message $\mathbf{m}' : A \to \mathcal{M}_\circ$ with $\mathbf{m}'(c) = \mathbf{m}(c)_{|t'\to t''}$ for all $c \in A$. For $C, D \subseteq \mathcal{C}$ and a (C, D)-rule $R = \mathbf{r} \Rightarrow \mathbf{s}$ the rule $R_{|t'\to t''}$ is defined by $R_{|t'\to t''} = \mathbf{r}_{|t'\to t''} \Rightarrow \mathbf{s}_{|t'\to t''}$. For a principal $P = (V, E, r, \lambda, l)$ the principal $P_{|t'\to t''}$ is defined by

$P_{|t'\to t''} = (V, E, r, \lambda, l')$ where for $(v, v') \in E$ the label $l'(v, v')$ is defined by $l'(v, v') = l(v, v')_{|t'\to t''}$. For a state $q = (\mathcal{K}, \overline{P}, \overline{\mathbf{m}}, \{(s_c, d_c)\}_{c\in\mathcal{SC}})$ of S_{Pr} we define

$$q_{|t'\to t''} = (\mathcal{K}_{|t'\to t''}, \{P_{a|t'\to t''}\}_{a\in\mathcal{H}}, \{\mathbf{m}_{a|t'\to t''}\}_{a\in\mathcal{H}}, \{(s_{c|t'\to t''}, d_c)\}_{c\in\mathcal{SC}}) \ .$$

For a formula ψ of the form $\Diamond_c\psi'$ or $\Box_c\psi'$, we set $\psi_{|t'\to t''} = \Diamond_{c'}\psi'$ or $\psi_{|t'\to t''} = \Box_{c'}\psi'$, respectively, where $c'(\mathcal{I}) = c(\mathcal{I})_{|t'\to t''}$, in case $\mathcal{I} \in \mathrm{dom}(c)$, and $c'(a) = c(a)$ for $a \in \mathrm{dom}(c) \setminus \{\mathcal{I}\}$. If ψ is not of the form $\Diamond_c\psi'$ or $\Box_c\psi'$, we define $\psi_{|t'\to t''} = \psi$. For a state $\alpha = (q, \psi)$ of $G^{\varphi}_{(S,q^0)}$ we set $\alpha_{|t'\to t''} = (q_{|t'\to t''}, \psi_{|t'\to t''})$. For a subgraph F of $G^{\varphi}_{(S,q^0)}$ the graph $F_{|t'\to t''}$ is defined in the obvious way.

The following lemma was proved in [KKW04].

Lemma 4.7.27 *Let E be a set of messages and τ, τ' be messages. Then, $\tau \in d^c(E \setminus \{\tau\})$ implies $d(E)_{|\tau\to\tau'} \subseteq d(E_{|\tau\to\tau'} \cup \{\tau'\})$.*

In order to define messages that can be replaced by an intruder atom from $\mathcal{A}_I$, we need to know how variables are substituted in instances. Therefore, we now define substitutions that keep track of this information. More specifically, let

$$\rho = q_0, q_1, \ldots, q_l$$

be a rooted path in the concurrent game structure $S = S_{Pr}$ (induced by protocol Pr), i.e., $q_0 = q^0$ is the initial state of S and q_{i+1} is a successor of q_i as defined in Section 5.1.1. For $i \in \{0, \ldots, l\}$ let

$$q_i = (\mathcal{K}^i, \overline{P}^i, \overline{\mathbf{m}}^i, \overline{s}^i) \ .$$

For $a \in \mathcal{H}$ let

$$P^i_a = (V^i_a, E^i_a, r^i_a, \lambda^i_a, l^i_a) \ .$$

For $i \in \{0, \ldots, l-1\}$ and $a \in \mathcal{H}$ let v^i_a such that

$$(\mathbf{m}^i_a, P^i_a) \overset{v^i_a}{\mapsto} (\mathbf{m}, P^{i+1}_a) \ .$$

Let $l^i_a(r^i_a, v^i_a) = \mathbf{r}^i_a \Rightarrow \mathbf{s}^i_a$. Let $\tau^{\rho}_{i,a}$ be the substitution with domain $\mathcal{V}(\mathbf{r}^i_a)$ such that for all $c \in \mathrm{dom}(\mathbf{r}^i_a) \cap \mathrm{dom}(\mathbf{m}^i_a)$ we have

$$\mathbf{r}^i_a(c)\tau^{\rho}_{i,a} = \mathbf{m}^i_a(c) \ .$$

We inductively define σ_i^ρ by

$$\begin{array}{rcll} \sigma_0^\rho & = & \emptyset & \\ \sigma_{i+1}^\rho & = & \sigma_i^\rho \cup \bigcup_{a\in\mathcal{H}} \tau_{i,a}^\rho & \quad \text{for } i \in \{0,\ldots,l-1\}\ . \end{array}$$

We set $\sigma_\rho = \sigma_l^\rho$. For a substitution σ and terms t and t' we say that t is a *σ-match of t'* ($t \sqsubseteq_\sigma t'$) if t is not a variable and $t\sigma = t'$. Now we define for a message m what it means that m does not match with a rooted path in S or $G^\varphi_{(S,q^0)}$.

Definition 4.7.28 *Let $\rho = q_0, q_1, \ldots, q_l$ be a rooted path in S. A message m* does not match with *ρ if $t \not\sqsubseteq_{\sigma_\rho} m$ for all $t \in Sub(Pr) \cup \mathcal{A}_I$.*

Let $\alpha = \alpha_0, \alpha_1, \ldots, \alpha_l$ be a rooted path in $G^\varphi_{(S,q^0)}$ where $\alpha_i = (q_i, \psi_i)$. Let $0 = i_0 < i_1 < \cdots < i_k \le l$ such that

- $q_{i_k} = q_l$,
- $q_{i_s} \ne q_{i_{s+1}}$ for all $s \in \{0,\ldots,k-1\}$, and
- $q_{i_s} = q_{i_s+1} = \cdots = q_{i_{s+1}-1}$ for all $s \in \{0,\ldots,k-1\}$.

Then $\rho = q_{i_0}, q_{i_1}, \ldots, q_{i_k}$ is a rooted path in S and we call ρ the *S-projection of α*. A message m *does not match with* α if m does not match with the S-projection of α.

The following lemma states that a message that does not match with a rooted path ρ in S can be replaced by a new intruder atom from $\mathcal{A}_I$ and after this replacement one still has a rooted path in S with essentially the same properties. In particular, at the end of the rooted path the intruder can derive exactly the same constants as he could before the replacement.

Lemma 4.7.29 *Let*

$$\rho = q_0, q_1, \ldots, q_l$$

be a rooted path in S. Let τ be a message that does not match with ρ. Furthermore, let $a_I \in \mathcal{A}_\mathcal{I}$ be a constant that does not occur anywhere in ρ or Pr, and define

$$\rho' = q'_0, q'_1, \ldots, q'_l$$

where $q'_j = q_{j|\tau\to a_I}$. Then, the following is true:

1) ρ' is a rooted path in S.

2) For each $j \in \{0,\ldots,l\}$ and $a \in \mathcal{H}$ we have $\Delta(q'_j, a) = \Delta(q_j, a)$.

3) For each $j \in \{0, \ldots, l\}$ we have that $\Delta(q_j, \mathcal{I})_{|\tau \to a_I} \subseteq \Delta(q'_j, \mathcal{I})$.

4) For each $j \in \{0, \ldots, l\}$, $a \in \mathcal{H}$, and $b \in \mathcal{P}$ we have that $\Delta(q'_j, \mathsf{sch}(a,b)) = \Delta(q_j, \mathsf{sch}(a,b))$.

5) We have that $\pi(q_l) = \pi(q'_l)$.

Proof First, assume that τ does not occur as a subterm anywhere in ρ. Then $\rho' = \rho$ and nothing is to show. In what follows, we show the properties claimed for ρ' under the assumption that τ occurs in ρ. The proof is organized in three steps. First, we will prove that the intruder can derive message τ when it first occurs in ρ. Second, with this proved we show some auxiliary claims for the states q'_i. Third, with these auxiliary claims we show claims 1) to 5) from above.

Before starting with the first step described above we introduce some notation. For $i \in \{0, \ldots, l\}$ let

$$q_i = (\mathcal{K}^i, \overline{P}^i, \overline{\mathbf{m}}^i, \overline{s}^i).$$

For $a \in \mathcal{H}$ let

$$P^i_a = (V^i_a, E^i_a, r^i_a, \lambda^i_a, l^i_a) \enspace .$$

For $i \in \{0, \ldots, l-1\}$ and $a \in \mathcal{H}$ let v^i_a such that

$$(\mathbf{m}^i_a, P^i_a) \overset{v^i_a}{\mapsto} (\mathbf{m}, P^{i+1}_a) \enspace .$$

Let $l^i_a(r^i_a, v^i_a) = \mathbf{r}^i_a \Rightarrow \mathbf{s}^i_a$. We also introduce primed versions of these symbols, for example, $q'_i = (\mathcal{K}'^i, \overline{P}'^i, \overline{\mathbf{m}}'^i, \overline{s}'^i)$.

First step Now we show that the intruder can derive τ, even with a composition rule at the end of a minimal derivation, when it first occurs in ρ. We call (*) the property of τ that $t \not\sqsubseteq_{\sigma_l} \tau$ for all $t \in \mathrm{Sub}(Pr) \cup \mathcal{A}_I$.

We know that there is $i \in \{0, \ldots, l\}$ such that τ occurs in q_i. Let $p \in \{0, \ldots, l\}$ be minimal such that τ occurs in q_p. We first show the following three claims:

i) $p > 0$,

ii) there is a channel c of the form $\mathsf{net}(e, a)$ or $\mathsf{dir}(d, a)$ where $a \in \mathcal{H}$, $d \in \mathcal{D}$, and $e \in \mathcal{P}$ such that $\tau \in \mathrm{Sub}(\mathbf{m}^p_a(c))$, and

iii) τ does not occur in components of q_p other then those described in ii).

Claim i): From (*) it follows that $\tau \notin \mathrm{Sub}(Pr) \cup \mathcal{A}_I$. Since initially there are no messages on secure channels and no messages on channels to honest instances τ does not occur in q_0.
Claim ii) and iii): We show that τ can not occur in components of q_p other than those described in ii). First assume that τ occurs in some scheduled secure channel, i.e., there is $c \in \mathcal{SC}$ such that s_c^p contains τ. Let m be a message in s_c^p that contains τ. Thus, m has to be sent in a step before step p by an honest instance, i.e., there is $s < p$ such that $m = \mathsf{s}_a^s\sigma_s$. From (*) it follows that there is a variable x in the domain of σ_s such that $\tau \in \mathrm{Sub}(\sigma_s(x))$. By definition of a-instances x occurs in step s or before in ρ. From this we get that τ occurs in a step before step p. This contradicts the minimality of p. Second, assume that τ occurs in $\mathcal{K}_p$. Since all messages in $\mathcal{K}_p$ are in $\mathcal{K}_0$ or are sent by honest principals to the intruder by a similar argument to the one used in the first case we get a contradiction to the minimality of p. Third, assume that τ occurs in an instance of q_p. By a similar argument to the one used in the first case we get a contradiction to the minimality of p. This concludes the proof of Claim ii) and iii).

Now we can show that (**) $\tau \in d^c(\mathcal{K}_p)$. From ii) and iii) it follows that $\tau \notin \mathrm{Sub}(\mathcal{K}_p)$ and $\tau \in \mathrm{Sub}(d(\mathcal{K}_p))$. By Lemma 4.7.24 we get that $\tau \in d^c(\mathcal{K}_p)$.

Second step We now do the second step of the proof, i.e., we show auxiliary claims about q_i' needed to prove claims 1) to 5). More precisely, by induction on $0 \leq j \leq l$ using (**) from above we will show that the following claims hold:

a) q_j' is a state of S,

b) for each $a \in \mathcal{H}$ and $j < l$ we have $\Delta(q_j', a) = \Delta(q_j, a)$,

c) $d(\mathcal{K}_j)_{|\tau \to a_I} \subseteq d(\mathcal{K}_j')$,

d) for each $a \in \mathcal{H}$ and $b \in \mathcal{P}$, with $\mathsf{sch}(a, b) \in \mathsf{ch}(Pr)$, we have that $\Delta(q_j', \mathsf{sch}(a, b)) = \Delta(q_j, \mathsf{sch}(a, b))$, and

e) if $j < l$ then q_{j+1}' is a successor of q_j'.

First, assume $j = 0$. Claims a),b),d) are obviously fulfilled since $q_0' = q_0$. To show claim c) we distinguish between two cases:

- $\tau \in d(\mathcal{K}_0)$: By (*) we know that $\tau \notin \mathrm{Sub}(\mathcal{K}_0)$. By Lemma 4.7.24 we get that $\tau \in d^c(\mathcal{K}_0)(= d^c(\mathcal{K}_0 \setminus \{\tau\}))$. By Lemma 4.7.27 we get c).

- $\tau \notin d(\mathcal{K}_0)$: Since we have that $\tau \notin \mathrm{Sub}(\mathcal{K}_0)$, by Lemma 4.7.24, we know that for all $m \in d(\mathcal{K}_0)$ we have that $\tau \notin \mathrm{Sub}(m)$. Thus, we have $d(\mathcal{K}_0)_{|\tau\to a_I} = d(\mathcal{K}_0) = d(\mathcal{K}_{0|\tau\to a_I}) = d(\mathcal{K}'_0)$.

To show claim e) we distinguish between two cases:

- $p > 1$: We have $q'_1 = q_1$ and thus we have that q'_1 is a successor of $q'_0 = q_0$.
- $p = 1$: Choose the ports which carry messages that contain τ. By the points above we have that the intruder can derive these messages and τ does not occur in other components of q_1. Thus, we have that q'_1 is a successor of $q'_0 = q_0$.

For the induction step assume that a) to e) are true for a $j-1$. We want to proof the statements for $j > 0$. Claim a) is fulfilled by induction and claim e) for $j-1$.

To prove claim b) we have to show that $\Delta(q_j, a) = \Delta(q'_j, a)$. For this it suffices to show that if a message m matches with a term t occurring in the left-hand side of a rule in q_j for $a \in \mathcal{H}$, then $m_{|\tau\to a_I}$ matches with $t_{|\tau\to a_I}$ and vice versa. More precisely, let $v \in V^j_a$ be a successor of r^j_a and let $l^j_a(r^j_a, v) = \mathbf{r} \Rightarrow \mathbf{s}$. Let $t = \mathbf{r}(c)$ for some $c \in \mathrm{dom}(\mathbf{r})$.

