Logical Analysis and Verification of Cryptographic Protocols

Thesis presented and defended by

MOUNIRA KOURJIEH

at Toulouse on the 9^{th} of December 2009 to obtain the degree of

Docteur de l'Université de Toulouse
(Speciality: Computer Science)

Supervisors:
Yannick Chevalier Philippe Balbiani

Reviewers:
Véronique Cortier Christopher Lynch

Examiners:
Jean-Paul Bodeveix Sais Lakhdar Yassine Lakhnech

Université de Toulouse -
École Doctorale Mathématiques, Informatique et
Télécommunications de Toulouse -
Institut de Recherche en Informatique de Toulouse

Acknowledgement

First of all, I owe a special and warm thank-you to Yannick Chevalier for giving me the opportunity to do this PhD. His high competence combined with his human qualities provided me a very pleasant and valuable supervision. I thank him for his great and constant availability, for his clear and precise answers to my questions, for his numerous advices and comments, and for his constant human support during this adventure. I cordially thank Philippe Balbiani for his encouragements, his advices, his constant support during my thesis, and for helping and encouraging me to pursue a PhD thesis.

Then I want to thank Véronique Cortier and Christopher Lynch for accepting to review my thesis, for their detailed reviews and for their relevant remarks. I also thank Jean-Paul Bodeveix, Sais Lakhdar and Yassine Lakhnech for the honor of having them in my jury and for their questions and comments during my defense.

I thank Mark Ryan who warmly welcomed me in his team at the Computer Science School, Birmingham, United Kingdom, and for the opportunity he gave me to work together. I also thank Ben Smyth and Steve Kremer.

I thank Michael Rusinowitch who warmly welcomed me several times in his team at LORIA, Nancy, and for all the discussions we had.

Finally, I thank all the members of LILaC team, my friends, François, Nadine, Fahima, Marwa, Bilal, Guillaume, Tiago, Nicolas, Enrica, Myrto, Manh-hung, Zein, Karine, Najah, *etc.*, and a special and big thank-you to my family.

Abstract

This thesis is developed in the framework of symbolic analysis of cryptographic protocols. The contributions of this thesis can be split into three mains parts:

1. We analyse the three classes of cryptographic protocols using respectively collision vulnerable hash functions, key substitution vulnerable digital signature schemes, and cryptographic primitives represented by convergent equational theories having the *finite variant property*.
   - We conjecture that the verification problem of the first class of protocols can be reduced to the verification problem of the class of cryptographic protocols using an associative symbol of concatenation, and we show the decidability of the verification problem for the last class.
   - We show the decidability of the verification problem for the second two classes of protocols.
2. We show the decidability of the ground entailment problem for a new fragment of first order logic, and we show the application of this result on the verification problem of cryptographic protocols.
3. We analyse the electronic-voting protocols, and we give a formal definition for the voter-verifiability property. We also show that some well-known electronic voting protocols satisfy this property.

Keywords: Security protocols, electronic-voting protocols, decision procedures, algebraic primitives, constraint systems, first order clauses, resolution, saturation, applied π calculus.

This thesis is presented and defended at Toulouse on the 9^{th} of December 2009, was performed under the supervision of Yannick Chevalier and Philippe Balbiani. The author obtained the degree of Docteur en Informatique de l'Université de Toulouse.

Résumé

Cette thèse est développée dans le cadre de l'analyse symbolique des protocoles cryptographiques. Les contributions de cette thèse peuvent être divisées en trois parties principales :

1. Nous analysons les trois classes des protocoles cryptographiques qui utilisent respectivement des fonctions de hachage vulnérables à la collision, des schémas de signature vulnérables à la substitution des clés, et des primitives cryptographiques représentées par des théories équationnelles convergentes ayant la *propriété de variante finie*.
   - Nous conjecturons que le problème de vérification de la première classe des protocoles peut être réduit au problème de vérification de la classe des protocoles cryptographiques qui utilisent un symbole associatif de la concaténation, et nous montrons la décidabilité du problème de vérification de la dernière classe.
   - Nous montrons la décidabilité du problème de vérification des deux dernières classes des protocoles.
2. Nous montrons la décidabilité du problème de déduction clos pour un nouveau fragment de la logique du premier ordre, et nous montrons l'application de ce résultat sur le problème de vérification des protocoles cryptographiques.
3. Nous analysons les protocoles de vote électronique, et nous donnons une définition formelle pour la propriété de vérifiabilité de vote. Nous montrons également que certains protocoles de vote électronique satisfont cette propriété.

Mots clés : protocoles cryptographiques, protocoles de vote électronique, procédures de décision, primitives algébriques, systèmes de constraintes, clauses de premier ordre, résolution, saturation, π calcul appliqué.

Cette thèse a été présentée et soutenue à Toulouse le 9 décembre 2009, a été réalisée sous la direction de Yannick Chevalier et Philippe Balbiani. L'auteur a obtenu le grade de Docteur en Informatique de l'Université de Toulouse.

Contents

Chapter 1

Introduction

In our society, the use of electronic applications such as e-communication, e-voting, e-banking, e-commerce, *etc* is increasing. Among several important requirements, security figures as one crucial aspect. To guarantee security, such applications use cryptographic protocols. These are small concurrent programs executed by several distant agents through a network. Messages, or part of the messages, are produced using cryptographic functions (encryption, signature, hashing, *etc*). Cryptography has been used for thousands of years to safeguard military and diplomatic communications. For example, the famous Roman emperor Julius Caesar used encryption to securely communicate with his troops.

Unfortunately, the usage of cryptographic primitives in a protocol is not sufficient to ensure its security and several attacks were found on established protocols [78]. The most stating example is the bug (a so-called *man-in-the-middle* attack) of the Needham-Schroeder protocol found by G. Lowe [144] 17 years after the publication of the protocol. This situation leads to, and shows the importance of, the development of tools and decision procedures for the formal verification of security protocols.

1.1 Cryptographic protocols

1.1.1 Communication via exchanges of messages

A *communication protocol*, or simply *protocol* can be described by exchanges of messages between many participants. These exchanges are usually described by a *scenario*, sequence of *rules* each one specifying the *sender*, the *receiver*, and the *exchanged message*. The scenario describes a *normal (also said correct) run* of the protocol, that is how the execution of the protocol should proceed in the

Figure 1.1 Example of protocol

$$\begin{cases} A \Rightarrow B: & \text{“Hello, I am A.”} \\ B \Rightarrow A: & \text{“Hi, I am B. Nice to meet you.”} \end{cases}$$

absence of the intruder. A simple example of protocol is given in Figure 1.1.

This protocol describes the first meeting between two people. In this protocol, we have two roles “A” (abbreviation of “Alice”) and “B” (abbreviation of “Bob”).

In the scenario, the *roles* represent *abstract participants*. In the protocol described in Figure 1.1, we have two roles “A” (the *initiator role*) and “B” (the *responder role*).

We call an *execution of the protocol* any coherent, with respect to the description of that protocol, set of exchanges of messages between its participants. In the execution of a protocol, the roles are instantiated by *concrete participants*, also called *agents*. When an agent a instantiates a role $\mathfrak{R}$, we say that *“the agent a plays the role $\mathfrak{R}$”*. We remark that a role can be played by many agents and any agent can play many roles or the same role many times. We call *session of a protocol* a set of exchanges of messages between the participants of the protocol which is (1) coherent with respect to the description of that protocol, (2) can be repeated, and (3) where each role is instantiated only once.

We remark that it is important to take into consideration many instances of the same protocol, and thus we talk about execution of one session of the protocol, also called *run of the protocol*, and execution of many sessions of the protocol.

Example 1 *The following set of exchanges of messages*

$$\begin{cases} (1).1 \;\; Alice(A) \Rightarrow Bob(B): & \textit{“Hello, I am Alice.”} \\ (1).2 \;\; Bob(B) \Rightarrow Alice(A): & \textit{“Hi, I am Bob. Nice to meet you.”} \end{cases}$$

represents an execution of one session of the protocol described in Example 1.1.

In this execution, we have two concrete participants, “Alice” playing the role “A” and “Bob” playing the role “B”.

Example 2 *The following set of exchanges of messages*

$$\begin{cases} (1).1 \;\; Alice(A) \Rightarrow Bob(B): & \textit{“Hello, I am Alice.”} \\ (2).1 \;\; Bob(A) \Rightarrow Marlie(B): & \textit{“Hello, I am Bob.”} \\ (1).2 \;\; Bob(B) \Rightarrow Alice(A): & \textit{“Hi, I am Bob. Nice to meet you.”} \\ (2).2 \;\; Marlie(B) \Rightarrow Bob(A): & \textit{“Hi, I am Marlie. Nice to meet you.”} \end{cases}$$

represents an execution of two sessions of the protocol described in Example 1.1. In this execution, we have three concrete participants, “Alice”, “Bob” and “Marlie”: “Alice”

(playing the role "A") instantiates a session (1) *with "Bob" (playing the role "B"), and then, "Bob" (playing the role "A") instances another session* (2) *with "Marlie" (playing the role "B").*

1.1.2 Participants

A cryptographic protocol is a communication protocol having several security goals such as *exchange secret information, authenticate the communication partner, anonymity, etc.*. We have two categories of the concrete participants of a protocol: "the honest participants" and "the dishonest participants".

Honest participant

A *honest participant* is a concrete participant of the protocol. It sends and receives messages in a pattern defined by the role it plays.

Dishonest participant

Cryptographic protocols are executed in a malicious (or unsafe) environment. This environment is represented by a special participant of the protocol: *the dishonest participant*, also called *the intruder, the adversary,* or *the attacker*. The intruder has a complete control over the communication medium (network), he listens to the communication and can obtain any message passing through the network, he can intercept, block, and/or redirect all messages sent by honest agents. He can masquerade his identity and take part in the protocol under the identity of an honest participant. He also can deduce new messages using some specific rules. This intruder is called *Dolev-Yao intruder* as it was first introduced by D. Dolev and A. Yao in [107].
The intruder is called:

- *eavesdropper* when it is only able to listen to the network.
- *passive* when it is only able to listen to the network and to use specific rules to compute new messages.
- *active* when it can furthermore send messages to honest participants.

If we have more than one intruder, we assume that all the intruders cooperate together.

1.1.3 Security properties

Cryptographic protocols are communication protocols aiming at ensuring some security objectives called *security properties*. We give below some of these properties.

Secrecy
The *secrecy* property is also called *confidentiality* or *privacy*. We distinguish three main levels of secrecy:

- Simple secrecy means that the "secret data" should be known only by some agents, and in particular should not be known by the intruder; we say that the secrecy of the data "sec" is preserved if the intruder is not able to deduce it.
- Strong secrecy means that the intruder should not be able to know anything about the secret data.
- Forward secrecy means that some secret data should be kept secret to the intruder even after revealing some other secret data.

In this document, and for sake of simplicity, we denote by "secrecy" the "simple secrecy".

Authentication
The *authentication* property is closely related to the *identification*. This property is applied to both entities and information. Two parties communicating together should identify each other. In a similar way, information delivered over a channel should be authenticated as to origin, data content, etc. The authentication property is divided into two classes: *entity authentication* and *data origin authentication*. We remark that *data origin authentication* implies *data integrity property* which addresses the unauthorised modification of data.

Non-repudiation
prevents an entity from denying previous commitments or actions.

We remark that other security properties such as *anonymity, certification, revocation, etc.* can be derived from the properties *secrecy, authentication, data integrity* and *non-repudiation*. We also remark that specific kinds of cryptographic protocols such as "electronic-voting protocols" require other security properties such as "coercion-resistance, receipt-freeness, voter verifiability, *etc*". These properties are discussed in more detail in Chapter 7. A complete list of security properties can be found in [179].

1.2 Cryptographic primitives

Cryptographic protocols are communication protocols in which cryptographic primitives are used to construct messages. We give below some of these cryptographic primitives, and for each of them some of its algebraic properties.

1.2.1 Concatenation

The *concatenation*, denoted by "$\cdot$", is commonly used in cryptographic protocols to construct messages. The concatenation algorithm "$\cdot$" is represented by a function that takes a couple of messages as input and outputs their concatenation, for example the concatenation of "a" and "b" is "a$\cdot$ b".

We give below the *algebraic properties* of the concatenation:

$$\begin{array}{l} Associativity: \ (x \cdot y) \cdot z = x \cdot (y \cdot z) \\ Unary: \ x \cdot \epsilon = \epsilon \cdot x = x \end{array}$$

1.2.2 Exclusive or

The *exclusive or*, denoted by "$\oplus$" or "XOR", is commonly used in cryptographic protocols to construct messages, it has the following algebraic properties:

$$\begin{array}{l} Associativity: \ (x \oplus y) \oplus z = x \oplus (y \oplus z) \\ Commutativity: \ x \oplus y = y \oplus x \\ Unary: \ x \oplus \epsilon = x \\ Nilpotence: \ x \oplus x = \epsilon \end{array}$$

1.2.3 Pairing

The *pairing*, denoted by "$\langle -, - \rangle$", is commonly used in cryptographic protocols to construct messages, it has the following algebraic properties:

$$\begin{array}{l} 1^{st}\ projection: \ \pi_1\langle x, y\rangle = x \\ 2^{nd}\ projection: \pi_2\langle x, y\rangle = y \end{array}$$

where π_1 and π_2 denote the projection functions. The main difference between the *pairing* and the *concatenation* is that the pairing is not associative, *i.e.* $\langle x, \langle y, z\rangle\rangle \neq \langle\langle x, y\rangle, z\rangle$.

1.2.4 Encryption schemes

We have two types of *encryption schemes*: the *public (or asymmetric) encryption schemes* and the *symmetric encryption schemes*.

Public encryption schemes

The *public encryption schemes*, also called *asymmetric encryption schemes*, have been initially introduced by W. Diffie and M. Hellman [101] and R. Merkle [154].

A "public encryption scheme" is defined by three algorithms: the *encryption algorithm* "enc^p", the *decryption algorithm* "dec^p", and the *key generation algorithm*

"G". The key generation algorithm takes as input an agent name A and a random generated number and returns a pair of public and secret keys, respectively denoted by $Pk(A)$ and $Sk(A)$, corresponding to that agent. This random number allows any agent to have a different pair of keys for each session while using the same key generation algorithm. The encryption algorithm takes as input a clear message, called *plaintext*, and an agent's public key, and outputs the encryption of that plaintext, called *ciphertext*, with respect to the given public key. The decryption algorithm takes an input a ciphertext and an agent's private key, and outputs the decryption of that ciphertext with respect to the given private key provided that the ciphertext has been obtained using the public key correspondant to the given private key.

Example 3 *One of the most common public encryption schemes is the encryption scheme due to R. Rivest, A. Shamir and L. Adleman [174] and denoted by "RSA public encryption scheme".*

A *perfect public encryption scheme* is represented by the following equation:

$$dec^p(enc^p(x, Pk(y)), Sk(y)) = x$$

Symmetric encryption schemes

A *symmetric encryption scheme* is similar to the *asymmetric encryption scheme* with the condition that the same key is used to encrypt and decrypt messages. This means that in these schemes, the key generation algorithm outputs one key instead of a pair of keys. Furthermore, in these schemes, the encryption algorithm is denoted by enc^s, and the decryption algorithm by dec^s.

A *perfect symmetric encryption scheme* is represented by the following equation:

$$dec^s(enc^s(x, y), y) = x$$

Algebraic properties

An (asymmetric or symmetric) encryption scheme may use some operators such as the concatenation "$\cdot$", or XOR "$\oplus$", or multiplication "$*$", or pairing "$\langle -, - \rangle$". Such schemes may have some of the following algebraic properties:

Commuting. The commuting property is represented by the following equation $enc(enc(x, y), z) = enc(enc(x, z), y)$. One of the most important commuting encryption schemes is *RSA public encryption scheme* with common modulus [69].

Homomorphism. The homomorphism property is represented by the following equation $enc(x, y) * enc(z, y) = enc(x * z, y)$, and it means that one can get

from the encryption of the messages m_1 and m_2 the encryption of the new message $m_1 * m_2$ without knowing the encryption key. The *RSA public encryption scheme* possesses this property [112].

Prefix property. The prefix property means that one can get from an encrypted message the encryption of any of its prefixes, and it is formally represented as follows: from a message $enc(< x, y >, z)$ one can get the message $enc(x, z)$.

1.2.5 Signature

Digital signature schemes first appeared in *W. Diffie and M.E. Hellman's* seminal paper [101]. Their most important goal is to demonstrate the authenticity of a digital message or document: a valid digital signature gives a recipient reason to believe that the message was created by a known sender, and that it was not altered in transit.

Such schemes are described by three algorithms: the *signature generation* "sig", the *verification* "ver", and the *key generation* "G" algorithms. The key generation algorithm takes as input an agent name A and a random generated number and returns a pair of public and secret keys, respectively denoted by $Pk(A)$ and $Sk(A)$, corresponding to that agent. This random number allows any agent to have a different pair of keys for each session, and that using the same key generation functions. The signature generation algorithm inputs a message and an agent's private key, and outputs the signature of the given agent for the given message. There are two types of verification algorithms:

- The verification algorithm of the first type takes as input a message, a signature, and an agent public key, and outputs "succeeds" if the given signature corresponds to the signature done by the given agent for the given message.

- A verification algorithm of the second type takes as input a signature and an agent's public key, recovers the message that has been signed from the signature and returns it provided that the given signature is done by the given agent.

When the verification algorithm is of the first type, we call the signature scheme "a signature scheme with appendix" and when the verification algorithm is of the second type, we call the signature scheme "a signature scheme with message recovery".

A perfect signature scheme with appendix is represented by the following equation

$$ver(x, sig(x, Sk(y)), Pk(y)) = 1$$

And, a perfect signature scheme with message recovery is represented by the following equation

$$ver(sig(x, Sk(y)), Pk(y)) = x$$

1.2.6 Blind signature

Blind signature schemes, initially introduced by D. Chaum [61], is a form of digital signature schemes in which the content of a message is blinded (disguised) before it is signed. The resulting blind signature can be publicly verified against the original, unblinded message in the manner of a regular digital signature. Blind signatures are typically employed in privacy-related protocols where the signer and message author are different parties; examples include electronic voting systems [115] and digital cash schemes.

The blind signature schemes are described by the following algorithms:

- the *signature generation* "sig", the *verification* "ver", and the *key generation* "G" algorithms which are defined as for the traditional digital signature schemes given in Section 1.2.5.
- The *blind* "Bl"and *unblind* "Ubl" algorithms defined as follows:

  Blind algorithm takes as input a message "m" and a random selected value "r", and returns the blinded value of "m" with respect to "r".

  Unblind algorithm takes as input a blinded message "b" and a random selected value "r", and outputs the original (unblinded) message provided that "b" has been constructed using "r" as random selected value.

The blind signature schemes have the following algebraic properties:

$$ver(sig(x, Sk(y)), Pk(y)) = x, \quad (or, ver(x, sig(x, Sk(y)), Pk(y)) = 1)$$
$$Ubl(Bl(x, y), y) = x$$
$$Ubl(sig(Bl(x, y), Sk(z)), y) = sig(x, Sk(z))$$

1.2.7 Hash function

A cryptographic hash function is a deterministic procedure that takes an arbitrary block of data and returns a fixed-size bit string. Cryptographic hash functions have many information security applications, notably in digital signatures, message authentication codes (MACs), and other forms of authentication.

Hash functions may have some of the following properties:

Preimage property
given a hash value n, one can compute a data m such that the hash value of m is equal to n.

Second preimage property
given a data m, one can compute another data $m' \neq m$ such that m and m' have the same hash value.

Collision property
one can compute two distinct data m, m' such that they have the same hash value.

These properties are described in more detail in Chapter 3. A *perfect hash function* does not have any of these properties.

1.3 Example: Needham-Schroeder protocol

Description of the protocol. The *Needham-Schroeder symmetric key protocol* is one of the most known cryptographic protocols. It has been designed By R. M. Needham and M. D. Schroeder in 1978 [163], and it forms the basis for the Kerberos protocol.

It is described as follows:

$$\mathfrak{P}_{NS} : \begin{cases} 1.\ A \Rightarrow S : & \langle A, \langle B, N_A \rangle \rangle \\ 2.\ S \Rightarrow A : & enc^s(\langle N_A, B, K_{AB}, enc^s(\langle K_{AB}, A \rangle, K_{BS}) \rangle, K_{AS}) \\ 3.\ A \Rightarrow B : & enc^s(\langle K_{AB}, A \rangle, K_{BS}) \\ 4.\ B \Rightarrow A : & enc^s(N_B, K_{AB}) \\ 5.\ A \Rightarrow B : & enc^s(N_B - 1, K_{AB}) \end{cases}$$

N_A (respectively N_B) represents the nonce freshly created by A (respectively B), K_{AS} (respectively K_{BS}) represents the secret key shared between A (respectively B) and the *trusted server*, and K_{AB} the session key shared between A and B.

This protocol aims at establishing a session key between the participants, typically to protect further communication.

Analysis of the protocol. When this protocol has been designed, R. M. Needham and M. D. Schroeder assumed that the private keys are never compromised [163]. Under this assumption, the protocol is secure [99]. D. Denning and G. Sacco [99] showed the insecurity of the protocol when private keys are compromised. Actually, assume that (i) an agent a has initialised a session with an agent b, (ii) the intruder $\mathcal{I}$ has obtained a copy of the key K_{ab}, and (iii) $\mathcal{I}$ has

intercepted all messages between a and b. $\mathcal{I}$ can then later trick b into using the key K_{ab} as follows: first $\mathcal{I}$ replays the message $enc^s(\langle K_{ab}, a\rangle, K_{bs})$ to b

$$\mathcal{I} \Rightarrow b : \quad enc^s(\langle K_{ab}, a\rangle, K_{bs})$$

Assuming that a has initiated a new conversation, b replies to a

$$b \Rightarrow \mathcal{I}(a) : \quad enc^s(n'_b, K_{ab})$$

$\mathcal{I}$ intercepts the message, deciphers it, and impersonates $a's$ response:

$$\mathcal{I}(a) \Rightarrow b : \quad enc^s(n'_b - 1, K_{ab})$$

Thereafter, $\mathcal{I}$ can send false messages to b that appear to be from a.

This attack, called a *replay attack* is the most famous attack on the *Needham-Schroeder symmetric key protocol*.

1.4 Analysis of cryptographic protocols

1.4.1 Overview

Cryptographic protocols are programs designed to ensure secure electronic communications between participants using an insecure network. They use cryptographic primitives such as encryption schemes, signature schemes, hash functions, and others to construct exchanged messages. These cryptographic primitives are based on mathematical notions such as modular exponentiation and elliptic curves and on algorithmically hard problems such as factorisation into prime numbers, extracting the modular logarithm, and others.

Unfortunately, the existence of cryptographic primitives is not sufficient to ensure security and several attacks were found on established protocols [78, 2]. The most relevant example is the bug of the Needham-Schroeder public key protocol [163] found by Lowe [144] using a model-checking tool. It took 17 years since the protocol was published to find the attack, a *man-in-the-middle* one. This situation shows that the design of a cryptographic protocol is tricky, and that it is easy to have it wrong. Thus, one needs formal verification. In the literature, we find two distinct worlds for the verification of cryptographic protocols: the *computational world* and the *symbolic world*. Let us now review briefly these two approaches:

The "computational" world

In the *computational models*, also called *probabilistic, or cryptographic, or concrete models*, messages are bit strings, and the intruder is an arbitrary probabilistic polynomial-time Turing machine. These models are closer to the reality than

the symbolic ones, and hence results obtained in these models yield stronger security guarantees, but the validation proofs are essentially manual [38, 142, 49]. Recently, I. Tsahhirov and P. Laud [194] have developed an automatic tool to verify cryptographic protocols in the computational models, and B. Blanchet [48] has developed another automatic tool "CryptoVerif" which is more developed than the tool in [194]. Many results have been obtained in the computational models [13, 33, 26, 56, 203, 155]. For instance, in [26], M. Backes and B. Pfitzmann proved that the Needham-Schroeder-Lowe protocol is secure under real, and active cryptographic attacks including concurrent protocol runs. Still, another computationally sound proof of security for the Needham-Schroeder-Lowe protocol has been given in [203], where B. Warinschi has proved that the protocol is secure if it is implemented with an encryption scheme that satisfies indistinguishability under chosen-ciphertext attack. In [155], D. Micciancio and B. Warinschi showed that, in the case of assymetric encryption, the perfect encryption hypothesis is a sound abstraction for encryption schemes satisfying the IND-CCA2 property. In [13], M. Abadi and Ph. Rogaway showed that if two messages are formally indistinguishable then they are computationally indistinguishable, and that in the case where the symmetric encryption is the unique encryption scheme in addition of some technical restrictions. This result has been extended by M. Baudet, V. Cortier and S. Kremer [33] to take into consideration cryptographic primitives represented by a wide class of equational theories including exclusive or, and by E. Bresson, Y. Lakhnech, L. Mazaré and B. Warinschi [56] who take into consideration the modular exponentiation.

The "symbolic" world

The first symbolic models for the verification of cryptographic protocols are attributed to R. Needham and M. Schroeder [163], and to D. Dolev and A. Yao [107]. These models represent an abstraction of the real world, and hence allow simpler reasoning and an automatic analysis of security protocols. In the symbolic world, the cryptographic primitives are represented by function symbols, messages by elements in a signature (a set of function symbols representing the cryptographic primitives), and the intruder by a set of deduction rules. Such approaches adopt the so-called *Dolev-Yao intruder model* [107] represented as follows:

> The intruder has a complete control over the communication medium (network), he listens to the communication and can obtain any message passing through the network, he can intercept, block, and/or redirect all messages sent by honest agents. He also can masquerade his identity and take part in the protocol under the identity of an honest agent. His control of the network is modelled by assuming that all messages sent by honest agents are sent directly to

Figure 1.2 Security problem of cryptographic protocols

Input: a protocol $\mathfrak{P}$, and a security property A.
Output: SECURE if and only if the $\mathfrak{P}$ satisfies A.

the intruder and that all messages received by the honest agents are always sent by the intruder. Besides the control of the network, the intruder has some specific rules to deduce new messages.

In this document, we follow the symbolic approach in order to analyse cryptographic protocols.

1.4.2 Symbolic analysis of cryptographic protocols

Cryptographic protocols are difficult to verify for several reasons: the unbounded complexity of the exchanged messages, the unbounded number of the new generated data, the unbounded number of participants, and the unbounded number of sessions.

The "security problem" of cryptographic protocols which is stated in Figure 1.2 is undecidable in general [111]. In [110], the authors showed that the problem remains undecidable even if data constructors, message depth, message width, number of distinct roles, role length, and depth of encryption are bounded by constants. Another restrictions consist in bounding the number of fresh values (nonces) used during the attack. However several undecidable problems like PCP may be encoded in that case [89]. In order to decide the security problem of cryptographic protocols, we need to add more restrictions. We describe below a few methods used to symbolically analyse cryptographic protocols, then we give some obtained (un)decidability results.

Search for logical attacks. A first attempt was to search for logical attacks in the case of bounded number of sessions. In [192], the authors showed that, in this method, it is sufficient to consider a unique intruder. The first tools were based on *model-checking*. Such tools represent the protocols as finite-state machines and security properties as formula in temporal logic. Examples of such tools are FDR [108], Murϕ [160], or Brutus [79]. They discovered several interesting attacks assuming a bounded number of sessions and messages of bounded size. The most famous attack discovered using these tools is the attack on the Needham-Schroeder public key protocol [145].

A. Huima [122] was the first to suggest that bounding the size of messages is not necessary, and to analyse cryptographic protocols in that case, proposed the use of symbolic constraints. The use of symbolic constraints to analyse

cryptographic protocols has been studied later by [15, 156, 53], and then by [68, 95, 73, 70, 69, 54] who considered cryptographic protocols with algebraic primitives such as modular exponentiation, exclusive or, etc. Constraint solving has become the standard model to analyse cryptographic protocols in the case of bounded number of sessions.

In this approach, other works have been done in order to capture other security properties such as the resistance to the dictionary attacks [96], equivalence-based properties [109], and the properties related to contract-signing protocols [135].

Search for proofs. The drawback of this approach is that, assuming a bounded number of sessions, it does not prove the correctness of the protocol. We show here another approach, the "search for proofs", which does not assume a bounded number of sessions. One of its inconvenient is that it may introduce false attacks, that is it may reject protocols that do not have logical attacks. We give below some of the methods elaborated in this approach:

- Methods that use theorem provers such as the inductive approach of L. Paulson [168] which uses Isabelle to prove security properties, and the approach of Bolignano [52].
- Methods based on process algebra such as pi-calculus, spi-calculus and applied pi-calculus [12, 11].
- Methods based on logics such as BAN logic [57] which is due to M. Burrows and M. Abadi and R. M. Needham.
- NRL protocol analyser [151] which is known to be the first tool that does not impose any restriction.
- Methods that use tree automata [161, 118, 85].
- Methods based on Horn clauses [46, 180, 84, 205].

Several automatic tools have been developed to symbolically verify cryptographic protocols such as "Proverif" [46, 50], "AVISS" [17], or "AVISPA" [19]. The symbolic methods have been used to analyse a wide class of cryptographic protocols such as key exchange and authentication protocols, voting protocols [136, 97], contract-signing protocols [128], recursive protocols [138], Web Services [41, 72], and others.

(Un)Decidability results

The security problem of cryptographic protocols, stated in Figure 1.2 is undecidable in general [111], but under some restrictions several decidability results

have been obtained. These results can be divided into two classes: the decidability results with (respectively without) the *perfect cryptography hypothesis*. An encryption scheme is said to be *perfect* when no (not even partial) information about the plaintext can be obtained from a ciphertext without knowing the decryption (secret) key. This hypothesis has been introduced by R. Needham and M. Schroeder [163], and D. Dolev and A. Yao [107] in their respective works. When the *perfect encryption hypothesis* is generalised to the other cryptographic primitives, we talk about the *perfect cryptography hypothesis*.

While *perfect cryptography hypothesis* is not realistic, (actually the intruder may use the algebraic properties of cryptographic primitives when he is attacking the protocol), several important attacks have been discovered. An example of such attacks would be the attack discovered by G. Lowe on the Needham-Schroeder public key [145]. Furthermore, several results have been obtained, we show below some of them.

Unbounded number of sessions. Assuming an unbounded number of sessions, the problem is undecidable [14, 81, 110, 111], and remains undecidable even if we suppose that no nonces (fresh data) are generated [81, 111], or we bound the size of messages [14, 110]. This problem becomes DEXPTIME-complete if we suppose that no nonces are generated and bound the size of messages [71, 110]. In [106], the authors showed that the security of Ping-pong protocols is decidable in PTIME.

Bounded number of sessions. Assuming a bounded number of sessions, M. Rusinowitch and M. Turuani [178] showed that the problem is co-NP-complete.

Later, The *perfect encryption hypothesis* has been relaxed, and several algebraic properties of cryptographic primitives have been considered in the analysis of cryptographic protocols. Several results have been obtained, we show below some of them. The problem is co-NP-complete for the Ping-pong protocols with *commutative equational theory* [69, 195]. The problem remains decidable for the bounded number of sessions with *exclusive Or* [87]. It also remains decidable for the *abelian groups* [188].

Many researches have been focused on general classes of algebraic properties, and several results have been obtained in that field. For example, M. Baudet [32] has proved the decidability of the security problem for the class of cryptographic protocols using primitives represented by subterm convergent equational theories. S. Delaune and F. Jacquemard [95] have proved the decidability of the security problem for the class of cryptographic protocols using primitives represented by convergent public-collapsing equational theories, *etc*. Other results can be found in [90].

1.5 Contributions and plan of this thesis

In this thesis, we relax the perfect cryptography hypothesis by taking into account some algebraic properties of cryptographic primitives that we formulate by equations. We follow the symbolic approach to analyse security protocols, and in particular, the approach based on the resolution of constraint systems. To this end, we formulate the capacity of the intruder by deduction rules, and the verification task of the protocol by a reachability problem. The latter is the problem of determining if a certain (finite) parallel program which models the protocol and the specification can reach an erroneous state while interacting with the environment. We provide decision procedures for the reachability problem in presence of several algebraic operators.

In Chapter 2, we give the basic notions and definitions we use in the most of this thesis. We define the *constraint systems*, and *reachability problem*. These notions have been initially introduced by J. Millen and V. Shmatikov [156], but as defined there, they are not adequate for the non empty equational theories. We actually follow the definitions introduced by Y. Chevalier and M. Rusinowitch [73] who generalised the initial definitions of [156] in order to capture non empty equational theories. We show how protocols are modeled in a high specification language, and we show how to reduce the insecurity problem of cryptographic protocols to the satisfiability problem of constraint systems. Several works follow this approach [156, 87, 73, 72].

1.5.1 Decidability results in presence of algebraic operators

Chapter 3: Analysis of protocols with collision vulnerable hash functions.

In Chapter 3, we consider the class of cryptographic protocols that use collision vulnerable hash functions. The collision vulnerability property for a hash function means that one can construct two different messages having the same hash value. We remark that only a few years ago, it was intractable to compute collisions on hash functions, so they were considered to be collision resistant by cryptographers, and collision was considered to be a possible attack on hash functions only from the nineties when collision attacks have been proved and showed by several ressearchers [98, 103, 199, 201]. Examples of collision vulnerable hash functions are "MD5" and "SHA-0".

In this chapter, we symbolically represent how the intruder may compute collisions on hash functions. We then reduce the insecurity problem of our class of cryptographic protocols to the ordered satisfiability problem for the intruder using the collision vulnerability property of hash functions when attacking a protocol execution. The ordered satisfiability problem is a variant of the satisfiability problem presented in Chapter 2. It was initially introduced by Y. Chevalier *et al.* [72]. Roughly following the results obtained in [74], we conjecture that

the ordered satisfiability problem for the intruder exploiting the collision vulnerability property of hash functions can be reduced to the ordered satisfiability problem for an intruder operating on words, that is with an associative symbol of concatenation. We show the decidability of the last problem, which is interesting in its own as it is the first decidability result that we are aware of for an intruder system for which unification is infinitary. Furthermore, it permits one to consider in other contexts an associative concatenation of messages instead of their pairing.

Chapter 4: Analysis of protocols with vulnerable digital signature schemes.

In chapter 4, we study two classes of cryptographic protocols: the class of protocols using digital signature schemes vulnerable to *constructive exclusive ownership property*, and the class of protocols using digital signature schemes vulnerable to *destructive exclusive ownership property*. The *constructive exclusive ownership vulnerability property* for a digital signature schemes permits the intruder, given a public verification key and a signed message, to compute a new pair of signature and verification keys such that the message appears to be signed with the new signature key; and the *destructive exclusive ownership vulnerability property* for a digital signature schemes permits the intruder, given a public verification key and a signed message, to compute a new pair of signature and verification keys, and a new message, such that the given signature appears to be the signature of the new computed message with the new signature key. We show the decidability of the insecurity problem for these two classes of cryptographic protocols, and that by reducing the insecurity problem to the reachability problem for our intruder deduction systems.

Chapter 5: Decidability results for saturated deduction systems.

In chapter 5, we generalise the results obtained in Chapter 4. We consider the class of cryptographic protocols where the cryptographic primitives are represented by equational theories generated by convergent rewrite systems having the *finite variant property*. This property has been introduced in [86], and means that one can compute all possible normal forms of the instances of a term t.

First, we show the decidability of the ground reachability problem for our class of deduction systems. This decidability result is obtained by (1) reducing the reachability problem modulo an equational theory to the reachability problem modulo the empty theory, (2) computing a transitive closure of the possible deductions (using a *saturation procedure*), and (3) showing that the termination of this computation implies the decidability of the ground reachability problem.

Next, we give a new criterion, based on counting the number of variables in a reachability problem before and after a deduction is guessed, that permits

us to reduce the general reachability problem to the ground reachability problem. This criterion is a generalisation of the one employed for the specific cases in Chapter 4, and we give an example showing that such additional criterion is needed, that is the decidability of the ground reachability problem without this criterion does not imply the decidability of the general reachability problem. Another contribution of this chapter is a decidability result of the ground reachability problem for the theory of blind signature [136], and a decidability result of the general reachability problem for a class of subterm convergent equational theories. Other decidability results have been obtained for the theory of blind signature in [9, 91] but they are different from our. Similarly, a more general decidability result for the subterm convergent theory was given in [31], but our proofs are simpler and can be easily generalised to other classes.

1.5.2 Chapter 6: Decidability result for the ground entailment problem in the first order logic

Deduction systems representing the intruder's deductive capabilities can be viewed as sets of Horn clauses with one unary predicate. We generalise in Chapter 6 the saturation procedure employed in Chapter 5 in order to study the ground entailment problem for a new set of first order clauses. It is well-known that the satisfiability and the ground entailment problem are undecidable for both clauses and Horn clauses sets, but several decidability results have been obtained for several fragments of first order logic [150, 28, 84, 180, 205]. In this chapter, we introduce a new fragment of first order logic and we prove the decidability of its ground entailment problem. This decidability result relies on the use of the *selected resolution* (widely studied in the literature [134, 133, 137, 146, 164]) and on the use of *an atom ordering compatible with a complete simplification term ordering*. We remark that when the complete term ordering is arbitrary, a saturated set of clauses does not necessarily have a decidable ground entailment problem. We also show how to use this result in order to decide the insecurity problem for cryptographic protocols in the case of bounded number of sessions.

While in this chapter the application of Horn clauses on security protocols is limited to the search of attacks, the analysis of cryptographic protocols using Horn clauses may go beyond that: actually one can use Horn clauses to prove the correctness of such protocols, and that by including the clauses describing the protocol in the saturation process.

1.5.3 Chapter 7: Analysis of electronic voting protocols: "voter verifiability" property

In Chapter 7, we study a specific class of cryptographic protocols: the "electronic-voting" protocols for which we analyse the "voter verifiability property". The *voter verifiability property* includes the the *individual verifiability* and *universal verifiability* properties. Intuitively, the *individual verifiability property* mans that a voter can check whether her ballot was included in the tally, and the *universal verifiability property* means that anybody can check the correctness of the published outcome. We formally define these two properties, and that using the *applied pi calculus* [11] which is well-suited for modelling security protocols and in particular electronic voting protocols [97, 25]. We apply our definition to some well-known e-voting protocols such as the protocol due to Fujioka, Okamoto & Ohta [115] and Lee *et al.* [141], and show that both protocols are voter verifiable according to our definition.

Chapter 2

Constraint systems to analyse cryptographic protocols

The security of a cryptographic protocol is assessed with respect to the environment in which the protocol is executed. Dolev and Yao [107] have described the environment by the deductions an intruder attacking a protocol is able to perform. They considered that the intruder has a complete control over the communication medium (network), he listen to the communication and can obtain any message passing through the network, he can intercept, block, and/or redirect all messages sent by honest agents. He also can masquerade his identity and take part in the protocol under the identity of an honest agent. His control of the network is modelled by assuming that all messages sent by honest agents are sent directly to the intruder and that all messages received by the honest agents are always sent by the intruder. Besides the control of the network, the intruder has some specific rules to deduce new messages [73, 72, 156].