First, suppose that $m = \mathbf{m}^j_a(c)$ matches with t, i.e., $m = t\sigma$ for some substitution σ. We have to show that $m_{|\tau\to a_I}$ matches with $t_{|\tau\to a_I}$, i.e., we have to show that there is a substitution σ' such that $m_{|\tau\to a_I} = t_{|\tau\to a_I}\sigma'$. Let $t^0 = l^0_a(r^j_a, v)$, i.e., t^0 is the term in $P^0_a = P_a$ that corresponds to t. We have that

$$\begin{aligned}
m_{|\tau\to a_I} &= t\sigma_{|\tau\to a_I}\\
&= (t^0(\sigma_j \cup \sigma))_{|\tau\to a_I}\\
&= (t^0\sigma_l)_{|\tau\to a_I}\\
&\overset{\oplus}{=} t^0(\sigma_{l|\tau\to a_I})\\
&= t^0(\sigma_j \cup \sigma)_{|\tau\to a_I}\\
&= t^0(\sigma_{j|\tau\to a_I} \cup \sigma_{|\tau\to a_I})\\
&= (t^0(\sigma_{j|\tau\to a_I}))\sigma_{|\tau\to a_I}\\
&\overset{\oplus}{=} (t^0\sigma_j)_{|\tau\to a_I}\sigma_{|\tau\to a_I} \qquad = t_{|\tau\to a_I}\sigma_{|\tau\to a_I}
\end{aligned}$$

where all steps are obviously fulfilled by definition, except for the steps marked with $\oplus$: For these steps, we use property (*) from above. Thus, $m_{|\tau\to a_I}$ matches with $t_{|\tau\to a_I}$.

Now, conversely, suppose that $m_{|\tau\to a_I}$ matches with $t_{|\tau\to a_I}$, i.e., $m_{|\tau\to a_I} = t_{|\tau\to a_I}\sigma$ for some substitution σ. We have to show that m matches with t, i.e., $m = t\sigma'$ for some substitution σ'. Using the fact that a_I is a new intruder atom we have that

$$\begin{array}{rcl} m & = & (m_{|\tau\to a_I})_{|a_I\to\tau} \\ & = & (t_{|\tau\to a_I}\sigma)_{|a_I\to\tau} \\ & = & (t_{|\tau\to a_I}{}_{|a_I\to\tau})\sigma_{|a_I\to\tau} \quad = t\sigma_{|a_I\to\tau} \ . \end{array}$$

Thus, m matches with t.

To show claim c) we distinguish between two cases:

- $\tau \in d(\mathcal{K}_j)$: First, assume that $j \geq p$. By (**), we have that $\tau \in d^c(\mathcal{K}_j \setminus \{\tau\})$. By Lemma 4.7.27 we get the desired fact. Second, assume that $j < p$. We have that $\tau \notin \mathrm{Sub}(\mathcal{K}_j)$. Thus, by Lemma 4.7.24, we have that $\tau \in d^c(\mathcal{K}_j \setminus \{\tau\})$. By Lemma 4.7.27 we get the desired fact.
- $\tau \notin d(\mathcal{K}_j)$: In this case we know that $j < p-1$. Thus, we have that $\tau \notin \mathrm{Sub}(\mathcal{K}_j)$. From this, by Lemma 4.7.24, it follows that for every $m \in d(\mathcal{K}_j)$ we have $\tau \notin \mathrm{Sub}(m)$ and therefore

$$d(\mathcal{K}_j)_{|\tau\to a_I} = d(\mathcal{K}_j) = d(\mathcal{K}_{j|\tau\to a_I}) = d(\mathcal{K}'_j) \ .$$

To prove claim d) we have to show that $\Delta(q'_j, \mathsf{sch}(a,b)) = \Delta(q_j, \mathsf{sch}(a,b))$. By definition of q'_j we know that each scheduled secure channel $\mathsf{sch}(a,b)$ contains as many messages as in q_j. Thus, we have $\Delta(q'_j, \mathsf{sch}(a,b)) = \Delta(q_j, \mathsf{sch}(a,b))$.

To prove claim e) we have to show that if $j < l$, then q'_{j+1} is a successor of q'_j. If $j < l$ then we know that q_{j+1} is a successor of q_j since ρ is a path in S. Let $\gamma \in \Delta^{\Sigma}_{q_j}$ be a total move such that $\delta(q_j,\gamma) = q_{j+1}$. Let γ' be defined by $\gamma'(a) = \gamma(a)$ for $a \in \mathcal{H} \cup \mathcal{SC}$ and $\gamma'(\mathcal{I}) = \gamma(\mathcal{I})_{|\tau\to a_I}$. By claims b),c), and d) we get that $\gamma' \in \Delta^{\Sigma}_{q'_j}$. Now we have to show that $\delta(q'_j,\gamma') = q'_{j+1}$, i.e., we have to show that $\delta(q'_j,\gamma') = \delta(q_j,\gamma)_{|\tau\to a_I}$. With similar arguments as used for claim b) one shows that if $(\mathbf{m}^j_a, P^j_a) \overset{v}{\mapsto} (\mathbf{m}, P)$ for some v, $\mathbf{m}$, and P, then $(\mathbf{m}^j_{a|\tau\to a_I}, P^j_{a\,|\tau\to a_I}) \overset{v}{\mapsto} (\mathbf{m}_{|\tau\to a_I}, P_{|\tau\to a_I})$. With this it is easy to see that $\delta(q'_j,\gamma') = \delta(q_j,\gamma)_{|\tau\to a_I}$.

Third step Now we are ready to prove claims 1) to 5) using claims a) to e) from above. Claim 1) is a direct consequence of points a) and e). Claims

2), 3), and 4) follow directly from b), c), and d), respectively. To show claim 5) we first show that for each atom c we have that $c \in d(\mathcal{K}_l)$ iff $c \in d(\mathcal{K}'_l)$. The implication from left to right follows from c) and the fact that $c \neq \tau$. The implication in the other direction, follows from Lemma 4.7.27 if we set E to be $\mathcal{K}_{q'_l}$, set τ (from Lemma 4.7.27) to be a_I, and τ' to be τ (from the lemma proved here). For all other propositional variables p it is obvious that $p \in \pi(q_l)$ iff $p \in \pi(q'_l)$. □

The following lemma states that in paths α of $G^{\varphi}_{(S,q^0)}$ starting in the initial vertex of $G^{\varphi}_{(S,q^0)}$ messages that do not match with α can be replaced by new constants and after this replacement one still has a path in $G^{\varphi}_{(S,q^0)}$.

Lemma 4.7.30 *Let*

$$\alpha = \alpha_0, \alpha_1, \dots, \alpha_l$$

be a rooted path in $G^{\varphi}_{(S,q^0)}$. Let τ be a message that does not match with α. Furthermore, let $a_I \in \mathcal{A}_{\mathcal{I}}$ be a constant that does not occur anywhere in α and Pr, and define

$$\alpha' = \alpha'_0, \alpha'_1, \dots, \alpha'_l$$

where $\alpha'_j = \alpha_{j|\tau \to a_I}$. Then we have that α' is a rooted path in $G^{\varphi}_{(S,q^0)}$.

Proof First, assume that τ does not occur as a subterm anywhere in α. Then $\alpha' = \alpha$ and nothing is to show. Now, assume that τ occurs in α. For $i \in \{0, \dots, l\}$ let $\alpha'_i = (q'_i, \psi'_i)$. Let $\rho = q_{i_0}, q_{i_1}, \dots, q_{i_k}$ be the S-projection of α. By Lemma 4.7.29 we know that $\rho' = q'_{i_0}, q'_{i_1}, \dots, q'_{i_k}$ is a rooted path in S. With statements 2),3), and 4) of Lemma 4.7.29 we can conclude that α' is a rooted path in $G^{\varphi}_{(S,q^0)}$. □

Let F be a finite subgraph of $G^{\varphi}_{(S,q^0)}$ such that the initial vertex α_0 of $G^{\varphi}_{(S,q^0)}$ is present in F and all vertices α in F are reachable from α_0 in F. A vertex $\alpha \in F$ is called *S-maximal* if there is no descendant α' of α in F such that an a-instance in α' differs from an a-instance in α, for some $a \in \mathcal{H}$.

We call a path in $G^{\varphi}_{(S,q^0)}$ *simple* if it is repetition free, i.e., all vertices in this path are pairwise distinct.

Definition 4.7.31 *Let F be a finite subgraph of $G^{\varphi}_{(S,q^0)}$ such that the initial vertex α_0 of $G^{\varphi}_{(S,q^0)}$ is present in F and all vertices α of F are reachable from α_0 in F. Let M be the set of S-maximal vertices in F. Let R be the set of all simple paths in F from α_0 to some vertex in M. Let R' be the set of* *all S-projections of paths in R. Let T be the set of all substitutions σ_ρ with $\rho \in R'$. A message m* does not match with F *if for all substitutions $\sigma \in T$ and all $t \in Sub(Pr) \cup \mathcal{A}_I$ we have that $m \not\sqsubseteq_\sigma t$.*

We now can extend Lemma 4.7.30 to subgraphs of $G^{\varphi}_{(S,q^0)}$.

Lemma 4.7.32 *Let F be a winning strategy graph for Player 0 in $G^{\varphi}_{(S,q^0)}$. Let τ be a message that does not match with F. Let $a_I \in \mathcal{A}_{\mathcal{I}}$ be a constant that does not occur anywhere in F and Pr. Then $F_{|\tau\to a_I}$ is a winning strategy graph for Player 0 in $G^{\varphi}_{(S,q^0)}$.*

Proof Let $F' = F_{|\tau\to a_I}$. We have to show the following two points:

1) F' is a strategy graph for Player 0 in $G^{\varphi}_{(S,q^0)}$.
2) F' is winning for Player 0.

1) By claim 1) of Lemma 4.7.30 we get that F' is a subgraph of $G^{\varphi}_{(S,q^0)}$. Thus, it suffices to show that for all vertices α of Player 1 in F' all successors of $\alpha_{|\tau\to a_I}$ in $G^{\varphi}_{(S,q^0)}$ are present in F'.

Let $\alpha = (q_{|\tau\to a_I}, \psi_{|\tau\to a_I})$ be a vertex of Player 1 in F'. Then, by definition, (q,ψ) is a vertex of Player 1 in F. We distinguish between the different forms of formula ψ. First, if ψ is of the form

$$\psi_1 \wedge \psi_2,\ p,\ \neg p,\ X,\ \mu X.\psi \text{ or } \nu X.\psi,$$

then we have that the only successor of (q,ψ) in $G^{\varphi}_{(S,q^0)}$ is of the form (q,ψ') for some ψ'. Since F is a strategy graph for Player 0 we know that (q,ψ') is present in F. Thus, $(q_{|\tau\to a_I}, \psi'_{|\tau\to a_I})$ is present in F'. By definition of $G^{\varphi}_{(S,q^0)}$, we know that $(q_{|\tau\to a_I}, \psi'_{|\tau\to a_I})$ is the only successor of α in F'.

Second, if ψ is of the form $\Box_c\psi'$ or $[\![A]\!]\psi'$, then we know that the choices of players that have to be specified are choices of honest participants of the protocol Pr or scheduled secure channels, because φ is $\mathcal{I}$-positive. Since F is a strategy graph for Player 0 we know that each such choice, there is a unique successor (q',ψ') of (q,ψ) in F. Now, by condition 2) and 4) of Lemma 4.7.29, we can conclude that all successors of α are present in F'.

2) Obviously, it suffices to check that for all vertices $(q,\psi) \in F$ the evaluation of propositional variables in (q,ψ) and $(q_{|\tau\to a_I}, \psi_{|\tau\to a_I})$ is the same. This follows directly from point 5) of Lemma 4.7.29. $\Box$

Now we can prove Lemma 4.7.4. The idea of the proof is to repeatedly apply Lemma 4.7.32 to a given winning strategy graph F for Player 0 with an exponential number of vertices to obtain a winning strategy graph F' for Player 0 in which all messages occurring as a subterm in F' match with F'. By this fact and the exponential number of vertices in F' we obtain the exponential bound of the size of F' as desired.

Proof (Lemma 4.7.4) Let F be a winning strategy graph for Player 0 in the game $G^{\varphi}_{(S,q^0)}$ such that the number of vertices of F is exponentially bounded and for each vertex $\alpha \in F$ the length of any simple path from the initial vertex to α in F is polynomially bounded. Thus, the number of simple paths in F from the initial vertex in F to S-maximal vertices in F is exponentially bounded. By Lemma 4.7.32, we may assume that all messages occurring as subterms in F match with F. Since the number of substitutions as described in Definition 4.7.28 is exponentially bounded, it is easy to see that F can be represented in size exponentially bounded in $|Pr| + |\varphi|$ by representing the set of all messages occurring in F by a single DAG. □

4.7.7 Lower Bound

In this section, we prove that the problem PAMC(greedy, dssc-containing, $\mathcal{I}$-positive) is NEXPTIME-hard. The proof is by reduction from the the exponentially bounded tiling problem, a known NEXPTIME-hard problem.

The *exponentially bounded tiling problem* is defined as follows (see, e.g., [CGLV03]): Given is a finite set U of *tiles*, two relations $H, V \subseteq U \times U$, two tiles $u_0, u_f \in U$, and an integer (encoded in unary) $m > 0$. The question is whether it is possible to tile a $(2^m \times 2^m)$-square so that the horizontal neighbors belong to H, vertical neighbors belong to V, the left-top tile is u_0, and the left-bottom tile is u_f. More formally, the question is whether there exists a function $t : \{0, \dots, 2^m - 1\}^2 \to U$ such that

(i) $\langle t(i,j), t(i+1,j)\rangle \in H$, for all $0 \le i < 2^m - 1$, and $0 \le j \le 2^m - 1$,

(ii) $\langle t(i,j), t(i,j+1)\rangle \in V$, for all $0 \le i \le 2^m - 1$, and $0 \le j < 2^m - 1$,

(iii) $t(0,0) = u_0$ and $t(0, 2^m - 1) = u_f$.

The function t is called a *solution* of the given tiling problem.

Given an instance $\mathcal{T}$ of this problem, i.e., given U, H, V, m, u_0, and u_f as above, we now (efficiently) construct a protocol Pr and an AMC-formula φ

such that $(S_{Pr}, q^0) \models \varphi$ where q^0 is the initial state of φ iff $\mathcal{T}$ has a solution.