Many procedures have been proposed to decide security problems of cryptographic protocols in the Dolev-Yao model with respect to a finite number of sessions [16, 53, 178]. Among the different approaches, there is the symbolic approaches [156, 75, 29, 87] on which, as mentionned in the introduction of this document, we work. In these approaches, cryptographic primitives are represented by function symbols, messages are represented by terms in a signature (a set of function symbols representing the cryptographic primitives), and intruder capacities are represented by *deduction rules* on sets of messages representing his knowledge. These deduction rules allow the intruder to derive new messages from a given (finite) set of messages representing his knowledge. Cryptographic primitives may have algebraic properties, such as the associativity for the concate-

nation primitive "$\cdot$", which are represented by an *equational theory*.

Among the different symbolic approaches, in this chapter we are focused on the symbolic approach that uses the resolution of constraint systems and reduces the insecurity problem of cryptographic protocols with bounded number of sessions to the satisfiability problem of constraint systems [205, 156, 73, 178]. In [29], the authors proved that this reduction is quite effective and were able to discover new attacks on several protocols.

Outline of the Chapter. We start by giving the basic technical definitions and notions used troughout this document in Section 2.1. In Section 2.2, we present how protocols are modeled. In Section 2.3, we show how the insecurity problem of cryptographic protocols can be reduced to the satisfiability problem of constraint systems.

2.1 Preliminaries

This section introduces the basic notions used in this document.
Given a set of elements E, we denote by E^* the set of all words over E in the sense of formal language theory. For example, if $E = \{1, 2, 3\}$, elements of E^* are of the form $\{\epsilon, 1, 2, 3, 1.2, 1.2.2.3, \ldots\}$ where "." denotes the *concatenation* and ϵ denotes the *empty word*. The cardinality of a set E is denoted by $\sharp E$. By infinite set, we mean a countably infinite set (*i.e.* the same cardinality as $\mathbb{N}$).

2.1.1 Multisets

Let E be a set of elements. A multiset $\mathcal{M}$ over E can be formally defined as a pair (E, f) where f is a function from E to $\mathbb{N}$. The multiset $\mathcal{M}$ can also be written as $\{(e_1, f(e_1)), (e_2, f(e_2)), \ldots\}$, or as $\{\underbrace{e_1, \ldots, e_1}_{f(e_1)}, \underbrace{e_2, \ldots, e_2}_{f(e_2)}, \ldots\}$ where $e_i \in E$. The support of the multiset $\mathcal{M}$, written $Supp(\mathcal{M})$, is defined as follows

$$Supp(\mathcal{M}) = \{e \text{ such that } f(e) > 0\}$$

A multiset $\mathcal{M}$ is said to be finite if the cardinality of its support is finite. For simplicity, we sometimes denote by $\mathcal{M}(e)$ the number of occurrences of the element e in the multiset $\mathcal{M}$.

2.1.2 Terms over a signature

Let $\mathcal{X}$ be an infinite set of variables, $\mathcal{C}$ be an infinite set of free constants , and $\mathcal{F}$ be a set of function symbols. We associate a special function $arity$ to

the function symbols, $arity : \mathcal{F} \rightarrow \mathbb{N}$. The arity of a function symbol indicates the number of arguments that it expects. We call *signature* the tuple $(\mathcal{F}, arity)$, and for simplicity, in this document, we denote by *signature* only the set $\mathcal{F}$. We define the set of *terms* $\mathcal{T}(\mathcal{F}, \mathcal{X})$ over the signature $\mathcal{F}$ to be the smallest set containing $\mathcal{X}$, $\mathcal{C}$, and such that if f is a function symbol in $\mathcal{F}$ with arity $n \geq 0$, and if $t_1, \ldots, t_n$ are terms in $\mathcal{T}(\mathcal{F}, \mathcal{X})$, then $f(t_1, \ldots, t_n)$ is a term in $\mathcal{T}(\mathcal{F}, \mathcal{X})$. We define the set of *ground terms* $\mathcal{T}(\mathcal{F})$ over the signature $\mathcal{F}$ to be biggest subset of the set of terms $\mathcal{T}(\mathcal{F}, \mathcal{X})$ such that $\mathcal{T}(\mathcal{F})$ does not contain variables. We denote the set of variables (respectively set of free constants) occurring in a term t by $Var(t)$ (respectively $Cons(t)$). A term without variables, *i.e.* a term in $\mathcal{T}(\mathcal{F})$, is called *ground term*. If T, T' are two sets of terms, and t, t' are two terms, we abbreviate $T \cup T'$ by T, T', $T \cup \{t\}$ by T, t, $T \setminus \{t\}$ by $T \setminus t$, and $\{t, t'\}$ by t, t'.

2.1.3 Positions and subterms

The *subterms* of a term t are denoted by the set of terms $Sub(t)$ which is defined recursively as follows. If $t \in \mathcal{X} \cup \mathcal{C}$ then $Sub(t) = \{t\}$, and if $t = g(t_1, \ldots, t_n)$ then $Sub(t) = \{t\} \cup (\bigcup_{i=1}^{n} Sub(t_i))$. The *strict (or proper) subterms* of a term t are denoted by the set of terms $SSub(t)$, and $SSub(t) = Sub(t) \setminus t$. We denote $t[s]$ a term t that admits s as subterm.

Example 4 *Consider the term* $t = Sig(h(a), Sk(Bob))$.
We have: $Sub(t) = \{t, h(a), Sk(Bob), a, Bob\}$, *and* $SSub(t) = \{h(a), Sk(Bob), a, Bob\}$.

We call *positions* the elements of $\mathbb{N}^*$. We say that a position p is smaller than a position q, and we write $p \leq q$, if p is a *prefix* of q, *i.e.* there exists a position p' such that $p.p' = q$. Given a term t, the set of *positions of* t, denoted by $Pos(t)$, is defined recursively as follows:

- if t is a variable or a constant then $Pos(t) = \{\epsilon\}$;
- if $t = f(t_1, \ldots, t_n)$ then $Pos(t) = \{\epsilon\} \cup (\bigcup_{1 \leq i \leq n} \{i.p \text{ such that } p \in Pos(t_i)\})$.

The position ϵ is called the *root position* of the term t. The *dag-size* of a term t, denoted by $\|t\|_{dag}$, is defined to be the number of distinct subterms of t.

The Top symbol of a term t, denoted by $Top(t)$, is defined by $Top : \mathcal{T}(\mathcal{F}, \mathcal{X}) \rightarrow \mathcal{F} \cup \mathcal{X} \cup \mathcal{C}$, with

- $Top(t) = t$ if $t \in \mathcal{X} \cup \mathcal{C}$
- $Top(t) = f$ if $t = f(t_1, \ldots, t_n)$

The *subterm* of t at position $p \in Pos(t)$, denoted by $t_{|p}$, is defined recursively as follows:

- if $t \in \mathcal{X} \cup \mathcal{C}$ or $p = \epsilon$ and $t_{|p} = t$;
- if $t = f(t_1, \ldots, t_n)$ and $p = i.p'$ with $1 \leq i \leq n$ then $t_{|p} = t_{i|p'}$.

Given a term t, and a position $p \in Pos(t)$, we denote by $t[p \leftarrow s]$ the term obtained from t by replacing the subterm at position p by s.

2.1.4 Substitutions

A substitution σ is a partial function from variables $\mathcal{X}$ to terms $\mathcal{T}(\mathcal{F}, \mathcal{X})$, such that its *domain* is finite. We define the *support* of a substitution σ, written $Supp(\sigma)$ as follows: $Supp(\sigma) = \{x | \sigma(x) \neq x\}$, is a finite set. The substitution σ with $Supp(\sigma) = \emptyset$ is called the *empty substitution* or the *identity substitution*. The substitution σ with $Supp(\sigma) = \{x_1, \ldots, x_n\}$ and $\sigma(x_i) = t_i$ for $1 \leq i \leq n$, can also be written as $\sigma = \{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$. The *range* of σ, denoted by $Ran(\sigma)$, is defined by the set $Ran(\sigma) = \{\sigma(x) \text{ such that } x \in Supp(\sigma)\}$. The substitution σ with $Supp(\sigma) = \emptyset$ is called the *identity substitution*. We denote by $Var(\sigma)$ the set $Var(Ran(\sigma))$. A substitution σ is said to be *ground* if $Var(\sigma) = \emptyset$, that is $Ran(\sigma)$ is a set of ground terms. A substitution σ instantiates a variable x if $x \in Supp(\sigma)$, and σ is said to be *grounding* x if $x \in Supp(\sigma)$ and $\sigma(x)$ is a ground term.

The application of a substitution σ to a term t is denoted $\sigma(t)$ and is equal to the term t where all variables x have been replaced by the term $\sigma(x)$. More formally, in order to apply a substitution σ to terms, we extend σ homomorphically on terms by the rule $\sigma(f(t_1, \ldots, t_n)) = f(\sigma(t_1), \ldots, \sigma(t_n))$. Let E be a set of terms, $E \subseteq \mathcal{T}(\mathcal{F}, \mathcal{X})$, we denote $\sigma(E) = \{\sigma(t) \text{ such that } t \in E\}$.

From now on, we mean by $x\sigma$ the application of σ to x, *i.e.* $x\sigma = \sigma(x)$. In similar way, if t (respectively E) is a term (respectively a set of terms), we will write $t\sigma$ (respectively $E\sigma$) instead of $\sigma(t)$ (respectively $\sigma(E)$).

A *renaming* ρ is an injective substitution such that $Ran(\rho) \subseteq \mathcal{X}$.

The *composition* $\sigma\theta$ of two substitutions σ and θ is defined as $x(\sigma\theta) = (x\sigma)\theta = x\sigma\theta$. A substitution θ is an *extension* of a substitution σ if $Supp(\sigma) \subseteq Supp(\theta)$, and $x\sigma = x\theta$ for all $x \in Supp(\sigma)$. A substitution σ is a *restriction* of a substitution θ if θ is an extension of σ. A substitution σ is *cyclic* if there exists $x_1, \ldots, x_n, x_{n+1} \in Supp(\sigma)$ with $n \geq 1$ such that $x_{i+1} \in Var(x_i\sigma)$ for all $1 \leq i \leq n$, with $x_{n+1} = x_1$. A substitution σ is *idempotent* if $\sigma = \sigma\sigma$. We remark that a idempotent substitutions are acyclic, and as we consider substitutions with finite domain, the converse also holds. In this document, we will only consider *idempotent substitutions*.

2.1.5 Equational theories and rewriting systems

Given a binary relation $\rightarrow$ over a set of terms S. The relations $\rightarrow^+, \rightarrow^*$ are respectively the transitive and reflexive-transitive closure of $\rightarrow$. The relation $\rightarrow$ is said to be:

- *well-founded* if there is no infinite chain $t \to t_1 \to \ldots$ for any term $t \in S$;
- *monotone* if $s \to t$ implies $s\sigma \to t\sigma$ for any substitution σ, and any terms s, t;
- *stable* if $s \to t$ implies $u[s] \to u[t]$ for any terms s, t and u;
- *a reduction* if it is stable, monotone and well-founded.

An equation over the signature $\mathcal{F}$ is an unordered pair of terms $\{u, v\}$, denoted $u \doteq v$ or $v \doteq u$, with $u, v \in \mathcal{T}(\mathcal{F}, \mathcal{X})$. An *equational presentation* $\mathcal{H}$ over the signature $\mathcal{F}$ is a set of equations $u \doteq v$ over $\mathcal{F}$, such that for any $u \doteq v \in \mathcal{H}$, $Cons(\{u, v\}) = \emptyset$. Given an equational presentation $\mathcal{H}$, we denote by $\leftrightarrow_{\mathcal{H}}$ the smallest symmetric relation containing $\mathcal{H}$, stable and monotone. Hence, for any couple of terms s, t, we have $s \leftrightarrow_{\mathcal{H}} t$ if and only if there exists an equation $u \doteq v \in \mathcal{H}$, a substitution σ, a position $p \in Pos(s)$ such that $s_{|p} = u\sigma$, and $t = s[p \leftarrow v\sigma]$.

The *equational theory generated* by $\mathcal{H}$ on $\mathcal{T}(\mathcal{F}, \mathcal{X})$, denoted by $=_{\mathcal{H}}$, is the smallest monotone congruence containing $\mathcal{H}$. We remark that a congruence is stable by definition. For any terms s, t, we say that s and t are equal modulo $\mathcal{H}$ if $s =_{\mathcal{H}} t$. We often do not distinguish between an equational theory generated by $\mathcal{H}$ and $\mathcal{H}$ itself. An equational theory $\mathcal{H}$ is said to be *consistent* if two free constants are not equal modulo $\mathcal{H}$.

A *rewrite rule* over the signature $\mathcal{F}$ is a pair of terms denoted $l \to r$ such that $Var(r) \subseteq Var(l)$. A *rewrite system* $\mathcal{R}$ over the signature $\mathcal{F}$ is a set of rewrite rules over $\mathcal{F}$. The *reduction relation* $\to_{\mathcal{R}}$ induced from $\mathcal{R}$ is the smallest relation containing $\mathcal{R}$, stable and monotone.

For any couple of terms t, s, we have that s reduces to (or can be reduced to, or rewrites to, or can be rewritten to) t, and we write $s \to_{\mathcal{R}} t$ or simply $s \to t$ if $\mathcal{R}$ is clear from the context, if and only if there exists a rewrite rule $l \to r \in \mathcal{R}$, a substitution σ, a position $p \in Pos(s)$ such that $s_{|p} = l\sigma$, and $t = s[p \leftarrow r\sigma]$. We call $s \to_{\mathcal{R}} t$ a *reduction step*, and a sequence of reduction steps is called *reduction sequence*. We write $s \to^*_{\mathcal{R}} t$ if we have $s \to_{\mathcal{R}} s_1 \to_{\mathcal{R}} \ldots \to_{\mathcal{R}} t$. We write $s \leftrightarrow_{\mathcal{R}} t$ if we have $s \to_{\mathcal{R}} t$ and $t \to_{\mathcal{R}} s$, and we write $s \leftrightarrow^*_{\mathcal{R}} t$ if we have $s \leftrightarrow^*_{\mathcal{R}} s_1 \leftrightarrow^*_{\mathcal{R}} \ldots \leftrightarrow^*_{\mathcal{R}} t$.

A term t is said to be in *$\mathcal{R}$-normal form* (or simply in *normal form* when $\mathcal{R}$ is clear from the context) if and only if there is no term s such that $t \to_{\mathcal{R}} s$. A term s is the *$\mathcal{R}$-normal form* (or simply the *normal form* when $\mathcal{R}$ is clear from the context) of the term t if and only if $t \to^*_{\mathcal{R}} s$ and s is in normal form. The normal form of a term t is denoted $t \downarrow_{\mathcal{R}}$ (or simply $t \downarrow$ when $\mathcal{R}$ is clear from the context).

A rewrite system $\mathcal{R}$ is said to be *terminating* if the reduction relation $\to_{\mathcal{R}}$ is well-founded, that is any reduction sequence is finite. A rewrite system $\mathcal{R}$ is said to be *confluent* if for any terms t, u, v such that $t \to^*_{\mathcal{R}} u$, $t \to^*_{\mathcal{R}} v$, there exists a term w verifying $u \to^*_{\mathcal{R}} w$ and $v \to^*_{\mathcal{R}} w$. If $\mathcal{R}$ is confluent, then when a term t has

a normal form, this normal form is unique. A rewriting system is said to be *convergent* if it is confluent and terminating. If $\mathcal{R}$ is convergent, each term has a normal form which is unique, and for any terms s, t we have $s \leftrightarrow^*_{\mathcal{R}} t$ if and only if $s \downarrow = t \downarrow$ [21].

A rewrite system $\mathcal{R}$ is said to be *ground confluent* if for any ground terms t, u, v such that $t \rightarrow^*_{\mathcal{R}} u$, $t \rightarrow^*_{\mathcal{R}} v$, there exists a ground term w verifying $u \rightarrow^*_{\mathcal{R}} w$ and $v \rightarrow^*_{\mathcal{R}} w$. A rewrite system $\mathcal{R}$ is said to be *ground convergent* if it is ground confluent and terminating. If $\mathcal{R}$ is ground convergent, each ground term has a normal form which is unique.

A substitution σ is said to be in normal form if for all $x \in Supp(\sigma)$, the term $x\sigma$ is in normal form.

2.1.6 The completion procedures

Given an equational theory $\mathcal{H}$, and a rewrite system $\mathcal{R}$, we say that $\mathcal{H}$ *is generated by* $\mathcal{R}$ if $\mathcal{R}$ and $\mathcal{H}$ are equivalent, that is for any terms s and t, we have $s =_{\mathcal{H}} t$ if and only if $s \leftrightarrow^*_{\mathcal{R}} t$ [21].

In this section, we present techniques to construct a convergent rewrite system or a ground convergent rewrite system generating a given equational theory. Given an equational theory $\mathcal{H}$, many procedures aiming to construct a convergent (or a ground convergent) rewrite system $\mathcal{R}$ equivalent to $\mathcal{H}$ have been given in the literature, for example [131, 23, 120]. In this section, we make use of the notions of *two terms are $\emptyset$-unifiable* and *a most general $\emptyset$-unifier of two terms*. Let u, v be two terms, a *$\emptyset$-unifier* (or *unifier in the $\emptyset$-theory*) of u and v is a substitution σ such that $u\sigma = v\sigma$. We say that u and v are unifiable in the $\emptyset$-theory (or *syntactically unifiable*, or *$\emptyset$-unifiable*) if they have a $\emptyset$-unifier. We say that a substitution σ is *more general modulo* $\emptyset$ than a substitution σ', and we write $\sigma \lesssim \sigma'$, if there exists a substitution θ such that $\sigma' = \sigma\theta$. Given two terms u and v, it is well-known that if u and v are $\emptyset$-unifiable then there exists a *unique most general $\emptyset$-unifier* θ, denoted by $mgu(u, v)$, such that for every unifier σ of u, v, there exists a substitution σ' verifying $\sigma = \theta\sigma'$ [22]. The notions of *$\emptyset$-unifier, $\emptyset$-unifiable, most general $\emptyset$-unifier* are generalised to an arbitrary theory in Section 2.1.7.

Given a pair of rewrite rules $l \rightarrow r$ and $g \rightarrow d$, and an integer $p \in Pos(l)$ such that $l_{|p} \notin \mathcal{X}$, $l_{|p}$ and g are $\emptyset$-unifiable with σ is their most general $\emptyset$-unifier, the pair $\langle l[d]_p)\sigma, r\sigma\rangle$ is a *critical pair* of the rules $l \rightarrow r$ and $g \rightarrow d$. The rules $l \rightarrow r$ and $g \rightarrow d$ do not need to be different in order to compute their critical pairs, furthermore, we assume that the rules $l \rightarrow r$ and $g \rightarrow d$ do not share variables, and to this end, we rename their variables before computing their critical pairs.

Given a rewrite system $\mathcal{R}$, we denote by $CP(\mathcal{R})$ the set of all critical pairs obtained from the rules of $\mathcal{R}$.

Figure 2.1 Knuth-Bendix completion procedure

Input:

An equational theory $\mathcal{H}$ and a reduction order $>$ over $\mathcal{T}(\mathcal{F}, \mathcal{X})$.

Output:

A finite convergent rewrite system $\mathcal{R}$ that is equivalent to $\mathcal{H}$, if the procedure terminates successfully;
or "fail", if the procedure terminates unsuccessfully.

Initialisation:

If there exists an equation $l \doteq r \in \mathcal{H}$ such that $l \neq r$, $l \not> r$ and $r \not> l$ then terminates with output fail
otherwise, $i = 0$ and $\mathcal{R}_0 = \left\{ l \to r \text{ such that } l \doteq r \in \mathcal{H} \text{ and } l > r \right\}$

repeat $\mathcal{R}_{i+1} = \mathcal{R}_i$;
for all $\langle l, r \rangle \in CP(\mathcal{R}_i)$ do

1. reduce l, r to some $\mathcal{R}_i$-normal form $l\downarrow, r\downarrow$;
2. if $l\downarrow \neq r\downarrow$ and neither $l\downarrow > r\downarrow$ nor $r\downarrow > l\downarrow$, then terminate with output fail;
3. if $l\downarrow > r\downarrow$, then $\mathcal{R}_{i+1} = \mathcal{R}_i \cup \{l\downarrow \to r\downarrow\}$;
4. if $r\downarrow > l\downarrow$, then $\mathcal{R}_{i+1} = \mathcal{R}_i \cup \{r\downarrow \to l\downarrow\}$;

od
i=i+1;

until $\mathcal{R}_i = \mathcal{R}_{i-1}$;

output $\mathcal{R}_i$;

Knuth-Bendix completion procedure

The *Knuth-Bendix completion procedures* [131] (also called *basic completion procedure*), which is described in Figure 2.1, starts with an equational theory $\mathcal{H}$ and tries to find a convergent rewrite system $\mathcal{R}$ that is equivalent to $\mathcal{H}$. We assume that the order $>$ used in the procedure is a reduction order and it is given as an input of the procedure. We recall that a reduction order $>$ is any order which is stable, well-founded, and monotone (see Section 2.1.5).

The *Knuth-Bendix* procedure removes automatically all trivial equations (respectively trivial critical pairs) of the form $l \doteq l$ (respectively $\langle l, l \rangle$ or $\langle l, r \rangle$ with $l\downarrow = r\downarrow$).

Thus, the basic completion procedure may show three different types of behaviour, depending on the particular input $\mathcal{H}$ and $>$:

1. It may terminate with failure because one of the $\mathcal{H}$-equations can not be ordered using $>$, or the normal forms of the terms in one of the critical pairs are distinct and can not be ordered using $>$. In this case, one could try to run the procedure again using another reduction order;

2. it may terminate successfully with output $\mathcal{R}_n$;

3. it may run for ever since infinitely many new rules are generated.

Given an equational theory $\mathcal{H}$ and a reduction order $>$, in [21], the authors showed that if the basic completion procedure applied on $(\mathcal{H}, >)$ terminates successfully and outputs $\mathcal{R}_n$, then $\mathcal{R}_n$ is a finite convergent rewrite system generating $\mathcal{H}$, and if the basic completion procedure applied on $(\mathcal{H}, >)$ does not terminate, then $\mathcal{R}_\infty = \cup_{i\geq 0}\mathcal{R}_i$ is an infinite convergent rewrite system generating $\mathcal{H}$.

Bachmair completion procedure

The *basic completion procedure* described above usually generates a huge number of rules, and all these rules must be taken into account when computing critical pairs. This implies that both run time and space requirements for the completion process are often too high and unacceptable. In what follows, we present an improved completion procedure that extends basic completion by simplification rules. The goal of this procedure is to transform an initial pair $(\mathcal{H}, \emptyset)$, where $\mathcal{H}$ is an equational theory, into a pair $(\emptyset, \mathcal{R})$ such that $\mathcal{R}$ is a convergent rewrite system equivalent to $\mathcal{H}$.

This procedure, introduced in [23], is described by the set of rules given in Figure 2.2.

A completion procedure is a program that accepts as input an equational theory $\mathcal{H}$ and a reduction order $>$, and uses the rules of Figure 2.2 to generate a (finite or infinite) sequence:

$$(\mathcal{H}_0, \mathcal{R}_0) \vdash (\mathcal{H}_1, \mathcal{R}_1) \vdash \ldots$$

where $\mathcal{H}_0 = \mathcal{H}$, $\mathcal{R}_0 = \emptyset$, and $(\mathcal{H}, \mathcal{R}) \vdash (\mathcal{H}', \mathcal{R}')$ means that $(\mathcal{H}', \mathcal{R}')$ is obtained from $(\mathcal{H}, \mathcal{R})$ by applying a rule from Figure 2.2. This sequence is called a *run* of the completion procedure on inputs $\mathcal{H}$ and $>$.

A run is said to be *fair* if

$$CP(\cup_{i\geq 0} \cap_{j\geq i} \mathcal{R}_j) \subseteq \cup_{i\geq 0}\mathcal{H}_i$$

Given a fair run, G. Huet [121] proved that if there is a step n in the run where $\mathcal{H}_n = \emptyset$ then $\mathcal{R}_n$ is convergent rewrite system equivalent to $\mathcal{H}$.

When an equational theory $\mathcal{H}$ is generated by a convergent rewrite system $\mathcal{R}$, we have that $s =_\mathcal{H} t$ if and only if $s \downarrow = t \downarrow$ [123, 120].

Figure 2.2 Bachmair completion procedure

Deduce:
$\frac{\mathcal{H},\mathcal{R}}{E\cup\{l\doteq r\},\mathcal{R}}$ if $\langle l, r\rangle \in CR(\mathcal{R})$

Orient:
$\frac{\mathcal{H}\cup\{l\doteq r\},\mathcal{R}}{E,\mathcal{R}\cup\{l\to r\}}$ if $l > r$

Delete:
$\frac{\mathcal{H}\cup\{l\doteq l\},\mathcal{R}}{E,\mathcal{R}}$

Simplify-Identity:
$\frac{\mathcal{H}\cup\{l\doteq r\},\mathcal{R}}{\mathcal{H}\cup\{l'\doteq r\},\mathcal{R}}$ if $l\to_{\mathcal{R}} l'$

R-Simplify-Rule
$\frac{\mathcal{H},\mathcal{R}\cup\{l\to r\}}{\mathcal{H},\mathcal{R}\cup\{l\to l'\}}$ if $r\to_{\mathcal{R}} l'$

L-Simplify-Rule
$\frac{\mathcal{H},\mathcal{R}\cup\{l\to r\}}{\mathcal{H}\cup\{l'\doteq r\},\mathcal{R}}$ if $l\to_{\mathcal{R}} l'$

Unfailing Knuth-Bendix procedure

The *basic completion procedure* described above will fail when an initial non trivial equation is unorientable or when it generates an unorientable non trivial critical pair or when it generates infinitely many new rules.

J. Hsiang and M. Rusinowitch [120] introduced an extension of the Knuth-Bendix completion procedure, called the *unfailing completion procedure* or *UKB-procedure* in short, which is a Knuth-Bendix type completion procedure that does not fail. This procedure is described in Figure 2.3.

In [120], the authors make use of a complete simplification ordering $>$, *i.e.* $>$ is well-founded, monotone, stable, total over ground terms, and $s[t] > t$ for any terms s, t. They also make use of *extended critical pairs* defined as follow:

Definition 1 *(extended critical pairs) Given two equations $g \doteq d$ and $l \doteq r$, and an integer $p \in Pos(g)$ such that*

1. $g_{|p} \notin \mathcal{X}$,
2. *$g_{|p}$ and l are unifiable with σ is their most general unifier,*
3. $r\sigma \not\geq l\sigma$,
4. $d\sigma \not\geq g\sigma$.

Figure 2.3 Unfailing Knuth-Bendix procedure

Input:

An equational theory $\mathcal{H}$ and a complete simplification ordering $>$ over $\mathcal{T}(\mathcal{F}, \mathcal{X})$.

Initialisation:

For each equation $l \doteq r \in \mathcal{H}$, orient this equation into rule if possible

repeat

- Find an extended critical pair $\langle l, r \rangle$
- Reduce it using the existing equations as much as possible.
- If the resulting pair is not trivial, orient it into a rule if possible.

until no more new extended critical pair is generated.

Output:

the system constructed from the set of rules and equations obtained at the end of the procedure.

then the pair $\langle d\sigma, g[p \leftarrow r]\sigma \rangle$ is an extended critical pair of the equations $l \doteq r$ and $g \doteq d$. The equations $l \doteq r$ and $g \doteq d$ do not need to be different in order to compute their extended critical pairs, furthermore, we assume that the equations $l \doteq r$ and $g \doteq d$ do not share variables, and to this end, we rename their variables before computing their extended critical pairs.

We remark that if the equations $l \doteq r$ and $g \doteq d$ are orientable, then (2) and (3) become $(2')$ $l\sigma > r\sigma$ and $(3')$ $g\sigma > d\sigma$, and the procedure to construct an extended critical pair become equivalent to the procedure to construct a critical pair.

Given a term s, s is said to be reducible by an equation $l \doteq r$ if there is a position $p \in Pos(s)$, and a substitution σ such that (i) $s_{|p} = l\sigma$, and (ii) $l\sigma > r\sigma$. In this case, we say that s is reducible to $s[p \leftarrow r\sigma]$ using the equation $l \doteq r$.

Given an equational theory $\mathcal{H}$ and a complete simplification ordering $>$, in [120], the authors showed that if the procedure terminates and the obtained system does not contain unorientable equations then the obtained system is a convergent rewrite system equivalent to $\mathcal{H}$. They showed also that if the procedure terminates and the obtained system contains unorientable equations then the obtained system is a ground convergent rewrite system equivalent to $\mathcal{H}$.

In the remainder of this chapter, we assume $\mathcal{F}$ to be a signature, $\mathcal{H}$ to be an equational theory on $\mathcal{T}(\mathcal{F}, \mathcal{X})$, and $\mathcal{R}$ to be a convergent rewrite system generating $\mathcal{H}$.

2.1.7 Unification

The unification is the process of solving the following problem: given an equational theory $\mathcal{H}$, two terms s and t, find a substitution σ such that $s\sigma =_{\mathcal{H}} t\sigma$. The *syntactic unification* or *standard unification* is a special case of the unification where $\mathcal{H} = \emptyset$.

Definition 2 *(Unification systems) Let $\mathcal{H}$ be an equational theory on $\mathcal{T}(\mathcal{F}, \mathcal{X})$. A $\mathcal{H}$-unification system $\mathcal{U}$ is a finite set of pairs of terms in $\mathcal{T}(\mathcal{F}, \mathcal{X})$ denoted by $\mathcal{U} = \{u_1 \stackrel{?}{=}_{\mathcal{H}} v_1, \ldots, u_n \stackrel{?}{=}_{\mathcal{H}} v_n\}$. It is satisfied by a substitution σ, and we note $\sigma \models_{\mathcal{H}} \mathcal{U}$, if for all $i \in \{1, \ldots, n\}$ we have $u_i\sigma =_{\mathcal{H}} v_i\sigma$. In this case we call σ a $\mathcal{H}$-*solution *or a $\mathcal{H}$-*unifier *of $\mathcal{U}$. We say that $\mathcal{U}$ is $\mathcal{H}$-unifiable if it has a $\mathcal{H}$-solution.*

When $\mathcal{H}$ is generated by a convergent rewrite system $\mathcal{R}$, the confluence of $\mathcal{R}$ implies that if σ is a solution of a $\mathcal{H}$-unification system, then $(\sigma)\downarrow$ is also a solution of the same unification system. Actually, if σ is a $\mathcal{H}$-unifier of the terms s, t then $u\sigma =_{\mathcal{H}} v\sigma$, which implies that $(u\sigma)\downarrow = (u\sigma)\downarrow$. Since $(u\sigma)\downarrow = (u(\sigma)\downarrow)\downarrow$, we deduce that if σ is a $\mathcal{H}$-unifier of s, t then $(\sigma)\downarrow$ is also a $\mathcal{H}$-unifier of s, t. Accordingly we will consider only solutions in normal form of unification systems when the equation theory $\mathcal{H}$ is generated by a convergent rewrite system.

Definition 3 *($\mathcal{H}$-Unifiability Problem) Given an equational theory $\mathcal{H}$, the $\mathcal{H}$-unifiability problem is defined as follows:*

Input: *A $\mathcal{H}$-unification system $\mathcal{U}$.*
Output: SAT *if and only if there exists a substitution σ such that $\sigma \models_{\mathcal{H}} \mathcal{U}$.*

Definition 4 *A substitution σ is more general modulo $\mathcal{H}$ than a substitution σ', and we write $\sigma \lesssim_{\mathcal{H}} \sigma'$, if there exists a substitution τ such that $\sigma\tau =_{\mathcal{H}} \sigma'$. When $\mathcal{H} = \emptyset$, the relation $\lesssim_{\mathcal{H}}$ is denoted $\lesssim$.*

The relations $\lesssim_{\mathcal{H}}$ and $\lesssim$ are quasi-order relations [21].

Definition 5 *(Complete set of unifiers) A* complete set of $\mathcal{H}$-unifiers *of a $\mathcal{H}$-unification system $\mathcal{U}$ is a set Σ of $\mathcal{H}$-solutions of $\mathcal{U}$ such that, for any $\mathcal{H}$-solution τ of $\mathcal{U}$, there exists a substitution $\sigma \in \Sigma$ such that $\sigma \lesssim_{\mathcal{H}} \tau$.*

Definition 6 *(minimal complete set of unifiers) A* minimal complete set of $\mathcal{H}$-unifiers *of a $\mathcal{H}$-unification system $\mathcal{U}$ is a complete set Σ_m of $\mathcal{H}$-unifiers of $\mathcal{U}$ that satisfies the condition*

$$\textit{for all substitutions } \sigma, \sigma' \in \Sigma_m, \sigma \lesssim_{\mathcal{H}} \sigma' \textit{ implies } \sigma = \sigma'.$$

Definition 7 *(most general unifiers) Let $\mathcal{H}$ be an equational theory, and let $\mathcal{U}$ be a $\mathcal{H}$-unification system. A substitution σ is a most general $\mathcal{H}$-unifier of $\mathcal{U}$ if and only if there exists a minimal complete set Σ_m of $\mathcal{H}$-unifiers of $\mathcal{U}$, such that $\sigma \in \Sigma_m$.*

Definition 8 *The equational theory $\mathcal{H}$ is of unification type*

Unitary *if for every satisfiable $\mathcal{H}$-unification system $\mathcal{U}$, there exists a minimal complete set of $\mathcal{H}$-unifiers with cardinality 1.*

finitary *if for every satisfiable $\mathcal{H}$-unification system $\mathcal{U}$, there exists a minimal complete set of $\mathcal{H}$-unifiers with finite cardinality.*

infinitary *if for every satisfiable $\mathcal{H}$-unification system $\mathcal{U}$, there exists a minimal complete set of $\mathcal{H}$-unifiers, and there exists an $\mathcal{H}$-unification system for which this set is infinite.*

zero *if there exists an $\mathcal{H}$-unification system that does not have a minimal complete set of $\mathcal{H}$-unifiers.*

The equational theory $\emptyset$ is of type *unitary*, that is, when $\mathcal{H} = \emptyset$, given a satisfiable set of equations $\mathcal{U} = \{u_i \stackrel{?}{=}_\emptyset v_i\}_{i\in\{1,\ldots,n\}}$, $\mathcal{U}$ has a unique most general unifier, denoted by $mgu(\mathcal{U})$ [21]. If $\mathcal{H}$ is *finitary*, then the set of all $\mathcal{H}$-unifiers of a given $\mathcal{H}$-unification system can always be represented as $\mathcal{H}$-instances of finitely many unifiers. The commutativity is a finitary equational theory that is not unitary [21]. A finite representation of all $\mathcal{H}$-unifiers via $\mathcal{H}$-instantiation is not always possible for equational theories of type *infinitary* and *zero*. The associativity theory is an example of infinitary equational theory [21], and the equational theory $\mathcal{H} = \left\{x.(y.z) \doteq (x.y).z, x.x \doteq x\right\}$ is of type zero [20, 184].

2.1.8 Finite variant property

We define in this section what means *an equational theory $\mathcal{H}$ has the finite variant property*. The *finite variant property* has been intially introduced in [86].

Definition 9 *(finite variant property) Let $\mathcal{H}$ be an equational theory generated by a convergent rewrite system $\mathcal{R}$. We say that $\mathcal{H}$ has the finite variant property if for any term t, there is a finite set of substitutions $\Sigma(t)$ such that for any substitution σ, there exists a substitution $\theta \in \Sigma(t)$, and a substitution τ verifying $(\sigma)\downarrow = \theta\tau$ and $(t\sigma)\downarrow = (t\theta)\downarrow\tau$. The substitutions in $\Sigma(t)$ are called variant substitutions of t. We say that $\mathcal{R}$ has the finite variant property if $\mathcal{H}$ has that property.*

It is easy to see that the variant substitutions of any term t are in normal form. The finite variant property and its application for cryptographic protocols are analysed in more details in Chapter 5.

Figure 2.4 Unification algorithm based on the finite variant property

Input:

An arbitrary equational theory $\mathcal{H}$ having the finite variant property, and a $\mathcal{H}$-unification system $\mathcal{U} = \left\{ s_1 \stackrel{?}{=}_{\mathcal{H}} t_1, \ldots, s_n \stackrel{?}{=}_{\mathcal{H}} t_n \right\}$.

Step 1:

Compute the term $M = g(s_1, t_1, \ldots, s_n, t_n)$ where g is a new function symbol not appearing in $\mathcal{F}$ playing the role of cartesian product.

Step 2:

Compute the set of variant substitutions $\Sigma(M)$.

Output:

- *"$\mathcal{U}$ is $\mathcal{H}$-unifiable"* if there is a substitution $\theta \in \Sigma(M)$ and a substitution τ such that $(s_j\theta)\downarrow\tau = (t_j\theta)\downarrow\tau$ for every $j \in \{1, \ldots, n\}$;
- *"$\mathcal{U}$ is not $\mathcal{H}$-unifiable"* otherwise.

Finite variant property and unification

We introduce in this section a general $\mathcal{H}$-unification procedure, where $\mathcal{H}$ is an arbitrary equational theory having the finite variant property. This algorithm is given in Figure 2.4.

It is easy to see that the finite variant property reduces the $\mathcal{H}$-unifiability problem to the syntactic unifiability problem. We show next the correctness and completeness of this unification procedure.

Lemma 1 *Let $\mathcal{H}$ be an equational theory having the finite variant property, and let s and t be two terms. Let $M = g(s, t)$ where g is a new function symbol playing the role of cartesian product. If there exists a variant substitution θ of M and a substitution τ such that $(s\theta)\downarrow\tau = (t\theta)\downarrow\tau$ then s and t are $\mathcal{H}$-unifiable and $\sigma = \theta\tau$ is one of their possible $\mathcal{H}$-unifiers.*

PROOF.

Let s, t be two terms such that $(s\theta)\downarrow\tau = (t\theta)\downarrow\tau$ where θ is a variant substitution of $g(s, t)$ and τ an arbitrary substitution. $(s\theta)\downarrow\tau = (t\theta)\downarrow\tau$ implies $(s\theta)\downarrow\tau =_{\mathcal{H}} (t\theta)\downarrow\tau$. Let $\sigma = \theta\tau$, we have $(s\sigma)\downarrow = (s\theta\tau)\downarrow = ((s\theta)\downarrow\tau)\downarrow$ and similarly, $(t\sigma)\downarrow = ((t\theta)\downarrow\tau)\downarrow$. Since $(s\theta)\downarrow\tau =_{\mathcal{H}} (t\theta)\downarrow\tau$, we conclude that $((s\theta)\downarrow\tau)\downarrow = ((t\theta)\downarrow\tau)\downarrow$ and hence $(s\sigma)\downarrow = (t\sigma)\downarrow$ which implies that σ is an $\mathcal{H}$-unifier of s and t. ■

Lemma 2 *Let $\mathcal{H}$ be an equational theory having the finite variant property, and let s and t be two terms. Let $M = g(s, t)$ where g is a new function symbol playing the*

role of cartesian product. If there exists a substitution σ such that $s\sigma =_{\mathcal{H}} t\sigma$ then there exists a variant substitution θ of M and a substitution τ such that $(s\theta)\downarrow\tau = (t\theta)\downarrow\tau$ and $(\sigma)\downarrow = \theta\tau$.