The formula (presented as an ATL-formula) is

$$\varphi = \langle\!\langle \mathcal{I} \rangle\!\rangle \Diamond p_c$$

for some propositional variable p_c. Note that φ is independent of $\mathcal{T}$, and hence, is fixed. Therefore and using Theorem 4.1.1, stating φ as an ATL-formula is w.l.o.g.

We now define Pr (which depends on $\mathcal{T}$). The constants used in Pr are c, e, h, v and the elements of U, where the constants e, h, v stand for "equal", "horizontal", and "vertical", and will be used as keys.

Potential solutions of tiling problems will be represented by messages that encode binary trees of depth $2 \cdot m$, using the pairing operator, with elements of U as their leafs. So, every path in such a tree has length $2 \cdot m$. The first m steps of such a path represent an integer i (encoded as bit string of length m) which stands for a column in the $(2^m \times 2^m)$-square. Analogously, the remaining m steps of the path represent an integer j which stands for a row in the $(2^m \times 2^m)$-square. The node the path is leading to represents the tile at position (i, j) in the square.

Following this intuition, we introduce the following notation: For a term s and a sequence $a \in \{0,1\}^*$, we recursively define $s[a]$ as follows: $s[\epsilon] = s$; $s[0a'] = s'[a']$, if $s = \langle s', s'' \rangle$, and otherwise $s[0a']$ is undefined; $s[1a'] = s''[a']$, if $s = \langle s', s'' \rangle$, and otherwise $s[1a']$ is undefined. Furthermore, for a term s and integers $i, j \in \{0, \ldots, 2^m - 1\}$, we write $s[i,j]$ for $s[ab]$, where $a \in \{0,1\}^m$ is the binary representation of i (with leading zeros, if necessary), $b \in \{0,1\}^m$ is the binary representation of j, and ab stands for the concatenation of the m-bit string a and b. Now, a function $t : \{0, \ldots, 2^m - 1\}^2 \rightarrow U$ (thus a potential solution of a tiling problem) can be represented by a term s such that, for each $0 \leq i, j < 2^m$, the expression $s[i,j]$ is defined and $s[i,j] = t(i,j)$. In that case, s is called *the term representation of* t. We call the term $s[i,j]$ (if defined) a *cell of* t.

The honest principals of Pr are $A_0, \ldots A_{2m+1}$. We also have one dishonest principal B, which we call the *the initiator*. (Recall that B will be played by the intruder.) The initial intruder knowledge is U.

The idea is that the initiator guess a solution of $\mathcal{T}$ (encoded by a message) and then the principals $A_0, \ldots A_{2m+1}$ are used to check whether the message is in fact a solution. More precisely, the message is sent by the initiator to A_0, converted in some way by A_0 and sent to A_1 over a direct secure channel, then converted again by A_1 and then sent to A_2 over a direct secure channel, and so on, until the message reaches A_{2m+1}, who will possibly output c to the initiator. The principals $A_0, \ldots, A_{2m+1}$ are defined in such a way that if the message given by the initiator to A_0 is in fact a solution, then no matter what the choices of the A_i are, at the end A_{2m+1} will output c. Otherwise, if the initiator did not send a solution, there will be at least one choice of the A_i such that A_{2m+1} does not output c.

We now describe the behavior of the honest principals in detail: While we do not formally define these principals in terms of trees, doing this is straightforward. We abbreviate messages of the form $\langle m_1, \langle m_2, \langle \cdots \langle m_{n-1}, m_n \rangle \cdots \rangle \rangle \rangle$ by $\langle m_1, \ldots, m_n \rangle$.

- A_0 waits to receive some message m_0 over a network channel from B, the initiator (and hence, the intruder). As a response, A_0 outputs $\{\langle m_0, m_0, m_0, m_0 \rangle\}_e^s$ to A_1 over a direct secure channel.

  Intuitively, m_0 represents a potential solution of $\mathcal{T}$. The purpose of $A_1, \ldots, A_m$ will then be to pick four bit strings $a^1 = a^1_1 \ldots a^1_m$, $a^2 = a^2_1 \ldots a^2_m$, $a^3 = a^3_1 \ldots a^3_m$, and $a^4 = a^4_1 \ldots a^4_m$, where $a^j_i \in \{0, 1\}$ is picked by A_i for $j = 1, \ldots, 4$. Analogously, the purpose of $A_{m+1}, \ldots, A_{2m}$ will be to pick four bit strings $b^1 = b^1_1 \ldots b^1_m$, $b^2 = b^2_1 \ldots b^2_m$, $b^3 = b^3_1 \ldots b^3_m$, and $b^4 = b^4_1 \ldots b^4_m$, where $b^j_i \in \{0, 1\}$ is picked by A_{i+m} for $j = 1, \ldots, 4$. Hence, $A_1, \ldots, A_{2m}$ pick for positions, namely $m_0[a^1 b^1]$, $m_0[a^2 b^2]$, $m_0[a^3 b^3]$, and $m_0[a^4 b^4]$ in the potential solution m_0. The principals are defined in such a way that $a^1 = b^1 = 0^m$, $a^2 = 0^m$, and $b^2 = 1^m$, i.e., the first two positions considered in m_0 are $(0, 0)$ and $(0, 2^m - 1)$. Principal A_{2m+1} will check for these positions whether $m_0[0, 0] = u_0$ and $m_0[0, 2^m - 1] = u_f$. Moreover, we either have that $a^4 = a^3 + 1$ (interpreted as integers) and $b^3 = b^4$, or that $a^3 = a^4$ and $b^4 = b^3 + 1$. In other words, the third and fourth position correspond to two positions in m_0 that are adjacent horizontally or vertically, respectively. Principal A_{2m+1} will use these positions to check whether the tilings at these positions are in a relationship in H or V, respectively.

- Principal A_i, $0 < i \leq m$, in response to the message from A_{i-1}, received over a direct secure channel, sends a message to A_{i+1} over a direct secure channel according to one of the following rules which all have the same priority, say 1, and are explained below:

$$\begin{aligned}
\{\langle\langle x_1,y_1\rangle,\langle x_2,y_2\rangle,\langle x_3,y_3\rangle,\langle x_4,y_4\rangle\rangle\}_e^s &\to \{\langle x_1,x_2,x_3,x_4\rangle\}_e^s,\\
\{\langle\langle x_1,y_1\rangle,\langle x_2,y_2\rangle,\langle x_3,y_3\rangle,\langle x_4,y_4\rangle\rangle\}_e^s &\to \{\langle x_1,x_2,y_3,y_4\rangle\}_e^s,\\
\{\langle\langle x_1,y_1\rangle,\langle x_2,y_2\rangle,\langle x_3,y_3\rangle,\langle x_4,y_4\rangle\rangle\}_e^s &\to \{\langle x_1,x_2,x_3,y_4\rangle\}_h^s,\\
\{\langle\langle x_1,y_1\rangle,\langle x_2,y_2\rangle,\langle x_3,y_3\rangle,\langle x_4,y_4\rangle\rangle\}_h^s &\to \{\langle x_1,x_2,y_3,x_4\rangle\}_h^s
\end{aligned}$$

If A_i does not receive a message from A_{i-1} in the current round, then A_i stays in the same state by performing a self-loop which is defined to have priority 0.

As explained above, we want that $a_i^1 = a_i^2 = 0$. Therefore, for the first two messages ($\langle x_1, y_1\rangle$ and $\langle x_2, y_2\rangle$), all rules pick the left components, x_1 and x_2. As for the last two messages, the first two rules pick the same component. This corresponds to choosing $a_i^3 = a_i^4$. In the third rule, the first component is picked for the third message and the second component for the fourth message. This corresponds to choosing $a_i^3 = 0$ and $a_i^4 = 1$. Note that now the encryption key is h (instead of e). In particular, all A_j, with $i+1 \le j \le m$, can then only choose the last rule which corresponds to picking $a_j^3 = 1$ and $a_j^4 = 0$. Hence, $a^4 = a^3 + 1$.

- Principal A_i, $m < i \le 2m$, in response to the message from A_{i-1}, received over a direct secure channel, sends a message to A_{i+1} over a direct secure channel according to one of the following rules which all have the same priority, say 1, and are explained below:

$$\begin{aligned}
\{\langle\langle x_1,y_1\rangle,\langle x_2,y_2\rangle,\langle x_3,y_3\rangle,\langle x_4,y_4\rangle\rangle\}_e^s &\to \{\langle x_1,x_2,x_3,x_4\rangle\}_e^s,\\
\{\langle\langle x_1,y_1\rangle,\langle x_2,y_2\rangle,\langle x_3,y_3\rangle,\langle x_4,y_4\rangle\rangle\}_e^s &\to \{\langle x_1,x_2,y_3,y_4\rangle\}_e^s,\\
\{\langle\langle x_1,y_1\rangle,\langle x_2,y_2\rangle,\langle x_3,y_3\rangle,\langle x_4,y_4\rangle\rangle\}_h^s &\to \{\langle x_1,x_2,x_3,x_4\rangle\}_h^s,\\
\{\langle\langle x_1,y_1\rangle,\langle x_2,y_2\rangle,\langle x_3,y_3\rangle,\langle x_4,y_4\rangle\rangle\}_h^s &\to \{\langle x_1,x_2,y_3,y_4\rangle\}_h^s,\\
\{\langle\langle x_1,y_1\rangle,\langle x_2,y_2\rangle,\langle x_3,y_3\rangle,\langle x_4,y_4\rangle\rangle\}_e^s &\to \{\langle x_1,x_2,x_3,y_4\rangle\}_v^s,\\
\{\langle\langle x_1,y_1\rangle,\langle x_2,y_2\rangle,\langle x_3,y_3\rangle,\langle x_4,y_4\rangle\rangle\}_v^s &\to \{\langle x_1,x_2,y_3,x_4\rangle\}_v^s,
\end{aligned}$$

If A_i does not receive a message from A_{i-1} in the current round, then A_i stays in the same state by performing a self-loop which is defined to have priority 0.

The intuition behind the rules is similar to the previous case. Here, A_i chooses the bits $b_i^1, \ldots, b_i^4$. If the message received is encrypted by h, then this means that in the previous case two (horizontally) adjacent positions in m_0 were chosen already. So, b_i^3 has to be equal to b_i^4. Otherwise, if the message is encrypted by e, two vertically adjacent positions can be chosen. This is done analogously to the previous case.

- A_{2m+1} receives a message from A_{2m} over a direct secure channel and sends a message to the initiator (and thus, to the intruder) over a network channel according to one of the following rules:

$$\begin{array}{ll}
\{\langle u_0, u_f, a, b\rangle\}_h^s \to c & \text{for each } (a,b) \in H, \\
\{\langle u_0, u_f, a, b\rangle\}_v^s \to c & \text{for each } (a,b) \in V, \\
\{\langle u_0, u_f, a, a\rangle\}_e^s \to c & \text{for each } a \in U.
\end{array}$$

From the explanation given above it should now be clear that the intruder has a strategy to obtain c iff m_0 encodes a solution of $\mathcal{T}$, and hence, iff $\mathcal{T}$ has a solution: Clearly, if $\mathcal{T}$ has, then the intruder can send this solution (encoded as a message) to A_0 and in any case will receive c at the end. Conversely, if m_0 does not have the correct format, i.e., does not encode a binary tree as explained above, then one of the A_i will not be able to apply a rule, and hence, the intruder will not obtain c. If m_0 is a binary tree as required but it nevertheless does not represent a solution of $\mathcal{T}$, then one of the conditions (i) to (iii) will be violated and then there exists a choice of the $A_1, \ldots, A_{2m}$ such that A_{2m+1} will not be able to apply any of the rules available.

We finally note that instead of direct secure channels one could as well use only network channels. In this case, more keys would be used to enforce the intruder to forward messages from one principal to the next as desired.

Chapter 5

Impossibility of Balance—an Axiomatic Approach

As mentioned in the introduction the logic ATL has been used to specify and analyze security properties of cryptographic protocols [KR01, KR02]. In order to take different kinds of behavior of agents into account, for example honest and dishonest behavior, it is common to introduce new agents that represent these behaviors [SM02, KR02].

In this chapter we introduce an extension of ATL called ATL$_{MS}$—ATL with move selectors—which is better suited to describe different behaviors of participants. The main idea is to distinguish between different kind of moves, e.g., an agent may have honest and dishonest moves available in a given state. We utilize this logic to prove an impossibility result about contract signing protocols in an axiomatic and model independent way. We use our result to give an alternative proof of the impossibility result shown by Chadha et al. in [CMSS05].

We first define the syntax and semantics of ATL$_{MS}$. Then we state and prove our impossibility result and show how to prove the result by Chadha et al. using our result.

5.1 ATL with Move Selectors

In this Section we first introduce the kind of concurrent game structures that we need to interpret ATL$_{MS}$formulas and then we define syntax and semantics of ATL_{MS}.

5.1.1 Concurrent Game Structures

Our definition of a concurrent game structure differs from the one in [AHK02] and the one in Chapter 4: As in Chapter 4 the structures that we consider may have an *infinite* state space and in one state players may have an *infinite* number of possible moves which may be an arbitrary value. The concurrent game structures introduced in this section consider propositional variables not only associated with states but also with moves of players of the concurrent game structure. To be able to define the semantics of ATL_{MS}formulas introduced in the next subsection, the transition function of the concurrent game structures has to be defined for cases where players do not have moves of a certain type available. Because of the differences of concurrent game structures utilized in this section to the ones used in [AHK02] and Chapter 4 we define concurrent game structures again.

A *concurrent game structure with propositions on moves (CGS)* is a tuple $\mathcal{C} = (\Sigma, Q, \mathbb{P}, \pi, \Delta, \mathbb{M}, \mu, \delta)$ where

- Σ is a non-empty, finite set of *players*,
- Q is a (possibly infinite) set of *states*,
- $\mathbb{P}$ is a finite set of *propositional variables/propositions*,
- $\pi : Q \to 2^{\mathbb{P}}$ is a *labeling function* (which assigns to every state the set of propositions true in this state),
- Δ is a function which for each state $q \in Q$ and each player $a \in \Sigma$ returns a (possibly infinite) set $\Delta(q, a)$ of *moves* available at state q to player a.

  For $A \subseteq \Sigma$ and $q \in Q$, an (A, q)-*move* is a partial function c that assigns to every $a \in A$ a value $c(a) \in \Delta(q, a)$ or is undefined. Given $A \subseteq \Sigma$ and a state q, we write $\Delta^A(q)$ for the set of (A, q)-moves. An (A, q)-move is called a *partial move* if $A \neq \Sigma$, and a *total move* if $A = \Sigma$.
- $\mathbb{M}$ is a finite set of *move propositions*, i.e., each element of $\mathbb{M}$ is a propositional variable. We assume that $\mathbb{M} \cap \mathbb{P} = \emptyset$.
- μ is a *move labeling function* which assigns to each player $a \in \Sigma$, $q \in Q$, and $m \in \Delta(q, a)$ a set of propositions $\mu_a(q, m)$ from $\mathbb{M}$,

- δ is a *transition function* which, for each state q and each total move $c \in \Delta^{\Sigma}(q)$, returns a state $\delta(q, c) \in Q$ (the state obtained when in state q all players simultaneously perform their moves according to c).

A tuple $\mathcal{C} = (\Sigma, Q, s_I, \mathbb{P}, \pi, \Delta, \mathbb{M}, \mu, \delta)$ is called a *pointed CGS* if $(\Sigma, Q, \mathbb{P}, \pi, \Delta, \mathbb{M}, \mu, \delta)$ is a CGS and $s_I \in Q$. The state s_I is called the *initial state of* $\mathcal{C}$. In what follows definitions are only given for CGS but they are implicitly defined for pointed CGS in the obvious way. A *computation* of $\mathcal{C}$ is an infinite sequence $\lambda = q_0, q_1, \ldots$ of states such that for each $i \geq 0$, the state q_{i+1} is a *successor* of q_i, i.e., $q_{i+1} = \delta(q_i, c)$ for some total move $c \in \Delta^{\Sigma}(q_i)$. We call λ a q-computation for some $q \in Q$ if $q_0 = q$. We refer to the ith state q_i in λ by $\lambda[i]$, to the sequence $q_i, q_{i+1}, \ldots, q_j$ by $\lambda[i, j]$, and to the sequence $q_i, q_{i+1}, \ldots$ by $\lambda[i, \infty]$.

Let $c \in \Delta^{A}(q)$ and $c' \in \Delta^{A'}(q)$ for $A, A' \subseteq \Sigma$ and $q \in Q$ with $A \subseteq A'$. We write $c \sqsubseteq c'$ if $c = c'_{|A}$. For a state q, a set of players $A \subseteq \Sigma$, and an (A, q)-move $c \in \Delta^{A}(q)$, we say that a state $q' \in Q$ is a *c–successor of* q if there is a total move $c' \in \Delta^{\Sigma}(q)$ with $c \sqsubseteq c'$ and $q' = \delta(q, c')$.

For $C \subseteq \Sigma$ we call a function $\alpha : C \to \mathcal{B}(\mathbb{M})$ a *move selector for C* (here $\mathcal{B}(\mathbb{M})$ denotes the set of propositional formulas with variables from $\mathbb{M}$). A move selector for Σ is simply called move selector. For a set $P \subseteq \mathbb{M}$ of move propositions we denote by β_P the evaluation that assigns `true` to all propositional variables in P and `false` to the propositional variables in $\mathbb{M} \setminus P$. A move $m \in \Delta^{A}(q)$ for some $A \subseteq \Sigma$ is called *α-consistent in q for A* if for every $a \in A$ we have that

1. if there is a move $c \in \Delta(q, a)$ such that $\beta_{\mu_a(q,c)} \models \alpha(a)$, then $m(a)$ is defined and $\beta_{\mu_a(q,m(a))} \models \alpha(a)$ and
2. if for all moves $c \in \Delta(q, a)$ we have that $\beta_{\mu_a(q,c)} \not\models \alpha(a)$, then $m(a)$ is undefined.

We call an α-consistent A move m *strict α-consistent* if for all $a \in A$ we have that $m(a)$ is defined. We call a strict α-consistent move of $a \in A$ in state q an *$\alpha(a)$ move in q*. For a state q of $\mathcal{C}$ a successor q' of q is called (strict) α-consistent if there is a (strict) α-consistent move $m \in \Delta^{\Sigma}(q)$ in q (for Σ) with $q' = \delta(q, m)$. A computation $\lambda = q_0, q_1, \ldots$ of $\mathcal{C}$ is (strict) α-consistent if q_{i+1} is a (strict) α-consistent successor of q_i for each i. A finite prefix of a computation is called a *computation segment.* A computation segment $\lambda = q_0, q_1, \ldots, q_{l-1}$ is called (strict) α-consistent if q_{i+1} is a (strict) α-consistent successor of q_i for each $i \in \{0, \ldots, l-2\}$. Note that for all move selectors α and states $q \in Q$ there is an α-consistent q-computation but not necessarily a strict α-consistent q-computation.

5.1.2 ATL with Move Selectors

Here we introduce the formal syntax of ATL_{MS}. One could also define an corresponding extension of ATL* but for our purpose it suffices to extend ATL.

The Logic ATL_{MS}extends ATL in two ways. First, the strategy operators are parameterized by move selectors. These move selectors specify which kind of moves a coalition is allowed to choose in their strategies and against which moves of the antagonists the coalition has to play. Second, quantifiers over moves are introduced that treat moves as first order objects. This quantification over moves allows to specify the relation between different kinds of moves. The subformulas that use quantifiers over moves are called *step formulas* because they describe properties of one step of the underlying concurrent game structure.

Syntax of ATL_{MS}

Formally, an ATL_{MS}formula is one of the following:

(S1) p, for propositions $p \in \mathbb{P}$.

(S2) $\neg\varphi$ or $\varphi_1 \vee \varphi_2$, where φ, φ_1, and φ_2 are ATL_{MS}formulas.

(S3) $\langle\langle A\rangle\rangle_\alpha \bigcirc\varphi$, $\langle\langle A\rangle\rangle_\alpha \Box\varphi$, and $\langle\langle A\rangle\rangle_\alpha \varphi_1 \mathbf{U} \varphi_2$, where $A \subseteq \Sigma$ is a set of agents, α is a move selector and φ, φ_1, and φ_2 are ATL_{MS}formulas.

(S4) A Σ-step formula.

A C-step formula for $C \subseteq \Sigma$ is one of the following:

(ST1) $\neg\varphi$ or $\varphi_1 \vee \varphi_2$, where φ, φ_1, and φ_2 are C-step formulas.

(ST2) $\langle C\rangle_\alpha \bigcirc\psi$ or $[C]_\alpha \bigcirc\psi$, where α is a move selector for C and ψ is an ATL_{MS}formula.

(ST3) $\langle A\rangle_\alpha \varphi$ or $[A]_\alpha\varphi$, where $\emptyset \subset A \subset C$, α is a move selector for A, and φ is a $(C \setminus A)$-step formula.

Semantics of ATL$_{MS}$

We interpret ATL$_{MS}$formulas over CGS with propositions on moves.

A *strategy* for an agent $a \in \Sigma$ is a partial function $f_a : Q^+ \rightarrow \mathcal{M}$ such that for all $\lambda \in Q^*$ and all $q \in Q$ we have that $f_a(\lambda \cdot q) \in \Delta(q, a)$ if $f_a(\lambda \cdot q)$ is defined. A strategy f_a for agent $a \in \Sigma$ is *α-consistent* for a move selector α for $\{a\}$ if for each $\lambda \in Q^*$ and $q \in Q$ such that $\lambda \cdot q$ is α-consistent, then the $\{a\}$ move $c : \{a\} \rightarrow \Delta^{\{a\}}(q)$ with $c(a) = \begin{cases} \text{undefined} & \text{if } f_a(\lambda \cdot q) \text{ is undefined} \\ f_a(\lambda \cdot q) & \text{otherwise} \end{cases}$ is an α-consistent move in q for $\{a\}$.

Given a state $q \in Q$, a set A of agents, and a set $F_A = \{f_a \mid a \in A\}$ of strategies, one for each agent in A, we define the *outcomes* of F_A from q to be the set $out(q, F_A)$ of all q-computations $\lambda = q_0, q_1, q_2, \ldots$ where q_{i+1} is a successor of q_i that is consistent with all strategies in F_A, i.e., q_{i+1} is a c-successor of q_i where c is an A move such that for $a \in A$ the value of $c(a)$ is undefined if $f_a(\lambda[0, i])$ is undefined and $c(a) = f_a(\lambda[0, i])$ otherwise.

As usual in temporal logics we use fair semantics to rule out computations. A *fairness constraint* is a function $\gamma \colon \Sigma \times Q \rightarrow 2^{\mathcal{M}}$ such that for each player $a \in \Sigma$ and state $q \in Q$ we have that $\gamma(a, q) \subseteq \Delta^{\{a\}}(q)$. For a state $q \in Q$, a player $a \in \Sigma$, and a move selector α for a we say that γ *is (a, α)-enabled in q* if there is an α-consistent move m for a in q such that $m \in \gamma(a, q)$. For a computation $\lambda = q_0 q_1 \ldots$ we say that γ *is a-taken in i* if q_{i+1} is an m successor of q_i for a move $m \in \gamma(a, q_i)$. For a move selector α we say that λ is *weakly $\langle\gamma, \alpha\rangle$-fair* if for each player $a \in \Sigma$, there are infinitely many $i \in \mathbf{N}$ such that γ is not (a, α)-enabled in q_i or there are infinitely many $i \in \mathbf{N}$ such that γ is a-taken in q_i. A *fairness condition* Γ is a set of fairness constraints. We say that λ is *weakly $\langle\Gamma, \alpha\rangle$-fair* if λ is weakly $\langle\gamma, \alpha\rangle$-fair for all $\gamma \in \Gamma$.

Intuitively, the semantics of a formula of the form $\langle\!\langle A \rangle\!\rangle_\alpha \psi$ is that the players of coalition A are allowed to use moves in their strategies that are consistent with α and the coalition plays against all the moves of $\Sigma \setminus A$ that are consistent with α.

Given a CGS $\mathcal{C}$, and a fairness condition Γ, we define the semantics of Fair ATL$_{MS}$ as follows:

- For $p \in \mathbb{P}$, we have $q \models_F p$ iff $p \in \pi(q)$.
- $q \models_F \neg\varphi$ iff $q \not\models_F \varphi$.

- $q \models_F \varphi_1 \vee \varphi_2$ iff $q \models_F \varphi_1$ or $q \models_F \varphi_2$.
- $q \models_F \langle\langle A\rangle\rangle_\alpha \bigcirc \varphi$ iff there exists a set F_A of α-consistent strategies, one for each agent in A, such that for all α-consistent, weakly $\langle \Gamma, \alpha\rangle$-fair computations $\lambda \in out(q, F_A)$, we have $\lambda[1] \models \varphi$.
- $q \models_F \langle\langle A\rangle\rangle_\alpha \Box \varphi$ iff there exists a set F_A of α-consistent strategies, one for each agent in A, such that for all α-consistent, weakly $\langle \Gamma, \alpha\rangle$-fair computations $\lambda \in out(q, F_A)$, we have $\lambda[i] \models_F \varphi$ for all $i \geq 0$.
- $q \models_F \langle\langle A\rangle\rangle_\alpha \varphi_1 \mathbf{U} \varphi_2$ iff there exists a set F_A of α-consistent strategies, one for each agent in A, such that for all α-consistent, weakly $\langle \Gamma, \alpha\rangle$-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 $\lambda[j] \models_F \varphi_1$.
- $q \models_F \varphi$ for a Σ-step formula iff $q, \emptyset \models_F \varphi$.
- $q, m \models_F \neg\varphi$ for a (C, q)-move m and a $(\Sigma \setminus C)$-step formula φ iff $q, m \not\models_F \varphi$.
- $q, m \models_F \varphi_1 \vee \varphi_2$ for a (C, q)-move m and $(\Sigma \setminus C)$-step formulas φ_1 and φ_2 iff $q, m \models_F \varphi_1$ or $q, m \models_F \varphi_2$.
- $q, m \models_F \varphi$ for a (C, q)-move m and a $(\Sigma \setminus C)$-step formula φ of the form $\varphi = \langle A\rangle_\alpha \psi$ where $\emptyset \subset A \subset (\Sigma \setminus C)$ iff there there is a strict α-consistent (A, q)-move m' such that $q, m \cup m' \models_F \psi$.
- $q, m \models_F \varphi$ for a (C, q)-move m and a $(\Sigma \setminus C)$-step formula φ of the form $\varphi = [A]_\alpha \psi$ where $\emptyset \subset A \subset (\Sigma \setminus C)$ iff for all strict α-consistent (A, q)-moves m' we have that $q, m \cup m' \models_F \psi$.
- $q, m \models_F \varphi$ for a (C, q)-move m and a $(\Sigma \setminus C)$-step formula φ of the form $\varphi = \langle \Sigma \setminus C\rangle_\alpha \bigcirc \psi$ iff there there is a strict α-consistent $(\Sigma \setminus C, q)$-move m' such that $\delta(q, m \cup m') \models_F \psi$.
- $q, m \models_F \varphi$ for a (C, q)-move m and a $(\Sigma \setminus C)$-step formula φ of the form $\varphi = [\Sigma \setminus C]_\alpha \bigcirc \psi$ iff for all strict α-consistent $(\Sigma \setminus C, q)$-moves m' we have that $\delta(q, m \cup m') \models_F \psi$.

Given a pointed CGS $\mathcal{C}$ with initial state s_I, a fairness condition Γ, and an ATL$_{MS}$ formula φ we write $\mathcal{C} \models_F \varphi$ instead of $s_I \models_F \varphi$.

We use the following common abbreviation: $\Diamond \varphi = \mathtt{true} \mathbf{U} \varphi$. For computations λ we will also use temporal operators such as $\Diamond$ and $\Box$ with the usual LTL-semantics.

If F_A is an α-consistent strategy for coalition A such that for all β-consistent computations $\lambda \in out(q, F_A)$ we have that $\lambda \models_F \varphi$ we say that *F_A is an α-consistent strategy for $\langle\langle A\rangle\rangle_\beta \varphi$ in q.*

5.2 Impossibility of Balance

In this section we prove an impossibility result concerning contract signing protocols in an axiomatic fashion. Our result is inspired by a work of Chadha et al. in [CMSS05]. There it is proved that in every fair, optimistic, timely contract signing protocol an optimistic player suffers a disadvantage against a malicious opponent. The authors prove their result with respect to a concrete communication model that is based on multiset rewriting and they formalize security properties and optimistic behavior of players in this model by looking at the set of traces imposed by a protocol specification. They state that "Any other formalism [...] that leads to an equivalent set of traces would support the same results about protocols." Our approach is to prove an impossibility result like the one mentioned above in a more model independent way, i.e., we try to axiomatize the properties that a protocol and communication model have to fulfill by ATL_{MS}formulas.