PROOF.

Let s and t be two terms, and let σ be a $\mathcal{H}$-unifier of s and t, that is $s\sigma =_{\mathcal{H}} t\sigma$. Let $M = g(s,t)$ where g represents the cartesian product, and let $\Sigma(M)$ be the finite set of variant substitutions of M. Definition 46 implies that there exists a substitution $\theta \in \Sigma(M)$ and a substitution τ such that $(\sigma)\downarrow = \theta\tau$ and $(M\sigma)\downarrow = (M\theta)\downarrow\tau$. We have $(g(s,t)\sigma)\downarrow = g((s\sigma)\downarrow,(t\sigma)\downarrow)$ and $(g(s,t)\theta)\downarrow\tau = g((s\theta)\downarrow\tau,(t\theta)\downarrow\tau)$. Since g is a new free function symbol, $(g(s,t)\sigma)\downarrow = (g(s,t)\theta)\downarrow\tau$ implies that $(s\sigma)\downarrow = (s\theta)\downarrow\tau$ and $(t\sigma)\downarrow = (t\theta)\downarrow\tau$. Due to the equality $(s\sigma)\downarrow = (t\sigma)\downarrow$, we conclude the lemma.■

The finite variant property implies that the set of variant substitutions for any term t is finite. It is then easy to see that the algorithm given in Figure 2.4 is a terminating $\mathcal{H}$-unification algorithm.

Definition 10 *Let $\mathcal{U} = \left\{ s_1 \stackrel{?}{=}_{\mathcal{H}} t_1, \ldots, s_n \stackrel{?}{=}_{\mathcal{H}} t_n \right\}$ be a $\mathcal{H}$-unification system, and let the term $T_{\mathcal{U}} = g(s_1,t_1,\ldots,s_n,t_n)$ where g is a new function symbol representing the cartesian product. We define the set $Sol_{Var}(\mathcal{U})$ as follows:*

$$Sol_{Var}(\mathcal{U}) =$$

$\{\theta$ *such that* $\theta = (\alpha\tau)\downarrow$ *where* α *is a variant substitution of* $T_{\mathcal{U}}$ *and* $\tau = mgu_{\emptyset}((U\alpha)\downarrow\}$

Lemma 3 *For any $\mathcal{H}$-unification system $\mathcal{U} = (s_1 \stackrel{?}{=}_{\mathcal{H}} t_1, \ldots, s_n \stackrel{?}{=}_{\mathcal{H}} t_n)$, $Sol_{Var}(\mathcal{U})$ is a complete set of $\mathcal{H}$-unifiers of $\mathcal{U}$.*

PROOF.

Let $\mathcal{U} = (s_1 \stackrel{?}{=}_{\mathcal{H}} t_1, \ldots, s_n \stackrel{?}{=}_{\mathcal{H}} t_n)$ be a $\mathcal{H}$-unification system. We first prove that any substitution in $Sol_{Var}(\mathcal{U})$ is a $\mathcal{H}$-unifier of $\mathcal{U}$. Let σ be a substitution in $Sol_{Var}(\mathcal{U})$, this means that there exists a variant substitution of $T_{\mathcal{U}}$ such that $\sigma = (\theta\tau)\downarrow$ where $\tau = mgu_{\emptyset}((U\theta)\downarrow)$. Lemma 1 implies that $\theta\tau \models_{\mathcal{H}} \mathcal{U}$, which implies that $(\theta\tau)\downarrow \models_{\mathcal{H}} \mathcal{U}$, and hence $\sigma \models_{\mathcal{H}} \mathcal{U}$.

Next, we show that $Sol_{Var}(\mathcal{U})$ is a complete set of $\mathcal{H}$-unifiers of $\mathcal{U}$. Let σ' be a $\mathcal{H}$-unifier of $\mathcal{U}$, Lemma 2 implies that there exists a variant substitution θ of $T_{\mathcal{U}}$, such that $(\sigma')\downarrow = \theta\alpha$ where α is a $\emptyset$-unifier $(U\theta)\downarrow$. Since $\alpha \models_{\emptyset} (U\theta)\downarrow$, there exists a most general $\emptyset$-unifier τ of $(U\theta)\downarrow$, a substitution τ' such that $\alpha = \tau\tau_1$. We have that $(\sigma')\downarrow = \theta\alpha$, then $\sigma' =_{\mathcal{H}} \theta\alpha$, and then $\sigma' =_{\mathcal{H}} \theta\tau\tau_1 =_{\mathcal{H}} (\theta\tau)\downarrow\tau_1$. By definition of θ and τ, we have $(\theta\tau)\downarrow \in Sol_{Var}(\mathcal{U})$ and hence we conclude the proof. ■

2.1.9 Narrowing

Narrowing is a process that can be used as a general $\mathcal{H}$-unification procedure where $\mathcal{H}$ is an equational theory.

The idea of narrowing was first mentioned by *J. R. Sagle* [190] and *D. Lankford* [140], and the first description of narrowing as a general $\mathcal{H}$-unification procedure is due to *M. Fay* [113].

Let $\mathcal{H}$ be an equational theory generated by the convergent rewrite system $\mathcal{R}$. If σ is a unifier of the equation $s \stackrel{?}{=}_{\mathcal{H}} t$, then $s\sigma$ and $t\sigma$ have a common $\mathcal{R}$-normal form, that is, there are chains of reduction steps starting from $s\sigma$ and $t\sigma$ that lead to the same $\mathcal{R}$-irreducible term. We recall that an $\mathcal{R}$-irreducible term is any term t such that there is no term t' verifying $t \rightarrow_{\mathcal{R}} t'$. The main idea underlying narrowing is to construct the unifier and the corresponding chains of reductions simultaneously.

The narrowing relation induced by the rewrite system $\mathcal{R}$, denoted by $\rightsquigarrow_{\mathcal{R}}$, is defined as follows [21, 123]:

Definition 11 *(Narrowing) Let s and t be two terms. We say that t is narrowed to s and we write $t \rightsquigarrow_{\mathcal{R}} s$ if and only if (1) there exists a renamed rule $l \rightarrow r$ of $\mathcal{R}$ such that t and $l \rightarrow r$ have no variables in common, (2) a position $p \in Pos(t)$ such that (2.1) $t_{|p} \notin \mathcal{X}$, and (2.2) the terms $t_{|p}$, l are syntactically unifiable, let σ be their most general $\emptyset$-unifier, and (3) $s = t[r]_p\sigma$.*
We denote by $\rightsquigarrow_{\mathcal{R}}$ the narrowing relation induced by $\mathcal{R}$, and we denote by narrowing derivation any derivation of the form $t_0 \rightsquigarrow_{\mathcal{R}} t_1 \rightsquigarrow_{\mathcal{R}} \ldots$ where t_0 is an arbitrary term.

We define now the *basic narrowing* which, intially introduced in [123], is a refinement of the narrowing.

Definition 12 *Let t be a term and let $NV.Pos(t)$ be the set of non-variable positions in t, $NV.Pos(t) = \{p \in Pos(t)$ such that $t_{|p} \notin \mathcal{X}\}$. We define by induction what it means for a narrowing derivation*

$$t = t_0 \rightsquigarrow_{\mathcal{R}} t_1 \rightsquigarrow_{\mathcal{R}} \ldots \rightsquigarrow_{\mathcal{R}} t_i$$

to be based on P_0 with $P_0 \subseteq NV.Pos(t)$, and we construct a set of positions $P_i \in Pos(t_i)$, $0 \leq i \leq n$, as follows:

- *the empty narrowing derivation is based on P_0,*
- *if the narrowing derivation above is based on P_0, then the derivation obtained from it by adding one step $t_i \rightsquigarrow_{\mathcal{R}} t_{i+1}$ is based on P_0 if and only $p_i \in P_i$, with p_i is the position in t_i on which the rule $l_i \rightarrow r_i \in \mathcal{R}$ is applied to obtain t_{i+1} from t_i, and in this case we take*

$$P_{i+1} = (P_i \setminus \{q \in P_i \text{ such that } p_i \leq q\}) \cup \{p_i \cdot q \text{ such that } q \in NV.Pos(r_i)\}$$

Definition 13 *(Basic narrowing) A narrowing derivation started from a term t*

$$t = t_0 \leadsto_{\mathcal{R}} t_1 \leadsto_{\mathcal{R}} \ldots \leadsto_{\mathcal{R}} t_n$$

is said to be a basic narrowing derivation if and only if it is based on $NV.Pos(t)$.

From now on, we denote by $\leadsto_{\mathcal{R}}$ the narrowing relation induced by $\mathcal{R}$, and by $\leadsto^b_{\mathcal{R}}$ the basic narrowing relation induced by $\mathcal{R}$.

J.M. Hullot [123] studied the correspondence between narrowing and rewriting, and obtained many results in that field. We present below some of these results:

Theorem 1 *(J.M. Hullot results) Let $\mathcal{H}$ be an equational theory on $\mathcal{T}(\mathcal{F}, \mathcal{X})$ generated by the convergent rewrite system $\mathcal{R}$.*

1. *Let t be an arbitrary term, and let σ be a substitution in normal form. Consider any $\rightarrow_{\mathcal{R}}$-derivation issuing from $t\sigma$:*

   $$t\sigma = t'_0 \rightarrow_{\mathcal{R}} t'_1 \rightarrow_{\mathcal{R}} \ldots \rightarrow_{\mathcal{R}} t'_n,$$

   there exists an associated $\mathcal{R}$-basic narrowing derivation issuing from t:

   $$t = t_0 \leadsto^b_{\mathcal{R}} t_1 \leadsto^b_{\mathcal{R}} \ldots \leadsto^b_{\mathcal{R}} t_n,$$

   and there exists a substitution μ in normal form such that $t_n\mu = t'_n$, and $\sigma = \theta_0\theta_1 \ldots \theta_{n-1}\mu$ with θ_i is the substitution used with the rule $l_i \rightarrow r_i \in \mathcal{R}$ in the step $t_i \leadsto^b_{\mathcal{R}} t_{i+1}$.

2. *Let s and t be two terms, M be $g(s,t)$ where g is a new function symbol ($g \notin \mathcal{F}$). Let Σ be the set of all substitutions σ such that $\sigma \in \Sigma$ if and only if there exists a $\mathcal{R}$-basic narrowing derivation*

   $$M = g(s,t) = M_0 \leadsto^b_{\mathcal{R}} M_1 = g(s_1,t_1) \leadsto^b_{\mathcal{R}} \ldots \leadsto^b_{\mathcal{R}} M_n = g(s_n,t_n),$$

   such that s_n and t_n are unifiable modulo $\emptyset$ theory, θ is a substitution in normal form, and $\sigma = \mu\theta$ where μ is the most general unifier of s_n and t_n. Then Σ is a complete set of $\mathcal{H}$-unifiers of s and t.

3. *If $\mathcal{R}$ is a convergent rewrite system such that for every rule $l \rightarrow r \in \mathcal{R}$, every $\mathcal{R}$-basic narrowing derivation starting from r terminates, then any $\mathcal{R}$-narrowing derivation starting from any term terminates.*

4. *If the convergent rewrite system $\mathcal{R}$ satisfies the hypothesis given in the above point (point (3)) then the construction given in the second point (point (2)) leads to a sound, complete and finite $\mathcal{H}$-unification algorithm.*

Remark. It is easy to see that the first point (point (1)) of Theorem 1 implies that: for any term t and for any substitution σ in normal form, there exists a term t' and a substitution σ' in normal form such that $t \overset{*}{\leadsto}^{b}_{\mathcal{R}} t'$ and $t'\sigma' = (t\sigma)\downarrow$.

Lemma 4 *Let $\mathcal{H}$ be an equational theory generated by a convergent rewrite system $\mathcal{R}$ such that the right hand side of every rule in $\mathcal{R}$ is not $\mathcal{R}$-narrowable. Let t be a term and D be a $\mathcal{R}$-basic narrowing derivation starting from t. Then, the length of D in bounded by $\|t\|_{dag}$.*

PROOF.

Let t be a term and D be a $\mathcal{R}$-basic narrowing derivation starting from t,

$$D : t = t_0 \leadsto^{b}_{\mathcal{R}} t_1 \leadsto^{b}_{\mathcal{R}} \ldots \leadsto^{b}_{\mathcal{R}} t_n$$

$\mathcal{R}$ is convergent and any basic narrowing derivation starting from the right members of the rules of $\mathcal{R}$ terminates (in fact, all right members of the rules of $\mathcal{R}$ are not $\mathcal{R}$-basic narrowable), then every $\mathcal{R}$-narrowing derivation starting from any term terminates (Theorem 1), and hence D terminates. We prove next that $\|D\| \leq \|t\|_{dag}$. Let Q_i be the number of distinct subterms of t_i where we can apply the basic narrowing. We note that if the basic narrowing can be applied on a term s at a position p and if there exists another subterm of s at position q such that $t_{|p} = t_{|q}$, we apply the basic narrowing at the positions p and q at the same time. Since all right members of $\mathcal{R}$ rules are not narrowable, and by definition of narrowing (Definition 11), we deduce that $Q_{i+1} < Q_i$, and hence, $\|D\| \leq Q_0$. By definition, we have $Q_0 \leq \|t\|_{dag}$, which implies that $\|D\| \leq \|t\|_{dag}$, and hence the length of any $\mathcal{R}$-basic narrowing derivation starting from any term t is bounded by $\|t\|_{dag}$. ■

Lemma 5 *Let $\mathcal{H}$ be an equational theory generated by a convergent rewrite system $\mathcal{R}$ such that the right hand side of every rule in $\mathcal{R}$ is not $\mathcal{R}$-narrowable. For any term t, and for any variant substitution θ of t, we can guess in $NPtime$ another variant substitution θ' of t such that θ' is more general modulo $\mathcal{H}$ than θ.*

PROOF.

Let $\mathcal{H}$ be an equational theory generated by a convergent rewrite system $\mathcal{R}$ such that the right hand side of every rule in $\mathcal{R}$ is not $\mathcal{R}$-narrowable. This implies that every $\mathcal{R}$-narrowing derivation starting from any term terminates 1, and hence $\mathcal{H}$ has the finite variant property [86]. Let t be a term, the finite variant property implies that we can construct a finite set of variant substitutions $\Sigma(t) = \{\sigma_1, \ldots, \sigma_n\}$ of t. We remark, by definition of finite variant property (Definition 46), that substitutions in $\Sigma(t)$ are in normal form. Let $\sigma \in \Sigma(t)$, and let the derivation $t\sigma = t'_0 \rightarrow_{\mathcal{R}} t'_1 \rightarrow_{\mathcal{R}} \ldots \rightarrow_{\mathcal{R}} t'_n$ such that t'_n is in normal form. Theorem 1 implies that there exists a $\mathcal{R}$-basic narrowing

derivation $D : \ t = t_0 \leadsto_{\mathcal{R}} t_1 \leadsto_{\mathcal{R}} \ldots \leadsto_{\mathcal{R}} t_n$ such that at each step $t_i \leadsto_{\mathcal{R}} t_{i+1}$, the rule $l_i \rightarrow r_i \in \mathcal{R}$ is used with the substitution θ_i; furthermore, there exists a substitution μ in normal form such that $\sigma = \theta_0\theta_1 \ldots \theta_{n-1}\mu$ and $t'_n = t_n\mu$. This implies that $(t\sigma)\downarrow = t'_n = t_n\mu = (t\theta_1\theta_2 \ldots \theta_n)\downarrow\mu$, and hence, $\theta_1 \ldots \theta_n$ is a variant substitution of t, and $\theta_1 \ldots \theta_n \lesssim_{\mathcal{H}} \sigma$. Assume that we always explore the right $\mathcal{R}$-basic narrowing derivation starting from t. By lemma 4, we have that any $\mathcal{R}$-basic narrowing derivation starting from t is bounded by $\|t\|_{dag}$, and hence, we conclude the proof. ■

2.1.10 Two intruder deduction systems

As mentioned in the introduction of this chapter, we work in the so called symbolic model. Cryptographic primitives can be seen as functions from messages to messages and messages themelves are obtained by applying operations to other messages. It is then natural to represent cryptographic primitives by function symbols $\mathcal{F}$, and messages by terms in $\mathcal{T}(\mathcal{F}, \mathcal{X})$. We specify how to apply functions symbols in order to get messages by deduction rules.

Definition 14 *(Deduction rule) Let $\mathcal{F}$ be a signature and $\mathcal{H}$ be an equational theory. A deduction rule over $\mathcal{F}$ is a tuple of terms $(l_1, \ldots, l_n, r)$, denoted by $l_1 \ldots, l_n \rightarrow r$, with $l_1, \ldots, l_n, r \in \mathcal{T}(\mathcal{F}, \mathcal{X})$. For the aim of simplicity, we denote the rule $l_1, \ldots, l_n \rightarrow r$ by $\tilde{l} \rightarrow r$ where $\tilde{l}$ is the set of terms $l_1, \ldots, l_n$.*

For example, consider the function symbol enc representing the encryption primitive, the deduction rule $x, y \rightarrow enc(x, y)$ specify the application of that function symbol.

We consider an intruder who, in addition to follow the *Dolev-Yao model* [107], uses the algebraic properties of cryptographic primitives in order to break security properties of cryptographic protocols. In the Dolev-Yao model [107], the intruder is the network, he can intercept, block, and/or redirect all messages sent by honest agents. He can also deduce new messages and take part in the protocol as an honest agent or under the identity of an honest agent (if he can masquerade the identity of that honest agent). In order to construct messages, the intruder follows some specific rules that he applies on his knowledge. The application of a rule on the intruder knowledge outputs a new term that will be added to the intruder knowledge. We call *intruder deduction system* the set of deduction rules representing the capacities of the intruder.

Before giving the two models of intruder deduction system, we give the intuition behind these models.

Intuition We have two types of cryptographic primitives, the *constructor primitives* and the *destructor primitives*. A primitive is said to be *constructor* if its application on a set of terms outputs a *new term*, that is a term for which we need

a dedicated function symbol to represent it. For example enc, representing the encryption, is a constructor primitive. A primitive is said to be *destructor* if its is not a constructor, that is if its application on a set of terms E outputs a term which can be represented without a dedicated function symbol, for example, the primitive dec, representing the decryption, is a destructor.

We have two ways to represent destructors, either they are explicit in the signature or they are implicit, and hence, we have two models for intruder deduction systems. Below, we present respectively the intruder deduction systems in the case of explicit and implicit destructors.

Intruder deduction system with explicit destructors

Assume the signature $\mathcal{F}$ is a disjoint union of two sets of function symbols $\mathcal{F}_{pub}$ and $\mathcal{F}_{pri}$, $\mathcal{F} = \mathcal{F}_{pub} \cup \mathcal{F}_{pri}$. A function symbol is said to be *public* if it can be performed by any parties, and *private* otherwise. As example, $\mathcal{F}_{pub} = \{enc, dec, Pk\}$, and $\mathcal{F}_{pri} = \{Sk\}$, where enc, dec, Pk and Sk represents respectively the encryption, decryption, public key generator function, and secret key generator function.

The definitions we give below were introduced in [73].

Definition 15 *(Intruder deduction rule with explicit destructors) An intruder deduction rule is a tuple of terms $(x_1, \ldots, x_n, f(x_1, \ldots, x_n))$, denoted by $x_1, \ldots, x_n \rightarrow f(x_1, \ldots, x_n)$, where x_i is a variable for every i, and $f \in \mathcal{F}_{pub}$ with arity n.*

Definition 16 *(Intruder deduction system with explicit destructors) An intruder deduction system $\mathcal{I}$, also called an intruder system, is a tuple $\mathcal{I} = \langle \mathcal{F}, \mathcal{T}_{\mathcal{I}}, \mathcal{H} \rangle$ where $\mathcal{F}$ is a signature, $\mathcal{T}_{\mathcal{I}} = \{f(x_1, \ldots, x_n)$ such that x_i is variable for every i, and $f \in \mathcal{F}_{pub}$ with arity n $\}$, and $\mathcal{H}$ is an equational theory over $\mathcal{T}(\mathcal{F}, \mathcal{X})$. To each $f(x_1, \ldots, x_n) \in \mathcal{T}_{\mathcal{I}}$, we associate a deduction rule, also called an intruder deduction rule, $x_1, \ldots, x_n \rightarrow f(x_1, \ldots, x_n)$. The set of rules $\mathcal{L}_{\mathcal{I}}$ is defined as the union of $x_1, \ldots, x_n \rightarrow f(x_1, \ldots, x_n)$ for all $f(x_1, \ldots, x_n) \in \mathcal{T}_{\mathcal{I}}$.*

Example 5 *We assume $\mathcal{F} = \{enc^s, dec^s\}$, and $\mathcal{F} = \mathcal{F}_{pub}$. The intruder deduction system $\mathcal{I}$ is given by the tuple $\langle \mathcal{F}, \mathcal{T}_{\mathcal{I}}, \mathcal{H} \rangle$ where:*

- $\mathcal{T}_{\mathcal{I}} = \{enc^s(x, y), dec^s(x, y)\}$,
- *and* $\mathcal{H} = \{dec(enc(x, y), y) = x\}$.

We conclude that $\mathcal{L}_{\mathcal{I}} = \begin{cases} x, y \rightarrow enc^s(x, y) \\ x, y \rightarrow dec^s(x, y) \end{cases}$

Intruder deduction system with implicit destructors

The definitions we give below were introduced in [72].

Definition 17 *(Intruder deduction rule with implicit destructors) An intruder deduction rule is a tuple of terms in $\mathcal{T}(\mathcal{F}, \mathcal{X})$ $(t_1, \ldots, t_n, t)$, denoted by $t_1, \ldots, t_n \rightarrow t$.*

Definition 18 *(Intruder deduction system with implicit destructors) An intruder deduction system $\mathcal{I}$, also called an intruder system, is a tuple $\mathcal{I} = \langle \mathcal{F}, \mathcal{L}_\mathcal{I}, \mathcal{H} \rangle$ where $\mathcal{F}$ is a signature, $\mathcal{L}_\mathcal{I}$ is a set of deduction rules of the form $t_1, \ldots, t_n \rightarrow t$, and $\mathcal{H}$ is an equational theory over $\mathcal{T}(\mathcal{F}, \mathcal{X})$.*

Example 6 *Assume $\mathcal{F} = \{enc^s\}$, and $\mathcal{F} = \mathcal{F}_{pub}$. The intruder deduction system $\mathcal{I} = \langle \mathcal{F}, \mathcal{L}_\mathcal{I}, \mathcal{H} \rangle$ where:*

- $\mathcal{L}_\mathcal{I} = \left\{ \begin{array}{l} x, y \rightarrow enc^s(x, y) \\ enc^s(x, y), y \rightarrow x \end{array} \right.$
- *and $\mathcal{H} = \emptyset$.*

Intruder derivations

Given an intruder system $\mathcal{I}$ (with explicit or implicit destructors) such that $\mathcal{L}_\mathcal{I}$ is the corresponding set of deduction rules, given two finite sets of terms E and F, we have $E \rightarrow_\mathcal{I} F$ if and only if there is an intruder deduction rule $\tilde{l} \rightarrow r \in \mathcal{L}_\mathcal{I}$ (where $\tilde{l}$ is a set of terms and r is a term), and a substitution σ, such that $\tilde{l}\sigma =_\mathcal{H} \tilde{l'}$, $r\sigma =_\mathcal{H} r'$, $\tilde{l'} \subseteq E$ and $F = E \cup \{r'\}$. We denote by $\rightarrow_\mathcal{I}^*$ the transitive closure of $\rightarrow_\mathcal{I}$. It is easy to see that for sets of terms E, E', F and F' such that $E =_\mathcal{H} E'$ and $F =_\mathcal{H} F'$, we have $E \rightarrow_\mathcal{I} F$ if and only if $E' \rightarrow_\mathcal{I} F'$. We simply denote by $\rightarrow$ the relation $\rightarrow_\mathcal{I}$ when there is no ambiguity about $\mathcal{I}$.

An *$\mathcal{I}$-derivation* D of length n, $n \geq 0$, is a sequence of the form $E_0 \rightarrow_\mathcal{I} E_0, t_1 \rightarrow_\mathcal{I} \cdots \rightarrow_\mathcal{I} E_n$ with finite sets of terms $E_0, \ldots, E_n$, and terms $t_1, \ldots, t_n$, such that $E_i = E_{i-1} \cup \{t_i\}$ for every $i \in \{1, \ldots, n\}$. The term t_n is called the *goal* of the derivation. Given a set of terms E, we define $\bar{E}^\mathcal{I}$ to be equal to the set of terms that can be derived from E with respect to $\mathcal{I}$, the set $\bar{E}^\mathcal{I} = \{t$ such that $\exists F$ with $E \rightarrow_\mathcal{I}^* F$, and $t \in F\}$. If E and t are respectively a set of terms in normal form and a term in normal form, it is easy to see that if $t \in \bar{E}^\mathcal{I}$ then there exists a $\mathcal{I}$-derivation starting from E of goal t, $D : E \rightarrow_\mathcal{I} E, t_1 \rightarrow_\mathcal{I} E, t_1, t_2 \rightarrow_\mathcal{I} \ldots \rightarrow_\mathcal{I} E, t_1, \ldots, t_n, t$, where $t_1, \ldots, t_n$ are in normal form, that is in each step $E, t_1, \ldots, t_i \rightarrow_\mathcal{I} E, t_1, \ldots, t_{i+1}$ where $\tilde{l} \rightarrow r$ is the applied rule, we have that $(\tilde{l}\sigma) \downarrow \subseteq E, t_1, \ldots, t_i$ and $t_{i+1} = (r\sigma) \downarrow$. From now on, we consider that in each derivation starting from E of goal t, $E \rightarrow_\mathcal{I} E, t_1 \rightarrow_\mathcal{I} E, t_1, \ldots, t_n, t$, with E and t are respectively a set of terms in normal form and a term in normal form, every term t_i $(1 \leq i \leq n)$ added in each step in the derivation is in normal

form. If there is no ambiguity on the intruder deduction system $\mathcal{I}$ we write $\bar{E}$ instead of $\bar{E}^{\mathcal{I}}$.

Example 7 *(The Dolev-Yao deduction system with implicit destructors) We present here the Dolev-Yao deduction system with implicit destructors.*

Let $< -, - >$ (concatenation), enc^s (symmetric encryption), enc^p (public encryption) and $^{-1}$ (the inverse key) be the cryptographic primitives. The rules $L_{<-,->}$ are defined as follows:

$$x, y \rightarrow < x, y > \tag{2.1}$$
$$< x, y > \rightarrow x \tag{2.2}$$
$$< x, y > \rightarrow y \tag{2.3}$$

The rules L_{enc^s} are defined as follows:

$$x, y \rightarrow enc^s(x, y) \tag{2.4}$$
$$enc^s(x, y), y \rightarrow x \tag{2.5}$$

The rules L_{enc^p} are defined as follows:

$$x, y \rightarrow enc^p(x, y) \tag{2.6}$$
$$enc^p(x, y), y^{-1} \rightarrow x \tag{2.7}$$
$$enc^p(x, y^{-1}), y \rightarrow x \tag{2.8}$$

The Dolev-Yao deduction system with implicit destructors is given by

$$\mathcal{I}^i_{DY} = \langle \mathcal{F}_{DY}, L_{DY}, \emptyset \rangle$$

where

- $\mathcal{F}_{DY} = \{< -, - >, enc^s, enc^p, ^{-1}\}$,
- $L_{DY} = L_{<-,->} \cup L_{enc^s} \cup L_{enc^p}$.

Example 8 *(The Dolev-Yao deduction system with explicit destructors) We present here the Dolev-Yao deduction system with explicit destructors.*

Let $< -, - >$ (concatenation), enc^s (symmetric encryption), dec^s (symmetric decryption), enc^p (public encryption), dec^p (public decryption), and $^{-1}$ (the inverse key) be cryptographic primitives. The rules $L_{<-,->}$ are defined as follows:

$$x, y \rightarrow < x, y > \tag{2.9}$$
$$x \rightarrow \pi_1(x) \tag{2.10}$$
$$x \rightarrow \pi_2(x) \tag{2.11}$$

and the associated equational theory is defined by:

$$\mathcal{H}_{<,-,->} : \begin{cases} \pi_1(< x, y >) = x \\ \pi_2(< x, y >) = y \end{cases}$$

The rules L_{enc^s} are defined as follows:

$$x, y \rightarrow enc^s(x, y) \tag{2.12}$$
$$x, y \rightarrow dec^s(x, y) \tag{2.13}$$

and the associated equational theory is defined by:

$$\mathcal{H}_{enc^s} = \{dec^s(enc^s(x, y), y) = x\}$$

The rules L_{enc^p} are defined as follows:

$$x, y \rightarrow enc^p(x, y) \tag{2.14}$$
$$x, y \rightarrow dec^p(x, y) \tag{2.15}$$

and the associated equational theory is defined by:

$$\mathcal{H}_{dec^p} : \begin{cases} dec^p(enc^p(x, y), y^{-1}) = x \\ dec^p(enc^p(x, y^{-1}), y) = x \end{cases}$$

The Dolev-Yao deduction system with explicit destructors is given by

$$\mathcal{I}^e_{DY} = \langle \mathcal{F}_{DY}, L_{DY}, \mathcal{H}_{DY} \rangle$$

where

- $\mathcal{F}_{DY} = \{< -, - >, enc^s, dec^s, enc^p, dec^p\}$,
- $L_{DY} = L_{<-,->} \cup L_{enc^s} \cup L_{enc^p}$,
- $\mathcal{H}_{DY} = \mathcal{H}_{<-,->} \cup \mathcal{H}_{enc^s} \cup \mathcal{H}_{enc^p}$.

From now on, we consider only intruder deduction systems with explicit destructors, and for simplicity, we will use the notation "intruder deduction systems" to mean "intruder deduction systems with explicit destructors". This intruder deduction system will be denoted by $\mathcal{I} = \langle \mathcal{F}, \mathcal{T}_\mathcal{I}, \mathcal{H} \rangle$ and $\mathcal{L}_\mathcal{I}$ denotes the set of deduction rules associated to $\mathcal{I}$.

2.1.11 Variant of the intruder deduction system

Let $\mathcal{I}$ be an intruder deduction system, $\mathcal{I} = \langle \mathcal{F}, \mathcal{T}_{\mathcal{I}}, \mathcal{H} \rangle$ and let $\mathcal{L}_{\mathcal{I}}$ be the set of intruder deduction rules associated to $\mathcal{I}$. We recall that rules in $\mathcal{L}_{\mathcal{I}}$ are of the form $x_1, \ldots, x_n \rightarrow f(x_1, \ldots, x_n)$ and f is a public function symbol in $\mathcal{F}$ with arity n.

Definition 19 *(Variant of the intruder deduction system) Let $\mathcal{H}$ be an equational theory having the finite variant property, and let $\mathcal{I} = \langle \mathcal{F}, \mathcal{T}_{\mathcal{I}}, \mathcal{H} \rangle$ be an intruder deduction system. The variant intruder deduction system $\mathcal{I}' = \langle \mathcal{F}, \mathcal{L}'_{\mathcal{I}}, \emptyset \rangle$ of the intruder deduction system $\mathcal{I}$ is defined as follows:*

$$\mathcal{L}'_{\mathcal{I}} = \bigcup_{\substack{x_1, \ldots, x_n \rightarrow f(x_1, \ldots, x_n) \in \mathcal{L}_{\mathcal{I}} \\ \theta \text{ variant substitution of } f(x_1, \ldots, x_n)}} x_1\theta, \ldots, x_n\theta \rightarrow (f(x_1, \ldots, x_n)\theta)\downarrow$$

where $\mathcal{L}_{\mathcal{I}}$ is the set of intruder deduction rules associated to $\mathcal{I}$.

Lemma 6 *Let $<$ be an arbitrary complete simplification ordering over $\mathcal{T}(\mathcal{F}, \mathcal{X})$, that is $<$ is well-founded, monotone, stable, total over ground terms $\mathcal{T}(\mathcal{F})$, and $t < s[t]$ for any terms s, t in $\mathcal{T}(\mathcal{F}, \mathcal{X})$, and let $t_1, t_2 \in \mathcal{T}(\mathcal{F}, \mathcal{X})$. If $t_1 \leq t_2$ then:*

1. $Var(t_1) \subseteq Var(t_2)$
2. $t_2 \notin SSub(t_1)$
3. *If $t_2 \in \mathcal{X}$ then either $t_1 = t_2$ or t_1 is the minimal ground term in $\mathcal{T}(\mathcal{F})$ for $<$.*

PROOF.

1. Let t_1 and t_2 be two terms and $t_1 \leq t_2$. If $t_1 = t_2$ then we have obviously $Var(t_1) = Var(t_2)$. Suppose $t_1 \neq t_2$ this implies that $t_1 < t_2$ and let us prove that $Var(t_1) \subseteq Var(t_2)$. By contradiction, suppose that $Var(t_1) \not\subseteq Var(t_2)$ and let $x \in Var(t_1) \setminus Var(t_2)$. By definition of $<$, we have $t_1\sigma < t_2\sigma$ for all substitutions σ. Let σ be a substitution such that $Supp(\sigma) = \{x\}$ and $\sigma(x) = t_2$. This implies that $t_2\sigma = t_2$ and $t_2 \in Sub(t_1\sigma)$ and hence $t_2 \leq t_1\sigma$ which contradicts $t_1 < t_2$.
2. If $t_2 \in SSub(t_1)$ this implies that $t_1 \neq t_2$ and $t_2 < t_1$ which contradicts $t_1 \leq t_2$.
3. Assume $t_2 = x$. $t_1 \leq t_2$ implies $Var(t_1) \subseteq \{x\}$ and point (2) implies that $x \notin SSub(t_1)$. Hence, either $t_1 = x$ or t_1 is a ground term in $\mathcal{T}(\mathcal{F})$. Let t_1 be a no minimal ground term in $\mathcal{T}(\mathcal{F})$, and let s be a ground term in $\mathcal{T}(\mathcal{F})$

with $s < t_1$. $t_1 < x$ implies $t_1 < x\sigma$ for all σ and then for $\sigma = \{x \mapsto s\}$, we have $t_1 < s$ which contradicts $s < t_1$. We conclude that either $t_1 = t_2$ or t_1 is the minimal ground term in $\mathcal{T}(\mathcal{F})$ for $<$ which exists due to the well-foundness of $<$ and which is unique due to the fact that $<$ is total over ground terms.

■

Lemma 7 *Let $\mathcal{I} = \langle \mathcal{F}, \mathcal{T}_\mathcal{I}, \mathcal{H} \rangle$ be an intruder deduction system, and let $\mathcal{I}' = \langle \mathcal{F}, \mathcal{L}', \emptyset \rangle$ be the variant intruder deduction system of $\mathcal{I}$. If $t_1, \ldots, t_n \rightarrow t \in \mathcal{L}'$, then $Var(t) \subseteq Var(t_1, \ldots, t_n)$.*

PROOF.

Let $t_1, \ldots, t_n \rightarrow t \in \mathcal{L}'$, we have two cases:

1. If $t_1, \ldots, t_n \rightarrow t \in \mathcal{L}$ then, by definition of $\mathcal{L}$, $t_1, \ldots, t_n \rightarrow t = x_1, \ldots, x_n \rightarrow f(x_1, \ldots, x_n)$ where f is a public function symbol in $\mathcal{F}$ and hence, $Var(t) \subseteq Var(t_1, \ldots, t_n)$.

2. If $t_1, \ldots, t_n \rightarrow t \notin \mathcal{L}$, then, by definition of $\mathcal{L}'$, there exists a rule $x_1, \ldots, x_n \rightarrow f(x_1, \ldots, x_n) \in \mathcal{L}$, a variant substitution θ of $f(x_1, \ldots, x_n)$ such that $t_i = x_i\theta$ for every i, and $(f(x_1, \ldots, x_n)\theta)\downarrow = t$. We have that $Var(f(x_1, \ldots, x_n)\theta) = \bigcup_{i=1}^{n} Var(x_i\theta)$, and $(f(x_1, \ldots, x_n)\theta)\downarrow \leq f(x_1, \ldots, x_n)\theta$. Lemma 6 implies that $Var((f(x_1, \ldots, x_n)\theta)\downarrow) \subseteq Var(f(x_1, \ldots, x_n)\theta)$ and hence, $Var((f(x_1, \ldots, x_n)\theta)\downarrow) \subseteq \bigcup_{i=1}^{n} Var(x_i\theta)$ which concludes the lemma.

■

We prove in what follows (Lemma 8) that when considering only deductions on ground terms in normal form and yielding ground terms in normal form, it is sufficient to consider derivations modulo the empty theory.

Lemma 8 *Let E and F be two sets of ground terms in normal form we have: $E \rightarrow_\mathcal{I} F$ if and only if $E \rightarrow_{\mathcal{I}'} F$.*

PROOF.

Let E and F be two sets of ground terms in normal form and assume there is a rule $x_1, \ldots, x_n \rightarrow f(x_1, \ldots, x_n) \in \mathcal{L}$ such that $E \rightarrow_{x_1,\ldots,x_n \rightarrow f(x_1,\ldots,x_n)} F$. By definition there exists a ground substitution σ in normal form such that $(x_1, \ldots, x_n)\sigma \subseteq E$ and $F = E \cup \{(f(x_1, \ldots, x_n)\sigma)\downarrow\}$. Due to the finite variant property, there exists a variant substitution θ of $f(x_1, \ldots, x_n)$ and a ground normal substitution σ' such that $(f(x_1, \ldots, x_n)\sigma)\downarrow = (f(x_1, \ldots, x_n)\theta)\downarrow\sigma'$ and $\sigma = \theta\sigma'$. The rule $x_1\theta, \ldots x_n\theta \rightarrow (f(x_1, \ldots, x_n)\theta)\downarrow$ was added to $\mathcal{L}'$ by definition of $\mathcal{I}'$ (Definition 19). This implies that $E \rightarrow_{\mathcal{I}'} F$. To prove the converse, notice that if $(x_1, \ldots, x_n)\theta \rightarrow (f(x_1, \ldots, x_n)\theta)\downarrow$ can be applied with the normal ground substitution σ' on E, then the rule $x_1, \ldots, x_n \rightarrow f(x_1, \ldots, x_n)$ can be applied with the ground substitution $\sigma = (\theta\sigma')\downarrow$ on E. ■

2.1.12 Constraint systems

In [156], the authors introduced the notions of *"deduction constraint"* and *"$\mathcal{I}$-constraint systems"*. They defined a *deduction constraint* to be an expression of the form $E \trianglerighteq t$ where E is a set of terms and t is a term, and they defined an *$\mathcal{I}$-constraint system* $\mathcal{C}$ as follows: $\mathcal{C} = (E_1 \trianglerighteq t_1, \ldots, E_n \trianglerighteq t_n)$ where $E_i \subseteq E_{i+1}$, and $Var(E_i) \subseteq Var(\{t_1, \ldots, t_{i-1}\})$. They defined also a solution of $\mathcal{C}$ as follows: a substitution σ is a solution of $\mathcal{C}$ if $t_i\sigma \in \overline{E_i\sigma}$ for every i. This notion of *$\mathcal{I}$-constraint system* has been defined with the $\emptyset$ equational theory in mind.