To model optimistic behavior of players we utilize move selectors and in order to specify the semantics of optimistic moves we need the quantification over moves in some places.

In our setting a protocol specification is given by a CGS with propositions on moves. We will now describe CGS that represent a contract signing protocol (protocol-CGS) in more detail.

In a protocol-CGS there are at least two players A and B that represent the contract signing parties.

There are at least three types of moves for each of the players A and B. There are *honest moves*, *optimistic moves*, and *silent moves*. The set of honest moves describes the protocol as it is intended. The set of optimistic moves is a subset of the set of honest moves. By the set of optimistic moves one could describe the interaction between the two participating signers in the protocol without the TTP. A silent move represents an action in which a participant does not interact with other participants, for example does not send a message to another participant. In some situations a silent move could be honest and in some situations a silent move could be dishonest.

The following definition formalizes what is said above:

Definition 5.2.1 *A pointed CGS* $\mathcal{C} = (\Sigma, Q, s_I, \mathbb{P}, \pi, \Delta, \mathbb{M}, \mu, \delta)$ *is called a* protocol-CGS *if it satisfies the following conditions*

- $\{A, B\} \subseteq \Sigma$,
- $\{c_A, c_B, \mathit{initial}_A, \mathit{initial}_B\} \subseteq \mathbb{P}$*, the propositions* c_A *and* c_B *indicate that player* A *and* B *have a signed contract, respectively. The propositions* $\mathit{initial}_A$ *and* $\mathit{initial}_B$ *show that player* A *and* B *are in an initial state, i.e., they have not participated in the protocol in an active manner.*
- $\{\mathit{initial}_A, \mathit{initial}_B\} \subseteq \pi(s_I)$,
- $\{\mathit{honest}, \mathit{optimistic}, \mathit{silent}\} \subseteq \mathbb{M}$.

5.2.1 Advantage and Balance

The main result of this chapter is to show that a CGS can not achieve balance for an optimistic player if the CGS fulfils certain conditions. A state of a CGS is unbalanced for an agent if the opponent has complete control over the outcome of the protocol, i.e., the opponent has two strategies one for each outcome he wants to achieve, in this case we say that the agent has an advantage against his opponent. In the case of contract signing protocols the two outcomes an agent may want to achieve are (i) getting a contract and (ii) preventing the other agent of getting a contract. Thus, assuming that propositional variables c_A and c_B indicate that agent A and agent B have a valid contract, respectively, a state q of a CGS is called unbalanced for A if B has two strategies in q (i) a strategy to get to a state in which c_B holds and (ii) a strategy to get to a state in which A can not get to a state where c_A holds without the help of B. The result of this section says something about the ability of a malicious agent to get to a state where he has an advantage against an agent that plays optimistically,i.e., uses only optimistic moves. In the following we introduce formulas that formalize this notion of optimistic advantage.

If player B has a strategy in state q against an optimistic player A to get to a state in which c_B holds we say that B has a *resolve strategy against an* *optimistic* A. Formally this is expressed by $q \models_F$ optResolve(B) where

$$\text{optResolve}(B) \equiv \langle\!\langle B \rangle\!\rangle_{A:\text{o}} \Diamond c_B \ .$$

In this formula and throughout the rest of this chapter we use the following convention for specifying move selectors. The move propositions honest, optimistic, and silent are abbreviated by h, o, and s, respectively. For a given move selector α instead of $\langle\!\langle X \rangle\!\rangle_\alpha \varphi$ we rather specify α directly by writing $x : \beta$ if $\alpha(x) = \beta$ and omit all $x : \texttt{true}$. In the above example $A : \text{o}$ stands for the move selector α with $\alpha(A) = \text{optimistic}$ and $\alpha(x) = \texttt{true}$ for all $x \in \Sigma \setminus \{A\}$.

If in a state q player A can not get to a state in which c_A holds without the help of player B we say that the protocol is *aborted for* B. We express this by $q \models_F$ notWithout(B) where

$$\text{notWithout}(B) \equiv \langle\!\langle\rangle\!\rangle_{B:\text{s}} \Box \neg c_A \ .$$

Here the phrase "not without the help of player B" is expressed by the fact that only silent moves of B are considered.

If player B has a strategy in state q against an optimistic player A to get to state in which the protocol is aborted for B we say that B has an *abort strategy against an optimistic* A. Formally this is expressed by $q \models_F$ optAbort(B) where

$$\text{optAbort}(B) \equiv \langle\!\langle B \rangle\!\rangle_{A:\text{o}} \Diamond \text{notWithout}(B) \ .$$

We say that player B has an *optimistic advantage* against player A in a state q if player B has (i) a resolve strategy against an optimistic A in q and (ii) an abort strategy against an optimistic A in q. This is expressed by $q \models_F$ optAdvantage(B) where

$$\text{optAdvantage}(B) \equiv \text{optResolve}(B) \wedge \text{optAbort}(B) \ .$$

5.2.2 Security Assumptions

In this subsection we formulate security properties needed to state our impossibility result. We need two properties that we impose on protocol-CGS: The protocol-CGS has to be optimistic and timely.

We say that a protocol-CGS is *optimistic* for B if B has an optimistic strategy in s_I against optimistic A to get a contract, i.e., to reach a state in which c_B holds. The optimistic behavior of the players is modeled by only allowing both players to use optimistic moves. Formally, a CGS $\mathcal{C}$ is optimistic for B if $\mathcal{C} \models_F \text{optimistic}(B)$ where

$$\text{optimistic}(B) \equiv \langle\!\langle B \rangle\!\rangle_{\substack{A:\text{o}\\B:\text{o}}} \Diamond c_B \ .$$

A protocol-CGS is *timely* for B if in every state that is reachable from s_I with B only performing honest moves B has a strategy to end the protocol when A quits participating in the protocol run. The fact that A quits participating in the protocol is modeled by the fact that A is allowed to only perform silent moves. For B to end the protocol means that B is in a resolved state for B or in a aborted state for B. Formally, a CGS $\mathcal{C}$ is timely for B if $\mathcal{C} \models_F \text{timely}(B)$ where

$$\text{timely}(B) \equiv \langle\!\langle\rangle\!\rangle_{B:\text{h}} \Box \langle\!\langle B \rangle\!\rangle_{\substack{A:\text{s}\\B:\text{h}}} \Diamond \text{end}(B)$$

and

$$\text{end}(B) \equiv (c_B \vee \text{notWithout}(B)) \ .$$

5.2.3 Additional Axioms

In this subsection we describe additional axioms that describe properties of protocol-CGS that are needed to formulate the impossibility result stated in Theorem 5.2.2.

We assume that no signer has a possibility to get a contract without the other signer participating in the protocol in an active way, i.e., the other signer performs at least one non silent move. This is formalized by the assumption that

$$\text{notWithout}(A) \wedge \text{notWithout}(B)$$

holds in the initial state of a protocol-CGS. In Theorem 5.2.2 we only need the fact that the protocol-CGS satisfies $\text{notWithout}(A)$ where A is the optimistic player.

The next two axioms describe the semantics of optimistic moves of players. The intuition is that if a player is optimistic he waits for an answer of the other participant to come before contacting for example a trusted third party. Thus, if A stays optimistic, then for B performing a silent move should not decrease the power of B against A. More formally, if we assume that a state q is reached by optimistic moves from player A and optimistic or silent moves from player B only, then there is no other participant involved in the protocol thus far. If in state q player B has an abort strategy against an optimistic player A, player B performs a silent move and player A stays optimistic, then player B has an abort strategy against an optimistic A in the successor state of q. This axiom is formalized by the assumption that $s_I \models_F \text{optAbortRemain}(B)$ where

$$\text{optAbortRemain}(B) \equiv \langle\!\langle\rangle\!\rangle_{\substack{A:\text{o}\\ B:\text{o}\vee\text{s}}} \Box(\text{optAbort}(B) \rightarrow \langle\!\langle\rangle\!\rangle_{\substack{A:\text{o}\\ B:\text{s}}} \bigcirc \text{optAbort}(B)) \ .$$

Similarly we assume that under the circumstances described above the existence of an optimistic resolve strategy remains. This is formalized similarly by $s_I \models_F \text{optResolveRemain}(B)$ where

$$\text{optResolveRemain}(B) \equiv \langle\!\langle\rangle\!\rangle_{\substack{A:\text{o}\\ B:\text{o}\vee\text{s}}} \Box(\text{optResolve}(B) \rightarrow \langle\!\langle\rangle\!\rangle_{\substack{A:\text{o}\\ B:\text{s}}} \bigcirc \text{optResolve}(B)).$$

The next additional axiom formalizes the intuition that if only communication between the two signers A and B has taken place and A stays optimistic, i.e., A waits for B's answer before communication with other parties, the existence of an abort strategy of B against a silent A implies the existence of an abort strategy against an optimistic A. This is motivated by the intuition that the only moves that an optimistic A may perform is communication with B and B may ignore messages coming from A and B may use the same strategy as against a silent A. This is formalized by $s_I \models_F \text{silOptAbort}(B)$ where

$$\text{silOptAbort}(B) \equiv \langle\!\langle\rangle\!\rangle_{\substack{A:\text{o}\\ B:\text{o}\vee\text{s}}} \Box(\text{silentAbort}(B) \rightarrow \text{optAbort}(B))$$

and

$$\text{silentAbort}(B) \equiv \langle\!\langle B\rangle\!\rangle_{A:\text{s}} \Diamond \text{notWithout}(B) \ .$$

The next additional axiom formalizes the intuition that an optimistic move of player A does not effect the power of player B against a silent player A. More precisely, if only communication between the two signers A and B has taken place, then for all moves of the players other then A we have that if A makes an optimistic and silent move such that in the next state

B has a silent abort strategy against player A, then for all other optimistic moves of A player B also has a silent abort strategy in the next state. This is formalized by $s_I \models_F$ optSil(B) where

$$\mathrm{optSil}(B) \equiv \langle\!\langle\rangle\!\rangle_{\substack{A:\mathrm{o}\\ B:\mathrm{o}\vee\mathrm{s}}} \Box[A^c](\langle A\rangle_{A:\mathrm{o}\wedge\mathrm{s}} \bigcirc \mathrm{silentAbort}(B) \to [A]_{A:\mathrm{o}} \bigcirc \mathrm{silentAbort}(B)).$$

The next additional axiom describes the semantics of the propositional variable initial$_A$. Intuitively, initial$_A$ should indicate if participant A has already started to participate in the protocol execution. In the initial state initial$_A$ has to be true and stays true until A performs a non silent move. This is formalized by $s_I \models_F$ initial(A) where

$$\begin{array}{rcl}\mathrm{initial}(A) & \equiv & \mathrm{initial}_A \wedge \\ & & \langle\!\langle\rangle\!\rangle\Box(\langle\Sigma\rangle_{A:\neg\mathrm{s}} \bigcirc \mathtt{true} \to [\Sigma]_{A:\neg\mathrm{s}} \bigcirc \neg\mathrm{initial}_A) \wedge \\ & & \langle\!\langle\rangle\!\rangle\Box(\mathrm{initial}_A \to [\Sigma]_{A:\mathrm{s}} \bigcirc \mathrm{initial}_A) \ .\end{array}$$

The last additional axiom is needed for some technical reasons. It states that there are always silent moves and optimistic moves available for B. This is formalized by $s_I \models_F$ moves(B) where

$$\mathrm{moves}(B) \equiv \langle\!\langle\rangle\!\rangle\Box(\langle\Sigma\rangle_{B:\mathrm{o}} \bigcirc \mathtt{true} \wedge \langle\Sigma\rangle_{B:\mathrm{s}} \bigcirc \mathtt{true}) \ .$$

5.2.4 Main Result

The main result of this chapter is a statement about the impossibility of protecting an optimistic participant against a malicious opponent. Formally we state this result in

Theorem 5.2.2 *Let $\mathcal{C}$ be a protocol-CGS and let Γ be a fairness condition such that*

$$\mathcal{C} \models_F \mathit{optimistic}(B) \wedge \mathit{timely}(B) \wedge \varphi_A \wedge \varphi_B \ ,$$

where

$$\begin{array}{rcl}\varphi_A & \equiv & \mathit{notWithout}(A) \wedge \mathit{initial}(A) \\ \varphi_B & \equiv & \mathit{optResolveRemain}(B) \wedge \mathit{optAbortRemain}(B) \wedge \\ & & \mathit{silOptAbort}(B) \wedge \mathit{optSil}(B) \wedge \mathit{moves}(B) \ .\end{array}$$

Then we have

$$\mathcal{C} \models_F \langle\!\langle B\rangle\!\rangle_{\substack{A:o\\ B:o\vee s}} \Diamond\big(\mathit{optAdvantage}(B) \wedge \neg\mathit{initial}_A\big) \ .$$

For the proof of Theorem 5.2.2 we need the following remark.

Remark 5.2.3 *Let p be a state of a CGS $\mathcal{C}$ such that $p \models_F \langle\!\langle\rangle\!\rangle_\alpha \Box\varphi$. Then for every α-consistent successor q of p we have $q \models_F \langle\!\langle\rangle\!\rangle_\alpha \Box\varphi$.* $\Box$

Proof Let $\mathcal{C}$ be a protocol CGS and let Γ be a fairness condition such that

$$\mathcal{C} \models_F \text{optimistic}(B) \wedge \text{timely}(B) \wedge \varphi_B \wedge \varphi_A \ .$$

We have to show that there is a $(B : \text{o} \vee \text{s}, A : \text{o})$-consistent strategy f_B for B such that for all $\langle \Gamma, (B : \text{o} \vee \text{s}, A : \text{o})\rangle$-fair computations $\rho \in \mathit{out}(s_I, f_B)$ we have

$$\rho \models_F \Diamond(\mathit{optAdvantage}(B) \wedge \neg\text{initial}_A) \ .$$

The strategy f_B is constructed from an $(A : \text{o}, B : \text{o})$-consistent strategy f'_B for optimistic(B) in s_I in the following way

$$f_B(q_0 q_1 \dots q_n) = \begin{cases} f'_B(q_0 q_1 \dots q_n) & \text{if } q_n \models \text{initial}_A \text{ and} \\ & \quad \text{there exists a } (A : \text{o} \wedge \text{s})\text{-move in } q_n \\ \text{any } (B : \text{s})\text{-move} & \text{otherwise} \ . \end{cases}$$

Note that since there exists an optimistic move in q_n for B the move $f_B(q_0 q_1 \dots q_n)$ is a $(B : \text{o} \vee \text{s})$-consistent move and thus f_B is an $(A : \text{o}, B : \text{o} \vee \text{s})$-consistent strategy for B.

We have to show that for all $\langle \Gamma, (B : \text{o} \vee \text{s}, A : \text{o})\rangle$-fair computations $\rho \in \mathit{out}(f_B, s_I)$ we have that

$$(*) \ \rho \models_F \Diamond(\mathit{optAdvantage}(B) \wedge \neg\text{initial}_A)$$

holds.

Let $\rho \in \mathit{out}(f_B, s_I)$ be a $\langle \Gamma, (B : \text{o} \vee \text{s}, A : \text{o})\rangle$-fair computation. We show (*) in two steps. First we show that (i) player A eventually makes a $(A : \neg\text{s})$-move in ρ and secondly we show that (ii) after this $(A : \neg\text{s})$-move $\mathit{optAdvantage}(B)$ holds. From (i), (ii), and $\mathcal{C} \models_F$ initial(A) it immediately follows that $\rho \models_F \Diamond(\text{optAdvantage}(B) \wedge \neg\text{initial}_A)$ holds.

To show (i) assume that on ρ player A does not make any $(A : \neg\text{s})$-move, i.e., there is no $i \geq 0$ such that $\rho(i+1)$ is an $(A : \neg\text{s})$-successor of $\rho(i)$. Together with $\mathcal{C} \models_F$ notWithout(A) we get that (**) $\rho \not\models_F \Diamond c_B$. By definition of f_B we know that for all $i \geq 0$ we have $f_A(\rho[0,i]) = f'_A(\rho[0,i])$

thus we know that ρ is a $\langle\Gamma,(A:\mathrm{o},B:\mathrm{o})\rangle$-fair computation from $out(f'_B,s_I)$. Thus $\rho\models_F\Diamond c_B$, this is a contradiction to (**).

We now show (ii), namely, that after the first $(A:\neg\mathrm{s})$-move of A we have that $optAdvantage(B)$ holds. For this let $i\geq 0$ be minimal such that $\rho(i)\models_F\neg\mathrm{initial}_A$, note that by (i) and $\mathcal{C}\models_F\mathrm{initial}(A)$ such an i exist and that $i>0$. We know that in $\rho(i-1)$ player A makes a $(A:\neg\mathrm{s})$-move. Let $m_B=f_B(\rho[0,i-1])$. According to the definition of strategy f_B we distinguish between two cases:

- In state $\rho(i-1)$ there is a $(A:\mathrm{o}\wedge\mathrm{s})$-move: We know that $m_B=f'_B(\rho[0,i-1])$. Since $\rho[0,i]$ is a prefix of a $\langle\Gamma,(A:\mathrm{o})\rangle$-fair computation $\rho'\in out(s_I,f'_B)$ we know that $\rho(i)\models_F\langle\langle B\rangle\rangle_{A:\mathrm{o}}\Diamond c_B$.
  To show that $\rho(i)\models_F\mathrm{optAbort}(B)$ let q be a $(A:\mathrm{o}\wedge\mathrm{s})$-successor of $\rho(i-1)$ for the B move m_B. From $\mathcal{C}\models_F\mathrm{timely}(B)$ we get that $q\models_F\langle\langle B\rangle\rangle_{A:\mathrm{s}}\Diamond\mathrm{end}(B)$. Let λ be a $(A:\mathrm{s})$-computation starting at q. From $\mathcal{C}\models_F\mathrm{notWithout}(A)$ and the fact that $\rho[0,i-1]\lambda$ is a $(A:\mathrm{s})$-computation starting at s_I we get that $\lambda\models_F\Box\neg c_B$. From this we get that each strategy f''_B for $\langle\langle B\rangle\rangle_{A:\mathrm{s}}\Diamond\mathrm{end}(B)$ in q is a strategy for $\langle\langle B\rangle\rangle_{A:\mathrm{s}}\Diamond\mathrm{notWithout}(B)$ in q. From $\mathcal{C}\models_F\mathrm{optSil}(B)$ we get that $\rho(i)\models_F\langle\langle B\rangle\rangle_{A:\mathrm{s}}\Diamond\mathrm{notWithout}(B)$. From $\mathcal{C}\models_F\mathrm{silOptAbort}(B)$ we get $\rho(i)\models_F\mathrm{optAbort}(B)$. Thus, we have $\rho(i)\models_F\mathrm{optAdvantage}(B)\wedge\neg\mathrm{initial}_A$.
- All $(A:\mathrm{o})$-moves in state $\rho(i-1)$ are $(A:\neg\mathrm{s})$-moves: Since $\rho[0,i-1]$ is a prefix of a $\langle\Gamma,(A:\mathrm{o})\rangle$-fair computation $\rho'\in out(s_I,f'_B)$ we know that $\rho(i-1)\models_F\langle\langle B\rangle\rangle_{A:\mathrm{o}}\Diamond c_B$. From $\mathcal{C}\models_F\mathrm{optResolveRemain}(B)$ and the fact that by definition $f_B(\rho[0,i-1])$ is an $(B:\mathrm{s})$-move we get $\rho(i)\models_F\mathrm{optResolve}(B)$. From $\mathcal{C}\models_F\mathrm{timely}(B)$ we know that $\rho(i-1)\models_F\langle\langle B\rangle\rangle_{A:\mathrm{s}}\Diamond\mathrm{end}(B)$. Similar as in the previous case we can conclude that $\rho(i-1)\models_F\mathrm{optAbort}(B)$. From $\mathcal{C}\models_F\mathrm{optAbortRemain}(B)$ and the fact that by definition $f_B(\rho[0,i-1])$ is an $(B:\mathrm{s})$-move we get $\rho(i)\models_F\mathrm{optAbort}(B)$. Thus, we have $\rho(i)\models_F\mathrm{optAdvantage}(B)\wedge\neg\mathrm{initial}_A$. $\Box$

5.3 Impossibility Result of Chadha et al.

In this section we show how Theorem 5.2.2 can be used to prove the impossibility result shown in Chadha et al. [CMSS05] by a straightforward translation of a protocol specification in the model of [CMSS05] into a protocol-CGS.

Our proof of the impossibility result of [CMSS05] is not easier than the one given there. The advantage of our proof is that it clearly shows which part of the proof is model independent and which part depends on the concrete communication and formal model used in [CMSS05].

We will first review the impossibility result shown in [CMSS05] and describe the model used therein. After this, we will explain the translation mentioned above exactly and give a sketch of how to apply Theorem 5.2.2 to prove the result of [CMSS05].

Chadha's Impossibility Result The impossibility result shown in [CMSS05] states that in every fair, optimistic, and timely contract signing protocol an optimistic player suffers a disadvantage against a malicious opponent. The security properties of protocols and the optimistic behavior of players are defined with respect to a concrete formal model, namely the multiset rewriting model (MSR model) introduced for protocol analysis in [CDL$^+$99]. We now give a very brief recap of this model. For a detailed description see [CMSS05].

MSR Model The model used in [CMSS05] is based on multiset rewriting with existential quantification (MSR) introduced in [CDL$^+$99] for protocol analysis. A protocol definition in this model defines the set of all possible execution traces for instances of a protocol.

To specify a protocol in the MSR model a first-order signature has to be chosen. This signature may contain different sorts, for example for public keys or messages. An atomic formula over the chosen signature without free variables is called a *fact*. A *state* is a finite multiset of facts.

To define possible transitions between states *state transition rules* are defined. A state transition rule is of the form

$$F_1, \dots, F_k \longrightarrow \exists x_1 \dots \exists x_j . G_1, \dots, G_n \ .$$

As an example, consider the state $\{P(f(a)), P(b)\}$ and rule $P(x) \longrightarrow \exists z.Q(f(x), z)$. With the ground substitution that assigns b to x the instantiated rule reads $P(b) \longrightarrow \exists z.Q(f(b), z)$. Choosing a new value c and replace $P(b)$ by $Q(f(b), c)$ we obtain the state $\{P(f(a)), Q(f(b), c)\}$.

A set of MSR rules is called a *theory*.

In [CMSS05] a *contract signing protocol* is defined to consist of three parties O (originator), R (responder), and T (trusted third party), and enables O (respectively, R) to obtain R's signature (respectively, O's signature) on some pre-agreed text.

The communication between O and R is modeled by a *network predicate*. The communication between the signers and the TTP is modeled by *TTP channel predicates* that describe the communication between the TTP T and O (respectively, R) over resilient channels.

The behavior of participants are described by so called *role theories* which are theories that satisfy additional properties. In the modelling of participants so called *role state predicates* are used. For a participant A there is a finite number $A_0, \dots, A_n$ of role state predicates. The role state predicate that describe the initial state of participant A is A_0.

A protocol is specified by an MSR theory, which is the disjoint union of theories describing the behavior of participants O, R, and T along with theories that describe so called *timers* which are used to model optimistic behavior of protocol participants. The theories for the dishonest behavior of participants O and R are derived from the theories describing the honest behavior of O and R, respectively, in a generic way. The dishonest behavior of participants allows them to stop participating in the protocol, to gather messages from the communication channels, and to ignore restrictions imposed by timers. An MSR theory describing a contract signing protocol is called a *protocol theory* and is the disjoint union of theories $\mathbf{O}, \mathbf{R}, \mathbf{T_0}, \mathbf{O_{timeouts}}, \mathbf{R_{timeouts}}, \mathbf{T_{timeouts}}, \mathbf{O_{threat}}, \mathbf{R_{threat}}$ specifying the honest behavior of O, R, and T, the timer rules of O, R, and T, and the dishonest behavior of O and R, respectively. The theory $\mathbf{T}$ is defined by the union of $\mathbf{T_0}$ and $\mathbf{T_{timeouts}}$.

Usual abilities of a standard Dolev-Yao intruder to manipulate messages, such as decomposing messages, encrypting messages, etc. can also be modeled by an MSR theory in a natural way. With this theory one can, for example, express which messages the intruder knows in a given state.

Security Properties A *trace from state S* is a chain of nodes, with the root labeled by S, each node labeled by a state, and each edge labeled by a triple $\langle t, \sigma, \mathbf{Q} \rangle$. Here $\mathbf{Q}$ is an element of

$\{\mathbf{O}, \mathbf{R}, \mathbf{T}, \mathbf{O_{timeouts}}, \mathbf{R_{timeouts}}, \mathbf{O_{threat}}, \mathbf{R_{threat}}\}$, $t \in \mathbf{Q}$ is a state transition rule, and σ is a ground substitution. If $\langle t, \sigma, \mathbf{Q}\rangle$ labels the edge from a node labeled by S_1 to a node labeled by S_2, it must be the case that the application of $t\sigma$ to S_1 produces S_2.

In [CMSS05] an interleaving semantics of concurrency is used. Thus, only one participant can perform a step at a time. According to this the notion of a *strategy* is defined by so called *continuation trees* (denoted by *ctr*) that comprises all possible traces starting at some state into a tree. Each edge in this tree of traces corresponds to one transition and thus belong to one of the agents. A *strategy* for some coalition X is then specified by a set E of edges under the control of X that are removed from the continuation tree (denoted by $ctr \setminus E$).

According to [CMSS05] we get the following definition.

Definition 5.3.1 *A coalition X has a strategy from S to reach a state in which some property φ holds if there is a strategy $ctr \setminus E$ from S for coalition X such that all leaf nodes of $ctr \setminus E$ are labeled by states S' that satisfy property $\varphi(S')$.*

We call a strategy $ctr \setminus E$ for coalition X a *minimal strategy* if $ctr \setminus E$ satisfies the following two properties

(M1) If in some node of $ctr \setminus E$ there are outgoing transitions not under the control of X, then there is no outgoing transition under the control of X in $ctr \setminus E$.

(M2) In every node of $ctr \setminus E$ there is at most one outgoing transition under the control of X.

We have the following remark that is used later in our proof of the impossibility result of [CMSS05]. The proof of this remark is straightforward.

Remark 5.3.2 *A coalition X has a strategy from S to reach a state in which some property φ holds iff X has a minimal such strategy.*

In [CMSS05] security properties of contract signing protocols are formulated in terms of the existence of strategies for some coalition of agents. There are four security properties formalized: *fairness*, *optimism*, *timeliness*, and *balance*. Beside these security properties of contract signing protocols different kinds of behaviors of protocol agents are described: *(dis-)honest*, *silent*, and *optimistic* behavior.

In [CMSS05] the authors model optimism by introducing so called *timers*, which are predicates of a special form. Some state transitions rules of role theories depend on the values of these timers and optimistic players are stuck to respect these values. To model that a player A is optimistic the control over the timer predicates of A is given to the opponent B of A. Thus, the opponent can control if A is allowed to perform protocol rules that depend on timers that are timed out.

Alternative Proof of Chadha's Result We now give an alternative proof of the following impossibility result shown in [CMSS05]:

Theorem 5.3.3 *In a fair, optimistic, timely protocol between signers A and B, if A is optimistic, then B has a strategy for reaching a non-initial state S^* such that B has an advantage against A at S^*.*

Our proof of this theorem uses a straightforward translation of protocol specifications in the MSR model into a protocol-CGS and then applies Theorem 5.2.2 to get the desired result.

From an MSR Protocol Specification to a protocol-CGS Let $\mathbf{P}$ be a protocol theory in the MSR model, i.e., $\mathbf{P}$ is the disjoint union of the theories $\mathbf{A}$, $\mathbf{B}$, $\mathbf{A_{timeouts}}$, $\mathbf{B_{timeouts}}$, and $\mathbf{T}$. Let $\mathbf{A_{threat}}$ and $\mathbf{B_{threat}}$ be the theories that are derived from $\mathbf{A}$ and $\mathbf{B}$, respectively, and which describe the dishonest rules of A and B, respectively. Let S_0 be the initial set of facts that describes the initial situation of the protocol execution.