Unfortunately, such definitions of $\mathcal{I}$-constraint systems are not adequate in presence of non empty equational theory. For instance, let us consider the equational theory $\mathcal{H} = \{f(x, x) = a\}$ which is generated by the convergent rewrite system $\mathcal{R} = \{f(x, x) \to a\}$, and let us consider the $\mathcal{I}$-constraint system $\mathcal{C} = (\{a, b\} \trianglerighteq f(x, y), \{a, b, x\} \trianglerighteq b)$. This constraint system follows the definition of constraint system given above, and the substitution $\sigma = \{x \mapsto y\}$ is a solution of $\mathcal{C}$ following the definition above. When we apply this substitution σ to $\mathcal{C}$ then normalise, we obtain the following system $\mathcal{C}' = (\{a, b\} \trianglerighteq a, \{a, b, y\} \trianglerighteq b)$. It is easy to see that $\mathcal{C}'$ does not satisfy the definition of constraint systems given above.

In order to avoid such problem, in [73], the authors introduced another definition of constraint systems. This definition, given below, is adequate with the non empty equational theories, and it is the definition adapted in this document.

Definition 20 *($\mathcal{I}$-Constraint systems) Let $\mathcal{I}$ be an intruder deduction system. An $\mathcal{I}$-*constraint system *$\mathcal{C}$ is denoted $(E_1 \vdash v_1, \ldots, E_n \vdash v_n, \mathcal{U})$ and is defined by a finite set of expressions $E_i \vdash v_i$, called deduction constraints, with:*

- *$v_i \in \mathcal{X}$ for $i \in \{1, \ldots, n\}$,*
- *$E_1 \subseteq \mathcal{T}(\mathcal{F})$, and $E_i \subseteq \mathcal{T}(\mathcal{F}, \mathcal{X})$ for $i \in \{2, \ldots, n\}$,*
- *$E_i \subseteq E_{i+1}$ for $i \in \{1, \ldots, n-1\}$,*
- *$Var(E_i) \subseteq \{v_1, \ldots, v_{i-1}\}$ for $i \in \{2, \ldots, n\}$,*
- *and a $\mathcal{H}$-unification system $\mathcal{U}$.*

An $\mathcal{I}$-Constraint system $\mathcal{C}$ is satisfied by a substitution σ, and we write $\sigma \models_{\mathcal{I}} \mathcal{C}$, if for all $i \in \{1, \ldots, n\}$ we have $v_i\sigma \in \overline{E_i\sigma}^{\mathcal{I}}$ and if $\sigma \models_{\mathcal{H}} \mathcal{U}$. We call such a substitution a solution of $\mathcal{C}$.

It is easy to see that if a substitution σ is a solution of a constraint system $\mathcal{C}$, the substitution $(\sigma)\downarrow$ is also a solution of $\mathcal{C}$.

Definition 21 *($\mathcal{I}$-ground Constraint systems) Let $\mathcal{I}$ be an intruder system. An $\mathcal{I}$-ground constraint system $\mathcal{C}$ is denoted $(E_1 \vdash v_1, \ldots, E_n \vdash v_n, \left\{v_1 \stackrel{?}{=}_{\mathcal{H}} t_1, \ldots, v_n \stackrel{?}{=}_{\mathcal{H}} t_n\right\})$ and is defined by a finite set of expressions $E_i \vdash v_i$, and expressions $v_i \stackrel{?}{=}_{\mathcal{H}} t_i$ with:*

- *$E_i \cup \{t_i\} \subseteq \mathcal{T}(\mathcal{F})$, $v_i \in \mathcal{X}$ for $i \in \{1, \ldots, n\}$, and*
- *$E_i \subseteq E_{i+1}$ for $i \in \{1, \ldots, n-1\}$.*

2.1.13 Modified $\mathcal{I}$-constraint systems

We introduce now a *modified $\mathcal{I}$-constraint systems*.

Definition 22 *(modified $\mathcal{I}$-Constraint systems) Let $\mathcal{I}$ be an intruder deduction system. A modified $\mathcal{I}$-*constraint system *$\mathcal{C}$ is denoted $(E_1 \rhd t_1, \ldots, E_n \rhd t_n)$ and is defined as follows:*

- *$t_i \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ for $i \in \{1, \ldots, n\}$,*
- *$E_1 \subseteq \mathcal{T}(\mathcal{F})$, and $E_i \subseteq \mathcal{T}(\mathcal{F}, \mathcal{X})$ for $i \in \{2, \ldots, n\}$,*
- *$E_i \subseteq E_{i+1}$ for $i \in \{1, \ldots, n-1\}$,*
- *$Var(E_i) \subseteq Var(\{t_1, \ldots, t_{i-1}\})$ for $i \in \{2, \ldots, n\}$.*

A modified $\mathcal{I}$-Constraint system $\mathcal{C}$ is satisfied by a substitution σ, and we write $\sigma \models_{\mathcal{I}} \mathcal{C}$, if for all $i \in \{1, \ldots, n\}$ we have $t_i\sigma \in \overline{E_i\sigma}^{\mathcal{I}}$ and $E_i\sigma, t_i\sigma$ are in normal form .

We remark that our definition of modified $\mathcal{I}$-constraint system is different than the definition of $\mathcal{I}$-constraint system introduced in [156] and showed in Section 2.1.12: while in [156] the authors considered each substitution σ with $t\sigma \in \overline{E\sigma}^{\mathcal{I}}$ is a solution of $E \trianglerighteq t$, we consider the fact that $t\sigma \in \overline{E\sigma}^{\mathcal{I}}$ is not sufficient to have σ solution of $E \rhd t$ and we add the condition that $E\sigma$ and $t\sigma$ must be in normal form.

Let us consider again the example given in Section 2.1.12, that is $\mathcal{C} = (\{a, b\} \trianglerighteq f(x, y), \{a, b, x\} \trianglerighteq b)$, and the equational theory is $\mathcal{H} = \{f(x, x) = a\}$. Following [156], the substitution $\sigma = \{x \mapsto y\}$ is a solution of $\mathcal{C}$, and we showed how its application is problematic. Definition 22 implies that σ is not a solution of $\mathcal{C} = (\{a, b\} \rhd f(x, y), \{a, b, x\} \rhd b)$.

Definition 23 *(modified $\mathcal{I}$-ground constraint systems) Let $\mathcal{I}$ be an intruder deduction system. A modified $\mathcal{I}$-ground constraint system $\mathcal{C}$ is denoted $(E_1 \rhd t_1, \ldots, E_n \rhd t_n)$, with:*

- *$E_i \cup \{t_i\} \subseteq \mathcal{T}(\mathcal{F})$ for $i \in \{1, \ldots, n\}$, and*

- $E_i \subseteq E_{i+1}$ *for* $i \in \{1, \ldots, n-1\}$.

In the rest of this document, we will make use of the modified $\mathcal{I}$-ground constraint systems (Definition 23) instead of the $\mathcal{I}$-ground constraint system (Definition 21), and for the aim of simplicity, we will abuse of the notation and call modified $\mathcal{I}$-ground constraint systems by $\mathcal{I}$-ground constraint systems.

An $\mathcal{I}$-ground constraint system $\mathcal{C} = (E_1 \rhd t_1, \ldots, E_n \rhd t_n)$ is *satisfied*, and we write $\models_{\mathcal{I}} \mathcal{C}$, if for all $i \in \{1, \ldots, n\}$ we have $t_i \in \overline{E_i}^{\mathcal{I}}$, and t_i, E_i in normal form.

Definition 24 *(solved form) A modified $\mathcal{I}$-constraint system $(E_1 \rhd t_1, \ldots, E_n \rhd t_n)$ is said to be in solved form if for all i, we have $t_i \in \mathcal{X}$.*

Lemma 9 *Let $\mathcal{C}$ be a modified $\mathcal{I}$-constraint system, $\mathcal{C} = (\mathcal{C}_\alpha, E \rhd t, \mathcal{C}_\beta)$ such that $\mathcal{C}_\alpha$ is in solved form and $t \notin \mathcal{X}$. Then, for all substitutions σ we have: $\sigma \models_{\mathcal{I}} \mathcal{C}$ if and only if $\sigma \models_{\mathcal{I}} (\mathcal{C}_\alpha, (E \setminus \mathcal{X}) \rhd t, \mathcal{C}_\beta)$.*

PROOF.

It suffices to prove that if $x \in E \cap \mathcal{X}$ and σ is a substitution such that $\sigma \models_{\mathcal{I}} \mathcal{C}$, then we have $\sigma \models_{\mathcal{I}} (\mathcal{C}_\alpha, (E \setminus \{x\}) \rhd t, \mathcal{C}_\beta)$. Given $x \in E$, by definition 20, there exists a set of terms $E' \subseteq E$ such that $E' \rhd x \in \mathcal{C}_\alpha$. Since $\sigma \models_{\mathcal{I}} \mathcal{C}$ we have $\sigma \models_{\mathcal{I}} E' \rhd x$, and by the fact that $E' \subseteq E \setminus \{x\}$ we have $\sigma \models_{\mathcal{I}} E \setminus \{x\} \rhd x$. Since we also have $\sigma \models_{\mathcal{I}} (E \rhd t)$ then, $\sigma \models_{\mathcal{I}} E \setminus \{x\} \rhd t$. The reciprocal is obvious since $E \setminus \mathcal{X} \subseteq E$. ■

Lemma 10 *Let $\mathcal{C} = (\mathcal{C}_\alpha, E \rhd x, \mathcal{C}_\beta)$ be a modified $\mathcal{I}$-constraint system such that $\mathcal{C}_\alpha$ is in solved form and $x \notin Var(\mathcal{C}_\beta)$ and let $\mathcal{C}' = (\mathcal{C}_\alpha, \mathcal{C}_\beta)$. We have:*

1. *If $\sigma \models \mathcal{C}$ then $\sigma \models \mathcal{C}'$.*
2. *If $\sigma' \models \mathcal{C}'$ then we can extend σ' to σ such that $\sigma \models \mathcal{C}$.*

PROOF.

1. Let $\mathcal{C} = (\mathcal{C}_\alpha, E \rhd x, \mathcal{C}_\beta)$ and let σ be a closed substitution such that $\sigma \models \mathcal{C}$. Since $x \notin Var(\mathcal{C}_\beta)$, $\mathcal{C}'$ is a constraint system. It is trivial that $\sigma \models \mathcal{C}'$.
2. Let σ' be a closed substitution such that $\sigma' \models \mathcal{C}'$. Since $Var(E) \subseteq Var(\mathcal{C}_\alpha)$, σ' is defined on $Var(\mathcal{C}_\alpha, E, \mathcal{C}_\beta)$. We have two cases:
   - If $x \notin Var(\mathcal{C}_\alpha)$ then $\sigma'(x)$ is not defined and $x \notin Var(E)$. We then extend σ' to σ as follows:

     $$\sigma(y) = \sigma'(y) \text{ for } y \in Supp(\sigma'), \sigma(x) \text{ is a closed term in E.}$$

     Since $x \notin Var(\mathcal{C}_\alpha, \mathcal{C}_\beta, E)$ and $x\sigma \in E\sigma$, we deduce that $\sigma \models \mathcal{C}$.
   - If $x \in Var(\mathcal{C}_\alpha)$ then there exists $E_x \rhd x \in \mathcal{C}_\alpha$. $\sigma' \models (\mathcal{C}_\alpha, \mathcal{C}_\beta)$ implies that $\sigma' \models E_x \rhd x$, and since $E_x \subseteq E$, we have $\sigma' \models E \rhd x$ and hence $\sigma' \models \mathcal{C}$.

■

2.1.14 Reachability problems

Definition 25 ($\mathcal{I}$-Reachability Problem) *Given an intruder deduction system $\mathcal{I}$, the $\mathcal{I}$-reachability problem is defined as follows:*

Input: *An $\mathcal{I}$-constraint system $\mathcal{C} = (E_1 \vdash v_1, \ldots, E_n \vdash v_n, \mathcal{U})$*
. ***Output:*** SAT *if and only if there exists a substitution σ such that $\sigma \models_{\mathcal{I}} \mathcal{C}$.*

Definition 26 ($\mathcal{I}$-Ground Reachability Problem) *Given an intruder deduction system $\mathcal{I}$, the $\mathcal{I}$-ground reachability problem is defined as follows:*

Input: *An $\mathcal{I}$-ground constraint system $\mathcal{C} = (E_1 \rhd t_1, \ldots, E_n \rhd t_n)$.*
Output: SAT *if and only if for all $1 \leq i \leq n$, $t_i \in \overline{E_i}^{\mathcal{I}}$.*

2.1.15 Variant of $\mathcal{I}$-constraint systems

Definition 27 *Let $\mathcal{C} = (E_1 \vdash v_1, \ldots, E_n \vdash v_n, \mathcal{U})$ be an $\mathcal{I}$-constraint system, and let $\mathcal{C}'$ be the $\mathcal{I}$-constraint system constructed from $\mathcal{C}$ as follows:*

$$\mathcal{C}' = (E_1 \vdash v_1, \ldots, E_n \vdash v_n, \mathcal{U}') \text{ where}$$
$$\mathcal{U}' = \mathcal{U} \cup \left\{ x_1^1 \stackrel{?}{=}_{\mathcal{H}} e_1^1, \ldots, x_m^1 \stackrel{?}{=}_{\mathcal{H}} e_m^1, \ldots, x_1^n \stackrel{?}{=}_{\mathcal{H}} e_1^n, \ldots, x_p^n \stackrel{?}{=}_{\mathcal{H}} e_p^n \right\}$$

such that for all i, j, x_j^i is a fresh variable $\notin \mathcal{X}$, and the set of terms $\left\{e_j^i\right\}_j$ represents the terms in E_i.

Since $\bigcup_{i,j} Var(e_j^i) \subseteq \{v_1, \ldots, v_n\}$ by definition of constraint systems (Definition 20), it is easy to see that any solution σ of $\mathcal{U}$ can be extended to a solution σ' of $\mathcal{U}'$ as follows:

- $\sigma'(x) = x\sigma$ for all $x \in Var(\mathcal{C})$,
- $\sigma'(x) = (t\sigma)\!\downarrow$ for every variable $x \in Var(\mathcal{C}') \setminus Var(\mathcal{C})$ such that $x \stackrel{?}{=}_{\mathcal{H}} t \in \mathcal{U}' \setminus \mathcal{U}$.

And, each solution σ' of $\mathcal{U}'$ is a solution of $\mathcal{U}$.

Definition 28 *(variant of $\mathcal{I}$-constraint system) Let $\mathcal{C} = (E_1 \vdash v_1, \ldots, E_n \vdash v_n, \mathcal{U})$ be a $\mathcal{I}$-constraint system, and let $\mathcal{C}' = (E_1 \vdash v_1, \ldots, E_n \vdash v_n, \mathcal{U}')$ be a $\mathcal{I}$-constraint system constructed from $\mathcal{C}$ as given in Definition 27. Let σ be a substitution in $Sol_{Var}(\mathcal{U}')$. The constraint system $((E_1\sigma)\!\downarrow \rhd (v_1\sigma)\!\downarrow, \ldots, (E_n\sigma)\!\downarrow \rhd (v_n\sigma)\!\downarrow)$ is a variant $\mathcal{I}$-constraint system associated to $\mathcal{C}$.*

Lemma 11 *Let $\mathcal{C}$ be a $\mathcal{I}$-constraint system and let $\mathcal{C}'$ be a variant $\mathcal{I}$-constraint system associated to $\mathcal{C}$. $\mathcal{C}'$ is a modified $\mathcal{I}$-constraint system.*

PROOF.

By hypothesis, we have $\mathcal{C} = (E_1 \vdash v_1, \ldots, E_n \vdash v_n, \mathcal{U})$, and $\mathcal{C}' = ((E_1\sigma)\downarrow \rhd (v_1\sigma)\downarrow, \ldots, (E_n\sigma)\downarrow \rhd (v_n\sigma)\downarrow)$ with σ a substitution in normal form. By definition of $\mathcal{I}$-constraint systems, we have $E_i \subseteq E_{i+1}$, which implies that $E_i\sigma \subseteq E_{i+1}\sigma$, and hence $(E_i\sigma)\downarrow \subseteq (E_{i+1}\sigma)\downarrow$. Since $E_1 \subseteq \mathcal{T}(\mathcal{F})$, then $E_1 = E_1\sigma = (E_1\sigma)\downarrow$ and hence $(E_1\sigma)\downarrow \subseteq \mathcal{T}(\mathcal{F})$. Now, we prove that $Var((E_i\sigma)\downarrow) \subseteq Var(\{(v_1\sigma)\downarrow, \ldots, (v_{i-1}\sigma)\downarrow\})$. We have that $Var((E_i\sigma)\downarrow) \subseteq Var(E_i\sigma)$. We have also that $Var(E_i) \subseteq \{v_1, \ldots, v_{i-1}\}$ which implies that $Var(E_i\sigma) \subseteq Var(\{v_1\sigma, \ldots, v_{i-1}\sigma\})$. By definition of $\mathcal{C}'$, we have that σ is in normal form, and hence $(v_i\sigma)\downarrow = v_i\sigma$. This implies that $Var((E_i\sigma)\downarrow) \subseteq Var(\{(v_1\sigma)\downarrow, \ldots, (v_{i-1}\sigma)\downarrow\})$, which concludes the proof. ■

Lemma 12 *Let $\mathcal{H}$ be a finitary equational theory, and let $\mathcal{C} = (E_1 \vdash v_1, \ldots, E_n \vdash v_n, \mathcal{U})$ be an $\mathcal{I}$-constraint system, and let $\mathcal{C}' = (E_1 \vdash v_1, \ldots, E_n \vdash v_n, \mathcal{U}')$ constructed from $\mathcal{C}$ as in Definition 27. We have:*

1. *If there exists a substitution σ in normal form such that $\sigma \models_{\mathcal{I}} \mathcal{C}$ then there exists a substitution $\theta \in Sol_{Var}(\mathcal{U}')$, and a substitution τ in normal form such that $\tau \models_{\mathcal{I}'} ((E_1\theta)\downarrow \rhd (v_1\theta)\downarrow, \ldots, (E_n\theta)\downarrow \rhd (v_n\theta)\downarrow)$.*
2. *If there exists a substitution $\theta \in Sol_{Var}(\mathcal{U}')$, and a substitution τ in normal form such that $\tau \models_{\mathcal{I}'} ((E_1\theta)\downarrow \rhd (v_1\theta)\downarrow, \ldots, (E_n\theta)\downarrow \rhd (v_n\theta)\downarrow)$, then there exists a substitution σ in normal form such that $\sigma \models_{\mathcal{I}} \mathcal{C}$.*

PROOF.