Given $\mathbf{P}$ and S_0 we now define the protocol-CGS

$$\mathcal{C} = \mathcal{C}_{(P,S_0)} = (\Sigma, Q, s_I, \mathbb{P}, \pi, \Delta, \mathbb{M}, \mu, \delta)$$

that corresponds to $\mathbf{P}$ and S_0. As we use real concurrency in our model, i.e., the players in Σ perform steps simultaneously in order to simulate the interleaving semantics of concurrency used in [CMSS05] we have to introduce a scheduler as a player in $\mathcal{C}$, i.e., $\Sigma = \{A, B, T, \text{sch}\}$. The set of states Q consists of all triples of the form $q = (S, \tau, F)$ where S is a reachable state from S_0 in the MSR model, and $\tau, F \in \{A, B, T\}$. We denote S, τ, and F

by *MSR component* of q, *active player* in q, and *fairness component* of q, respectively. Intuitively, the value of τ specifies who of the players A, B, and T is allowed to perform a real step in state q that corresponds to a step in the MSR model. The value of F is used to define the fairness condition that has to be fulfilled by the scheduler sch. This fairness condition ensures that in every computation each of the players A, B, and T performs infinitely many real steps. The initial state s_I is (S_0, T, O) (we use T as the active player in s_I because in a protocol theory that describes an optimistic protocol there is no rule in $\mathbf{T}$ that can be applied to S_0 and thus after the first step of $\mathcal{C}$ the MSR component is still S_0). We set $\mathbb{P} = \{c_A, c_B, \text{initial}_A, \text{initial}_B\}$ and the intuitive meaning of these propositional variables are as described in the definition of a protocol CGS: for a state (S, τ, F) the propositional variable c_A and c_B are in $\pi(S, \tau, F)$ iff A and B have a signed contract in state S, respectively. The variable initial_A and initial_B are in $\pi(S, \tau, F)$ iff the role state predicate A_0 and B_0, respectively, is present in S. The set $\mathbb{M}$ of move propositions is set to be $\mathbb{M} = \{\text{honest}, \text{optimistic}, \text{silent}\}$. The possible moves of the players A, B, and T along with the associated move propositions in state $q = (S, \tau, F)$ depend on S, τ, and the corresponding role theories. To define the move function Δ and the assignment of move propositions μ we distinguish between different cases. For $\tau' \in \{A, B, T\} \setminus \{\tau\}$ there is exactly one move available, i.e., $\Delta((S, \tau, F), \tau') = \{\text{def}\}$. The move def is called the *default move* and it is optimistic, honest and silent, i.e., $\mu_a(q', \text{def}) = \{\text{optimistic}, \text{honest}, \text{silent}\}$ for all $a \in \{A, B, T\}$ and $q' \in Q$ such that $\text{def} \in \Delta(q', a)$. For τ we distinguish between the following cases:

$\tau = B$: There are two types of moves available for B in q. First, there is a *skip move* skip. Second, there are *MSR moves*. MSR moves for B are of the form $m = \langle t, \sigma, \mathbf{Q} \rangle$ where $\mathbf{Q} \in \{\mathbf{B}, \mathbf{B}_{\mathbf{threat}}, \mathbf{B}_{\mathbf{timeouts}}, \mathbf{A}_{\mathbf{timeouts}}\}$, $t \in \mathbf{Q}$, and there is a state S' such that S' is obtained from S by transition $\langle t, \sigma, \mathbf{Q} \rangle$ in the MSR model.

The skip move skip is optimistic, honest, and silent, i.e., $\mu_B(q, \text{skip}) = \{\text{optimistic}, \text{honest}, \text{silent}\}$. MSR moves $m = \langle t, \sigma, \mathbf{Q} \rangle$ with $Q \in \{\mathbf{B}_{\mathbf{timeouts}}, \mathbf{A}_{\mathbf{timeouts}}\}$ are honest moves, i.e., $\mu_B(q, m) = \{\text{honest}\}$. MSR moves of the form $m = \langle t, \sigma, \mathbf{B} \rangle$ are optimistic and honest, i.e., $\mu_B(q, m) = \{\text{optimistic}, \text{honest}\}$. For MSR moves of the form $m = \langle t, \sigma, \mathbf{B}_{\mathbf{threat}} \rangle$ we set $\mu_B(q, m) = \emptyset$.

$\tau = A$: The only moves that are possibly available for A are MSR moves. Note that there may be no move available for A. MSR moves for A are of the form $m = \langle t, \sigma, \mathbf{Q}\rangle$ where $\mathbf{Q} \in \{\mathbf{A}, \mathbf{A_{threat}}\}$, $t \in \mathbf{Q}$, and there is a state S' such that S' is obtained from S by transition $\langle t, \sigma, \mathbf{Q}\rangle$ in the MSR model. MSR moves of the form $m = \langle t, \sigma, \mathbf{A}\rangle$ are optimistic and honest, i.e., $\mu_A(q, m) = \{\text{optimistic}, \text{honest}\}$. For MSR moves of the form $m = \langle t, \sigma, \mathbf{A_{threat}}\rangle$ we set $\mu_A(q, m) = \emptyset$.

$\tau = T$: The only moves that are possibly available for T are MSR moves. Note that there may be no move available for T. MSR moves for T are of the form $m = \langle t, \sigma, \mathbf{Q}\rangle$ where $\mathbf{Q} \in \{\mathbf{T}\}$, $t \in \mathbf{Q}$, and there is a state S' such that S' is obtained from S by transition $\langle t, \sigma, \mathbf{Q}\rangle$ in the MSR model. MSR moves $m = \langle t, \sigma, \mathbf{T}\rangle$ are honest, i.e., $\mu_T(q, m) = \{\text{honest}\}$. (For our purpose it does not matter which value $\mu_T(q, m)$ has.)

For player sch we set $\Delta((S, \tau, F), \text{sch}) = \{A, B, T\}$ and $\mu_{\text{sch}}((S, \tau, F), m) = \emptyset$ for all $m \in \{A, B, T\}$.

Now we define the transition function δ. Mainly, the transition function δ is given by the transitions in the MSR model, i.e., by the moves that are described by the rules in the role theories of the participants. More precisely, let $q = (S, \tau, F) \in Q$ be a state of $\mathcal{C}$ and let c be a total move in $\Delta^{\Sigma}(q)$. Then $\delta(q, c) = (S', \tau', F')$ where S' is the successor of S given by $c(\tau)$ in the MSR model if $c(\tau)$ is defined and is not the skip move and $S' = S$ otherwise. The value of τ' is simply given by $c(\text{sch})$. And the value of F' is F if $\tau' \neq F$ and $z(F)$ otherwise, where $z(A) = B$, $z(B) = T$, and $z(T) = A$.

The fairness condition Γ consists of only one fairness constraint γ, where $\gamma(a, (S, \tau, F)) = \emptyset$ for $a \neq \text{sch}$ and $\gamma(\tau, (S, \tau, F)) = \{F\}$. With this fairness condition and the definition of the transition function we achieve that for every move selector α we have that in every weakly $\langle\Gamma, \alpha\rangle$-fair computation ρ of $\mathcal{C}$ every agent in $\{A, B, T\}$ performs infinitely many real moves, i.e., for each $a \in \{A, B, T\}$ there are infinitely many i such that a is the active player in $\rho(i)$.

It is easy to see that the following remark holds.

Remark 5.3.4 *For a protocol theory* $\mathbf{P}$ *and an initial set of facts* S_0 *the CGS* $\mathcal{C}_{(P,S_0)}$ *is a protocol-CGS.*

Apply Theorem 5.2.2 To apply Theorem 5.2.2 to prove Theorem 5.3.3 we show two things. First, we prove that for every fair, optimistic, and timely protocol in the MSR model the corresponding protocol-CGS fulfils the assumptions of Theorem 5.2.2. Second, we show that a strategy of player B for $\langle\!\langle B \rangle\!\rangle_{\substack{A:\mathrm{o} \\ B:\mathrm{o}\vee\mathrm{s}}} \Diamond\big(\mathrm{optAdvantage}(B) \wedge \neg\mathrm{initial}_A\big)$ in s_I of $\mathcal{C}$ can be transformed into a strategy from S_0 for player B in the MSR model to reach a non initial state S^* where B has an advantage against optimistic A in the MSR model.

We now show that the protocol-CGS $\mathcal{C} = \mathcal{C}_{(P,S_0)}$ fulfils the assumptions of Theorem 5.2.2. The main problem in showing that $\mathcal{C}$ satisfies the assumptions of Theorem 5.2.2 are due to the different notions of strategies in the MSR model and the CGS model. It is not obvious that strategies from one of the model carry over to the other model and vice versa. We will not give detailed proofs but will describe the constructions involved in a non technical manner. From this it should be clear how the real proofs work.

$\mathcal{C} \models_F \mathrm{optimistic}(B)$: According to the definition of optimistic protocols in the MSR-model, see [CMSS05], we have that: If A is honest and B controls the timeouts of both A and B, B has an honest strategy at S_0 such that

- All edges are labeled by transitions in $\mathbf{A} \cup \mathbf{B}$.
- Every leaf node is labeled by a state in which B possesses A's signature.

Let $ctr \setminus E$ be a minimal strategy for B that meets the above properties. We have to define an $(A:\mathrm{o}, B:\mathrm{o})$-consistent strategy f_B for B such that for all $\langle \Gamma, (A:\mathrm{o}, B:\mathrm{o})\rangle$-fair computations $\rho \in out(f_B, s_I)$ we have that $\rho \models_F \Diamond c_B$. Given a segment $q_0 q_1 \dots q_n$ where $q_i = (S_i, \tau_i, F_i)$ we set $f(\lambda)$ to be a silent optimistic move (which by definition always exist) if one of the following conditions are satisfied:

- $\tau_n \neq B$,
- S_n is not a B-node,
- S_n is a leaf.

In case that $\tau_n = B$, S_n is a B-node, and S_n is not a leaf there is exactly one transition present in $ctr \setminus E$ at S_n, note that $ctr \setminus E$ is supposed to be a minimal strategy. This transition corresponds to a rule in $\mathbf{B}$. By the definition of $\mathcal{C}_{(P,S_0)}$ the move in q_n that corresponds to that transition is optimistic. It is easy to see that f_B is a strategy for optimistic(B) in s_I.

$\mathcal{C} \models_F$ timely(B): In [CMSS05] timeliness is defined in a way that it could not directly be transfered to timeliness as understood in our model. To formalize timeliness as understood in [CMSS05] we could take the axiom

$$\langle\langle B \rangle\rangle_{\substack{A:\mathrm{o}\\B:\mathrm{o}}} (\Diamond c_B \wedge \Box \langle\langle B \rangle\rangle_{\substack{A:\mathrm{s}\\B:\mathrm{h}}} \Diamond \mathrm{end}B)$$

instead of the one used so far for timeliness, note that this formula is not an ATL_{MS}formula anymore but an ATL^*_{MS}formula. The proof of Theorem 5.2.2 can easily be adapted if this version of timeliness is used.

$\mathcal{C} \models_F$ notWithout(A): This follows from the fact that in [CMSS05] it is assumed that B will not get a contract if A does not perform at least one step.

$\mathcal{C} \models_F$ initial(A): Follows directly from the definition of $\mathcal{C}$.

$\mathcal{C} \models_F$ optResolveRemain(B): We have to show that for every $(A : \mathrm{o}, B : \mathrm{o} \vee \mathrm{s})$-consistent computation segment $\rho = q_0 q_1 \ldots q_n$ such that $q_n \models_F$ optResolve(B) holds and every $(A : \mathrm{o}, B : \mathrm{s})$-consistent successor q' of q_n we have that $q' \models_F$ optResolve(B). Let $q_n = (S, \tau, F)$ and let f_B be an $(A : \mathrm{o})$-consistent strategy for optResolve(B) in q_n. We distinguish between two cases:

> $\tau \neq B$: In this case we obviously have $q' \models_F$ optResolve(B) since q' is an f_B-consistent successor of ρ.
>
> $\tau = B$: Let $m_B = f_B(\rho)$. If m_B is a silent move then nothing is to show since according to the definition of $\mathcal{C}$ all silent moves of B have the same effect on the MSR component of a state of $\mathcal{C}$. If m_B is a non-silent move we construct a strategy f'_B from f_B by postponing m_B to the next state when it is B's turn. Note that a similar move to m_B can be applied in that state. Then it is easy to show that for some $\rho' \in out(q', f'_B)$ such that $\rho' \models_F \Box\neg c_B$ there is some $\rho'' \in out(q_n, f_B)$ such that $\rho'' \models_F \Box\neg c_B$.

$\mathcal{C} \models_F$ optAbortRemain(B): This can be shown similar to the previous property.

$\mathcal{C} \models_F$ silOptAbort(B): This axiom corresponds to one part of Lemma 24 of [CMSS05]. Note that timer rules of A and B are not optimistic and not silent for B thus on every computation that is $(A : \mathrm{o}, B : \mathrm{o} \vee \mathrm{s})$-consistent no MSR moves that correspond to timer rules occur. One can prove that $\mathcal{C}$ fairly satisfies silOptAbort(B) with similar arguments as used in the proof of Lemma 24 of [CMSS05].

$\mathcal{C} \models_F$ optSil(B): Since by definition of $\mathcal{C}$ there are no optimistic moves of A that are silent this axiom is trivially fulfilled.

With Theorem 5.2.2 we can conclude that

$$\langle\langle B \rangle\rangle_{\substack{A:\mathrm{o} \\ B:\mathrm{o}\vee\mathrm{s}}} \Diamond \big(\mathrm{optAdvantage}(B) \wedge \neg\mathrm{initial}_A\big)$$

holds in s_I of $\mathcal{C}$. One can show that a strategy of B for $\langle\langle B \rangle\rangle_{\substack{A:\mathrm{o} \\ B:\mathrm{o}\vee\mathrm{s}}} \Diamond \big(\mathrm{optAdvantage}(B) \wedge \neg\mathrm{initial}_A\big)$ can be translated into a strategy for B in P at S_0 against an optimistic A to reach a non initial state in which B has an advantage against optimistic A in the MSR model. This finishes our proof of Theorem 5.3.3.

Chapter 6

Conclusion

The central topic of this thesis was to investigate how strategy properties of cryptographic protocols can be handled. For trace based security properties much research has been done in the past. Decidability results and algorithms were available. These results had led to industrial strength tools for analyzing cryptographic protocols. In this thesis we transfer some of the results available for trace based properties to strategy properties.

In Chapter 3 we showed how to use the constraint solving approach know from the analysis of reachability properties for strategy security properties. In our approach standard constraint solvers can be utilized and thus it should be possible to implement tools for the analysis of strategy properties of cryptographic protocols on top of existing tools based on constraint solvers.

In Chapter 4 we introduced AMC for cryptographic protocols. We have interpreted AMC-formulas over concurrent game structures (CGS) that model real concurrent behavior of protocol participants in the sense that each player in these CGS performs a move in each step of the system. We have shown undecidability results and have identified a decidable fragment of AMC (the $\mathcal{I}$-monotone fragment) that is sufficient to express all strategy security properties of cryptographic protocols we have found in the literature.

In Chapter 5 we introduced a small extension called $\text{ATL}_{MS}$of ATL. This extension allows us in a natural way to take different kinds of behavior of participants into account. We utilized $\text{ATL}_{MS}$to reformulate the fundamental impossibility result of Chadha et al. [CMSS05] in an axiomatic and model independent fashion.

There are many questions concerning strategy properties of cryptographic protocols that were not answered here. We want to state some of these questions.

In this thesis we formulated security properties of contract signing protocols in terms of strategies. Are there security properties of other kinds of protocols that can be formulated as strategy properties?

We have shown that under certain reasonable assumptions the $\mathcal{I}$-monotone fragment of AMC is decidable. What decidability results are there for the non $\mathcal{I}$-monotone fragment of AMC for cryptographic protocols?

We have shown that the constraint solving approach can be used to decide simple strategy properties of cryptographic protocols. Can one extend the constraint solving approach to decide properties formulated in the $\mathcal{I}$-monotone fragment of AMC for cryptographic protocols?

Danksagung

Ohne die Unterstützung, Hilfe und Zuspruch vieler Menschen hätte ich diese Arbeit niemals beenden können. In erster Linie bedanke ich mich bei meinem Betreuer Thomas Wilke für seine Hilfe, sein Vertrauen und seine Führung, er ist mir nicht nur „als begeisterter, sorgfältiger Forscher und liebenswerter Mensch ein wichtiges Vorbild“ [Fri05].

Ohne die Anleitung durch Ralf Küsters wäre ich nicht sehr weit gekommen, er hat mich durch subtilste Motivationsmethoden u.a. in Form von gesetzten Fristen, freundlichen E-Mail-Nachrichten und telefonischen Nachfragen immer wieder zum Arbeiten animiert. Vielen Dank dafür! Jetzt weiß ich: „Um zwölf Uhr wird gegessen.“

Für die anstrengende Arbeit als Gutachter dieser Arbeit und für die vielen hilfreichen Anmerkungen möchte ich Steve Kremer danken.

Die Arbeitsathmosphäre hier am Lehrstuhl von Thomas Wilke hätte besser nicht sein können, vielen Dank an all die Kollegen, die ich hier kennengelernt habe und zu dieser Atmosphäre beigetragen haben: Margrit Krause, Brigitte Scheidemann, Erich Valkema, Carsten Fritz, Imran Khan, Tomasz Truderung (ein wichtiger Co-Autor), Misha Aizatulin, Klaas Ole Kürtz und in der letzten Zeit Franziska Lorenz und Henning Schnoor.

Auch außerhalb der Universität gibt es Menschen, ohne deren Zuspruch ich diese Arbeit nicht beendet hätte: Meine Familie, insbesondere meine Mutter und mein Bruder, meinen Freund Axel, der mich häufig aufzubauen hatte und Chritof und Nikola, bei denen ich sehr erholsame Auszeiten genossen und dabei die vegetarische Küche in ihrer Vielfalt schätzen gelernt habe.

Zu guter Letzt danke ich Alice, Bob und Charlie für ihre Bereitschaft in die unterschiedlichsten Rollen und Charaktere zu schlüpfen, dafür habt ihr einen Oscar verdient.

Bibliography

[ABB$^+$05] A. Armando, D.A. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuéllar, P.H. Drielsma, P.-C. Héam, O. Kouchnarenko, J. Mantovani, S. Mödersheim, D. von Oheimb, M. Rusinowitch, J. Santiago, M. Turuani, L. Viganò, and L. Vigneron. The AVISPA Tool for the Automated Validation of Internet Security Protocols and Applications. In K. Etessami and S.K. Rajamani, editors, *Computer Aided Verification, 17th International Conference (CAV 2005)*, volume 3576 of *Lecture Notes in Computer Science*, pages 281–285. Springer-Verlag, 2005.

[AG97] M. Abadi and A.D. Gordon. A calculus for cryptographic protocols: The spi calculus. In *Fourth ACM Conference on Computer and Communications Security*, pages 36–47. ACM Press, 1997.

[AHK97] R. Alur, T.A. Henzinger, and O. Kupferman. Alternating-time temporal logic. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science (FOCS'97)*, pages 100–109. IEEE Computer Society Press, 1997.

[AHK02] R. Alur, T.A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *J. ACM*, 49(5):672–713, 2002.

[ASW98] 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. IEEE Computer Society, 1998.

[BAN90] M. Burrows, M. Abadi, and R. Needham. A Logic of Authentication. *ACM Transactions on Computer Systems*, 8(1):18–36, 1990.

[BDD$^+$05] M. Backes, A. Datta, A. Derek, J.C. Mitchell, and M. Turuani. Compositional analysis of contract signing protocols. In *CSFW '05: Proceedings of the 18th IEEE Computer Security Foundations Workshop (CSFW'05)*, pages 94–110, Washington, DC, USA, 2005. IEEE Computer Society.

[Bla06] B. Blanchet. A Computationally Sound Mechanized Prover for Security Protocols. In *IEEE Symposium on Security and Privacy (S&P 2006)*, pages 140–154. IEEE Computer Society, 2006.

[BMV03] D. Basin, S. Mödersheim, and L. Viganò. An On-The-Fly Model-Checker for Security Protocol Analysis. In E. Snekkenes and D. Gollmann, editors, *Proceedings of the 8th European Symposium on Research in Computer Security (ESORICS 2003)*, volume 2808 of *Lecture Notes in Computer Science*, pages 253–270. Springer, 2003.

[CDL$^+$99] I. Cervesato, N.A. Durgin, P.D. Lincoln, J.C. Mitchell, and A. Scedrov. A Meta-notation for Protocol Analysis. In *Proceedings of the 12th IEEE Computer Security Foundations Workshop (CSFW-12)*, pages 55–69. IEEE Computer Society, 1999.

[CGLV03] D. Calvanese, G .De Giacomo, M. Lenzerini, and M.Y. Vardi. View-based query containment. In *PODS 2003*, pages 56–67. ACM Press, 2003.

[CKRT03] Y. Chevalier, R. Küsters, M. Rusinowitch, and M. Turuani. An NP Decision Procedure for Protocol Insecurity with XOR. In *Proceedings of the Eighteenth Annual IEEE Symposium on Logic in Computer Science (LICS 2003)*, pages 261–270. IEEE, Computer Society Press, 2003.

[CKS01] R. Chadha, M.I. Kanovich, and A. Scedrov. Inductive methods and contract-signing protocols. In P. Samarati, editor, *8-*

th ACM Conference on Computer and Communications Security (CCS 2001), pages 176–185. ACM Press, 2001.

[CKS04] R. Chadha, S. Kremer, and A. Scedrov. Formal analysis of multi-party contract signing. In R. Focardi, editor, *17th IEEE Computer Security Foundations Workshop (CSFW-17)*, pages 266–279. IEEE Computer Society Press, 2004.

[CMSS05] R. Chadha, J.C. Mitchell, A. Scedrov, and V. Shmatikov. Contract Signing, Optimism, and Advantage. *Journal of Logic and Algebraic Programming*, 64(2):189–218, 2005.

[CV01] Y. Chevalier and L. Vigneron. A Tool for Lazy Verification of Security Protocols. In *Proceedings of the 16th IEEE Conference on Automated Software Engineering (ASE 2001)*, pages 373–376. IEEE CS Press, 2001.

[CV02] Y. Chevalier and L. Vigneron. Automated Unbounded Verification of Security Protocols. In *Proceedings of the 14th International Conference on Computer Aided Verification (CAV 2002)*, volume 2404 of *Lecture Notes in Computer Science*, pages 324–337. Springer, 2002.

[DLMS04] N.A. Durgin, P. Lincoln, J.C. Mitchell, and A. Scedrov. Multiset rewriting and the complexity of bounded security protocols. *Journal of Computer Security*, 12(2):247–311, 2004.

[DM04] P. H. Drielsma and S. Mödersheim. The ASW Protocol Revisited: A Unified View. In *Workshop on Automated Reasoning for Security Protocol Analysis (ARSPA)*, 2004.

[DMP03] N.A. Durgin, J.C. Mitchell, and D. Pavlovic. A compositional logic for proving security properties of protocols. *Journal of Computer Security*, 11(4):677–721, 2003.

[DY83] D. Dolev and A.C. Yao. On the Security of Public-Key Protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.

[EG83] S. Even and O. Goldreich. On the Security of Multi-Party Ping-Pong Protocols. Technical Report 285, Technion – Israel Institut of Technology, 1983.

[EJ91] E. Allen Emerson and Charanjit S. Jutla. Tree automata, mu-calculus and determinacy (extended abstract). In *32nd Annual Symposium on Foundations of Computer Science (FOCS '91)*, pages 368–377. IEEE Computer Society Press, 1991.

[EY80] S. Even and Y. Yacobi. Relations among public key signature systems. Technical Report 175, Technion, Haifa, Israel, March 1980.

[Fri05] C. Fritz. *Simulation-Based Simplification of omega-Automata.* PhD thesis, Christian-Albrechts-Universiät zu Kiel, 2005.

[GJM99] J.A. Garay, M. Jakobsson, and P. MacKenzie. Abuse-free optimistic contract signing. In *Advances in Cryptology – CRYPTO'99, 19th Annual International Cryptology Conference*, volume 1666 of *Lecture Notes in Computer Science*, pages 449–466. Springer-Verlag, 1999.

[GTW02] E. Grädel, W. Thomas, and Th. Wilke, editors. *Automata, Logics, and Infinite Games: A Guide to Current Research*, volume 2500 of *Lecture Notes in Computer Science*. Springer, 2002.

[KK05] D. Kähler and R. Küsters. Constraint Solving for Contract-Signing Protocols. In M. Abadi and L. de Alfaro, editors, *Proceedings of the 16th International Conference on Concurrency Theory (CONCUR 2005)*, volume 3653 of *Lecture Notes in Computer Science*, pages 233–247. Springer, 2005.

[KKT07a] D. Kähler, R. Küsters, and T. Truderung. Infinite State AMC-Model Checking for Cryptographic Protocols. Technical report, CAU Kiel, Germany, 2007. Available from http://people.inf.ethz.ch/kuestral/publications_html/KKT-TR-AMC-2007.pdf.

[KKT07b] D. Kähler, R. Küsters, and T. Truderung. Infinite state amc-model checking for cryptographic protocols. In *22nd IEEE Symposium on Logic in Computer Science (LICS 2007)*, pages 181–192. IEEE Computer Society, 2007.

[KKW04] D. Kähler, R. Küsters, and Th. Wilke. Deciding Properties of Contract-Signing Protocols. Technical Report 0409, Institut für Informatik und Praktische Mathematik, CAU Kiel, Germany, 2004.

[KKW05] D. Kähler, R. Küsters, and Th. Wilke. Deciding Properties of Contract-Signing Protocols. In Volker Diekert and Bruno Durand, editors, *Proceedings of the 22nd Symposium on Theoretical Aspects of Computer Science (STACS 2005)*, volume 3404 of *Lecture Notes in Computer Science*, pages 158–169. Springer-Verlag, 2005.

[KKW06] D. Kähler, R. Küsters, and Th. Wilke. A Dolev-Yao-based Definition of Abuse-free Protocols. In M. Bugliesi, B. Preneel, V. Sassone, and I. Wegener, editors, *Proceedings of the 33rd International Colloqium on Automata, Languages, and Programming (ICALP 2006)*, volume 4052 of *Lecture Notes in Computer Science*, pages 95–106. Springer, 2006.

[KR01] Steve Kremer and Jean-François Raskin. A game-based verification of non-repudiation and fair exchange protocols. In *12th International Conference on Concurrency Theory (CONCUR 2001)*, volume 2154 of *Lecture Notes in Computer Science*, pages 551–565. Springer-Verlag, 2001.

[KR02] S. Kremer and J.-F. Raskin. Game analysis of abuse-free contract signing. In *Computer Security Foundations Workshop 2002 (CSFW 2002)*, pages 206–220. IEEE Computer Society, 2002.

[Low95] G. Lowe. An attack on the Needham-Schroeder public-key authentication protocol. *Information Processing Letters*, 56:131–133, 1995.

[Mar75] D.A. Martin. Borel Determinacy. *Annals of Mathematics*, 102:363–371, 1975.

[Mos91] A.W. Mostowski. Games with forbidden positions. Technical Report 78, Uniwersytet Gdański, Instytut Matematyki, 1991.

[MPW92] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, i. *Inf. Comput.*, 100(1):1–40, 1992.

[MS01] J. K. Millen and V. Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *Proceedings of the 8th ACM conference on Computer and Communications Security*, pages 166–175. ACM Press, 2001.

[MvV97] A.J. Menezes, P. C. von Oorschot, and S.A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, New York, 1997.

[NS78] R. Needham and M. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12):993–999, 1978.

[PG99] H. Pagnia and F.C. Gärtner. On the impossibility of fair exchange without a trusted third party. Technical Report TUD-BS-1999-02, Darmstadt University of Technology, Department of Computer Science, Darmstadt, Germany, March 1999.

[RT03] M. Rusinowitch and M. Turuani. Protocol insecurity with a finite number of sessions, composed keys is NP-complete. *Theoretical Computer Science*, 299(1–3):451–475, 2003.

[SF06] S. Schewe and B. Finkbeiner. Satisfiability and Finite Model Property for the Alternating-Time mu-Calculus. In *CSL 2006*, volume 4207 of *Lecture Notes in Computer Science*, pages 591–605. Springer, 2006.

[SM02] V. Shmatikov and J.C. Mitchell. Finite-state analysis of two contract signing protocols. *Theoretical Computer Science (TCS), special issue on Theoretical Foundations of Security Analysis and Design*, 283(2):419–450, 2002.

[THG99] F.J. Thayer, J.C. Herzog, and J.D. Guttman. Strand spaces: Proving security protocols correct. *Journal of Computer Security*, 7(1), 1999.

[Wil01] Th. Wilke. Alternating tree automata, parity games, and modal μ-calculus. *Bull. Soc. Math. Belg.*, 8(2), May 2001.

[Zie98] Wiesław Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theoretical Computer Science*, 200(1-2):135–183, 1998.