Let $\mathcal{C} = (E_1 \vdash v_1, \ldots, E_n \vdash v_n, \mathcal{U})$ be an $\mathcal{I}$-constraint system, and let $\mathcal{C}' = (E_1 \vdash v_1, \ldots, E_n \vdash v_n, \mathcal{U}')$ constructed from $\mathcal{C}$ as given in Definition 27.

1. Let σ be a substitution in normal form such that $\sigma \models_{\mathcal{I}} \mathcal{C}$. σ can be extended, as given above, to σ' such that $\sigma' \models_{\mathcal{I}} \mathcal{C}'$, and hence, $\sigma' \models_{\mathcal{I}} E_i \vdash v_i$ for all i, and $\sigma' \models_{\mathcal{H}} \mathcal{U}'$. This implies that there exists a variant substitution of $T_{\mathcal{U}'}$, a substitution τ in normal form such that for any term $u \in \mathcal{C}'$, $(u\sigma)\downarrow = (u\theta)\downarrow\tau$. This implies that $\tau \models_{\emptyset} (\mathcal{U}'\theta)\downarrow$, and hence, there exists a most general $\emptyset$-unifier, μ of $(\mathcal{U}'\theta)\downarrow$, and a substitution τ' in normal form such that $\tau = \mu\tau'$. We have $(E_i\sigma)\downarrow = (E_i\theta)\downarrow\tau = (E_i\theta)\downarrow\mu\tau' = (E_i\theta\mu)\downarrow\tau' = (E_i(\theta\mu)\downarrow)\downarrow\tau'$, and $(v_i\sigma)\downarrow = (v_i\theta)\downarrow\tau = (v_i\theta)\downarrow\mu\tau' = (v_i\theta\mu)\downarrow\tau' = (v_i(\theta\mu)\downarrow)\downarrow\tau'$ for all i. We recall that $(E_i\sigma)\downarrow \rightarrow^*_{\mathcal{I}} (v_i\sigma)\downarrow$, then $(E_i(\theta\mu)\downarrow)\downarrow\tau' \rightarrow^*_{\mathcal{I}} (v_i(\theta\mu)\downarrow)\downarrow\tau'$, and by Lemma 8, we have $(E_i(\theta\mu)\downarrow)\downarrow\tau' \rightarrow^*_{\mathcal{I}'} (v_i(\theta\mu)\downarrow)\downarrow\tau'$ By definition of θ and μ, we have $(\theta\mu)\downarrow \in Sol_{Var}(\mathcal{U}')$ which concludes the proof of (1).
2. Let $\theta \in Sol_{Var}(\mathcal{U}')$, and let τ be a substitution in normal form such that $(E_i\theta)\downarrow\tau \rightarrow^*_{\mathcal{I}'} (v_i\theta)\downarrow\tau$ for all i. Lemma 8 implies that $(E_i\theta)\downarrow\tau \rightarrow^*_{\mathcal{I}} (v_i\theta)\downarrow\tau$ for all i. This implies that $(E_i\theta\tau)\downarrow \rightarrow^*_{\mathcal{I}} (v_i\theta\tau)\downarrow$ for all i, and hence, $(E_i(\theta\tau)\downarrow)\downarrow \rightarrow^*_{\mathcal{I}} (v_i(\theta\tau)\downarrow)\downarrow$ for all i. Let $\sigma = (\theta\tau)\downarrow$, we have $(E_i\sigma)\downarrow \rightarrow^*_{\mathcal{I}} (v_i\sigma)\downarrow$

for all i. Since $\theta \in Sol_{Var}(\mathcal{U}')$, we deduce that $\theta \models_{\mathcal{H}} \mathcal{U}'$, and hence $\theta \models_{\mathcal{H}} \mathcal{U}$. This implies that $\theta\tau \models_{\mathcal{I}} \mathcal{U}$, then $(\theta\tau)\downarrow \models_{\mathcal{I}} \mathcal{U}$ and thus $\sigma \models_{\mathcal{I}} \mathcal{U}$, which concludes the proof of (2).

■

2.2 Cryptographic protocols

In this section, we introduce how we model protocols.

2.2.1 Specification of protocols

A k-party protocol consists in k roles glued together with an association that maps each step of a role that expects a message m to the step of another role where the message m is produced. This association essentially defines how the execution of the protocol should proceed in the absence of the intruder. We give next the high level specification of a protocol.

Definition 29 *(Protocol) The high level specification of a k-party protocol is given by a scenario, sequence of rules of the form "$\mathfrak{R}_1 \Rightarrow \mathfrak{R}_2 : m$" with $\mathfrak{R}_1, \mathfrak{R}_2$ are two roles and $m \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ is the exchanged message. This scenario describes how the execution of a protocol should proceed in the absence of the intruder.*

Example 9 *The Needham-Schroeder symmetric key protocol [163] (presented in chapter 1, at Section 1.3) is specified as follows:*

$$\mathfrak{P}_{NS} : \begin{cases} 1.\ A \Rightarrow S : & \langle A, \langle B, N_A \rangle \rangle \\ 2.\ S \Rightarrow A : & enc^s(\langle N_A, B, K_{AB}, enc^s(\langle K_{AB}, A \rangle, K_{BS}) \rangle, K_{AS}) \\ 3.\ A \Rightarrow B : & enc^s(\langle K_{AB}, A \rangle, K_{BS}) \\ 4.\ B \Rightarrow A : & enc^s(N_B, K_{AB}) \\ 5.\ A \Rightarrow B : & enc^s(N_B - 1, K_{AB}) \end{cases}$$

N_A (respectively N_B) represents the nonce freshly created by A (respectively B), K_{AS} (respectively K_{BS}) represents the secret key shared between A (respectively B) and the trusted server, and K_{AB} the session key shared between A and B.

In this protocol, we have three roles, the trusted server (S), sender's role (A) and receiver's role (B).

Definition 30 *(Specification of role) A role $\mathcal{R}$ is given by a couple $(\{v_i \Rightarrow S_i;\ \mathcal{U}_i\}_{i \in I}, K_{\mathfrak{R}})$ where:*

- *for every i, $v_i \in \mathcal{X}$, $S_i \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, and $\mathcal{U}_i$ is an unification system,*

- *$K_{\mathfrak{R}}$ is a set of terms in $\mathcal{T}(\mathcal{F}, \mathcal{X})$ describing the knowledge of the role $\mathcal{R}$,*
- *$\{v_i \Rightarrow S_i; \mathcal{U}_i\}_{i \in I}$ is the set of rules describing the behaviour of that role, that is the set of actions that he should follow, and*
- *I is a totally ordered set of integers.*

For simplicity, we assume $I = \{1, 2, \ldots\}$. A rule $v_i \Rightarrow S_i; \mathcal{U}_i$ means: at step i, the role will receive a message, stored in v_i, and then send a message represented by S_i if the tests represented by $\mathcal{U}_i$ succeed. At each step i, the message S_i is created by the role from his initial knowledge, from the previously received messages stored in $v_1, \ldots, v_i$, and from fresh created values. A fresh value is represented by a variable appearing in S_i and not in $\{v_1, \ldots, v_i\}$. The set of fresh values for the role $\mathcal{R}$ is the set $\bigcup_{i=1}^{\sharp I}(Var(S_i) \setminus \{v_1, \ldots, v_i\})$. We denote by parameters of a role $\mathfrak{R}$ the set constructed from his initial knowledge $K_{\mathfrak{R}}$, and his fresh values. We then remark that each variable in S_i is either in $\{v_1, \ldots, v_i\}$ or represents a parameter. The fact that I is totally ordered means that the rules describing the role are sequential, and they are executed in a specific order given by the protocol. A role represents an abstract participant in the protocol, and thus in the description of a role, we do not specify which concrete participant plays the role. We remark that a role can be played by many concrete participants and the same concrete participant can play many roles or the same role many times. We observe that a "receive" is always coupled with a "send". This is because we suppose that if a received message is as expected, the role will send his response.

Example 10 *We consider the Needham-Schroeder symmetric key protocol described in Example 9. In that protocol, we have three roles: the trusted server (S), sender's role (A) and receiver's role (B). The role server (or S) is given by the tuple constructed as follows:*

- $K_S = \{A, B, S, K_{AS}, K_{BS}, K_{AB}\}$,
- *and the set of rules is:*

$$S1 : v_1 \Rightarrow enc^s(\langle \pi_2(\pi_2(v_1)), \langle \pi_1(\pi_2(v_1)), \langle K_{\pi_1(v_1)\pi_1(\pi_2(v_1))}, enc^s(\langle K_{\pi_1(v_1)\pi_1(\pi_2(v_1))}, \pi_1(v_1)\rangle, K_{\pi_1(\pi_2(v_1))S})\rangle\rangle\rangle, K_{\pi_1(v_1)S});\ v_1 \stackrel{?}{=} \langle X_1, Y_1, Z_1\rangle$$

The role sender (or A) is given by a tuple constructed by the follow:

- $K_A = \{B, S, A, N_A, K_{AS}\}$,
- *and the set of rules is:*

$$A1 : \ \emptyset \Rightarrow \langle A, \langle X_2, N_A\rangle\rangle; \emptyset$$
$$A2 : \ v_2 \Rightarrow \pi_2(\pi_2(\pi_2(dec^s(v_2, K_{AS})))); \ v_2 \stackrel{?}{=} enc^s(\langle N_A, \langle X_2, \langle Y_2, Z_2\rangle\rangle\rangle, K_{AS})$$
$$A3 : \ v_3 \Rightarrow enc^s(dec^s(v_3, \pi_1(\pi_2(\pi_2(dec^s(v_2, K_{AS}))))) - 1, \pi_1(\pi_2(\pi_2(dec^s(v_2, K_{AS}))))); v_3 \stackrel{?}{=} enc^s(X_2', \pi_1(\pi_2(\pi_2(dec^s(v_2, K_{AS}))))))$$

And the role receiver (or B) is given by a tuple constructed by the follow:

- $K_B = \{B, S, A, N_B, K_{BS}\}$,
- *and the set of rules is:*

$$B1: \quad v_4 \Rightarrow enc^s(N_B, \pi_1(dec^s(v_4, K_{BS}))); v_4 \stackrel{?}{=} enc^s(\langle X_3, Y_3\rangle, K_{BS})$$
$$B2: \quad v_5 \Rightarrow \emptyset; v_5 \stackrel{?}{=} enc^s(N_B - 1, \pi_1(dec^s(v_4, K_{BS})))$$

Before describing the execution of a protocol, we need to describe the execution of a role. In the high level specification of a protocol, a role represents an *abstract participant* in the protocol, while a *concrete participant* of the protocol, denoted *an agent (or instance of role)*, is as represented below:

Definition 31 *(Instance of role) An instance of role $\mathfrak{R}$ is obtained from the role $\mathfrak{R}$ by instantiating its parameters, that is its initial knowledge, $K_{\mathfrak{R}}$, and its fresh values. If $\mathfrak{R}$ is a role, and σ is a substitution instantiating the parameters of $\mathfrak{R}$, $\mathfrak{R}\sigma$ is an instance of that role.*

Let an instance of role defined by $(\{v_i \Rightarrow S_i; \mathcal{U}_i\}_{i\in I}, K)$, by definition of role (Definition 30), and by definition of instance of role (Definition 31), it is easy to see that $Var(S_i) \subseteq \{v_1, \ldots, v_i\}$ for $i \in \{1, \ldots, \natural I\}$.

An *agent*, also called a *concrete participant in the protocol*, should play a role in that protocol. To this end, he should instantiate that role. An agent could instantiate many roles, or instantiate the same role many time.

Definition 32 *(Instance of protocol) An instance of a protocol $\mathfrak{P}$ is defined by a union of instances of $\mathfrak{P}$ roles and by a set of ground terms representing the initial knowledge of the intruder $\mathcal{I}$. We denote an instance of the protocol $\mathfrak{P}$ as follows: $(\{v_i \Rightarrow S_i; \mathcal{U}_i\}_{i\in I}, <_I, K_{\mathcal{I}})$ where*

- *$K_{\mathcal{I}}$ is a set of ground terms in $\mathcal{T}(\mathcal{F})$ representing the initial knowledge of the intruder,*
- *I is a set of integers partially ordered by $<_I$, and*
- *for every i, $v_i \Rightarrow S_i; \mathcal{U}_i$ is a rule where $v_i \in \mathcal{X}$, S_i is a term in $\mathcal{T}(\mathcal{F}, \mathcal{X})$, and $\mathcal{U}_i$ is an unification system.*

The set of rules $\{v_i \Rightarrow S_i; \mathcal{U}_i\}_{i\in I}$ is the union of rules specifying the different instances of roles we are considering.

Since each instance of role instantiates role parameters, we remark that if v is a variable in S_i for an integer i, then there is another integer j such that $j \leq_I i$ and $v = v_j$, that is v represents a message previously received.

Example 11 *We consider again the Needham-Schroeder symmetric key protocol. We consider three agents $Alice, Bob$, and s such that $Alice$ instantiates once the sender role (or role A), Bob instantiates once the receiver role (or role B), and s instantiates once the server role (or role S). This instance of Needham-Schroeder symmetric key protocol is given by $(\{r_1, r_2, r_3, r_4, r_5, r_6\}, <_I, K_{\mathcal{I}})$ where:*
- $r_1 = S_1(s)$:
$v_1 \Rightarrow enc^s(\langle \pi_2(\pi_2(v_1)), \langle \pi_1(\pi_2(v_1)), \langle K_{\pi_1(v_1)\pi_1(\pi_2(v_1))}, enc^s(\langle K_{\pi_1(v_1)\pi_1(\pi_2(v_1))}, \pi_1(v_1)\rangle, K_{\pi_1(\pi_2(v_1))s})\rangle\rangle\rangle, K_{\pi_1(v_1)s})$;
$v_1 \stackrel{?}{=} X_1, Y_1, Z_1$
- $r_2 = A_1(Alice) : \emptyset \Rightarrow \langle Alice, \langle Bob, n_a\rangle\rangle; \emptyset$
- $r_3 = A_2(Alice) : v_2 \Rightarrow \pi_2(\pi_2(\pi_2(dec^s(v_2, K_{as})))); v_2 \stackrel{?}{=} enc^s(\langle n_a, \langle Bob, \langle Y_2, Z_2\rangle\rangle\rangle, K_{as})$
- $r_4 = A_3(Alice) : v_3 \Rightarrow enc^s(dec^s(v_3, \pi_1(\pi_2(\pi_2(dec^s(v_2, K_{as}))))) - 1, \pi_1(\pi_2(\pi_2(dec^s(v_2, K_{as})))))$;
$v_3 \stackrel{?}{=} enc^s(X_2', \pi_1(\pi_2(\pi_2(dec^s(v_2, K_{as})))))$
- $r_5 = B_1(Bob) : v_4 \Rightarrow enc^s(n_b, \pi_1(dec^s(v_4, K_{bs}))); v_4 \stackrel{?}{=} enc^s(\langle X_3, Y_3\rangle, K_{bs})$
- $r_6 = B_2(Bob) : v_5 \Rightarrow \emptyset; v_5 \stackrel{?}{=} enc^s(n_b - 1, \pi_1(dec^s(v_4, K_{bs})))$
- *The integers in the set $I = \{1, \ldots, 6\}$ are partially ordered as follows: $2 < 3 < 4$, and $5 < 6$,*
- $K_{\mathcal{I}} = \{Alice, Bob, s, \mathcal{I}, K_{\mathcal{I}s}\}$

2.2.2 Execution of protocols

Definition 33 *(Execution of protocol) Given an instance of a protocol defined by the tuple $(\{v_i \Rightarrow S_i; \mathcal{U}_i\}_{i\in I}, <_I, K_{\mathcal{I}})$, an execution of that instance is defined as follows:*

Let $I' \subseteq I$ such that for each $i, j \in I$, if $j <_I i$ and if $i \in I'$ then $j \in I'$. Let $<_{I'}$ be a total order over I' such that for any $i, j \in I'$, if $j <_I i$ then $j <_{I'} i$. An execution is defined by the couple $(\{v_i \Rightarrow S_i; \mathcal{U}_i\}_{i\in I'}, <_{I'})$.

We remark that $Var(S_i) \subseteq \{v_1, \ldots, v_i\}$ for $i \in \{1, \ldots, \sharp I'\}$. Note that in an execution of protocol, not all instantiated roles need to be represented and instantiated roles need not execute all their rules. Note also that in the first rule in any execution, that is the rule $v_{i_0} \Rightarrow S_{i_0}; \mathcal{U}_{i_0}$ where i_0 is the smallest integer in I' with respect to the order $<_{I'}$, $v_{i_0} = \mathcal{U}_{i_0} = \emptyset$. Actually, the first rule represents the first rule of the agent who runs the execution, and S_{i_0} is the first message sent and which is not a response to a received message. It is easy to see that if the instance is given by a finite set of rules then we have a finite number of possible executions of that instance.

Example 12 *We consider the instance of protocol given in Example 11. An execution of that instance is given by $\{r_2, r_1, r_3, r_5\}$ where the integers $1, 2, 3$ and 5 are totally*

ordered as follows: $2 < 1 < 3 < 5$. *More formally, the following sequence of rules*

$$\begin{array}{l}
\emptyset \Rightarrow \langle Alice, \langle Bob, n_a\rangle\rangle; \emptyset \\
v_1 \Rightarrow enc^s(\langle \pi_2(\pi_2(v_1)), \langle \pi_1(\pi_2(v_1)), \langle K_{\pi_1(v_1)\pi_1(\pi_2(v_1))}, enc^s(\langle K_{\pi_1(v_1)\pi_1(\pi_2(v_1))}, \pi_1(v_1)\rangle, \\
\qquad K_{\pi_1(\pi_2(v_1))s})\rangle\rangle\rangle, K_{\pi_1(v_1)s}); \\
\qquad v_1 \stackrel{?}{=} X_1, Y_1, Z_1 \\
v_2 \Rightarrow \pi_2(\pi_2(\pi_2(dec^s(v_2, K_{as})))); \; v_2 \stackrel{?}{=} enc^s(\langle n_a, \langle Bob, \langle Y_2, Z_2\rangle\rangle\rangle, K_{as}) \\
v_4 \Rightarrow enc^s(n_b, \pi_1(dec^s(v_4, K_{bs}))); v_4 \stackrel{?}{=} enc^s(\langle X_3, Y_3\rangle, K_{bs})
\end{array}$$

represents an execution of the instance given in Example 11.

2.3 From cryptographic protocols to constraint systems

Constraint systems are quite common in modelling security protocols for a bounded number of sessions [156, 87, 73, 72]. Actually, many protocol security properties can be characterised as reachability problems which are converted to constraint solving problems. This happens for properties such as secrecy, where the objective of protocol analysis is to search for an execution of the protocol in which some secret data has been released publicly by the intruder. Although the reachability is undecidable for cryptographic protocols in the general case [110], some decidability results have been obtained for restricted cases, for instance, in [178, 15, 156], the authors proved the decidability of reachability for cryptographic protocols in the case of finite number of sessions.

We show here how constraint systems can be used to analyse cryptographic protocols. We present in Section 2.3.1 how to built constraint system from an execution of a protocol, and in Section 2.3.2 we show how to reduce insecurity problem of a protocol with a bounded number of sessions to the satisfiability problem of constraint systems. We follow the same definitions, constructions and notations given in previous works [156, 73].

2.3.1 From an execution of a protocol to a constraint system

Given an instance of a protocol, and let $\mathfrak{exec}$ be an execution of the given instance of protocol. Assume $\mathfrak{exec} = \{\emptyset \Rightarrow S_1, v_2 \Rightarrow S_2; \mathcal{U}_2, \ldots, v_n \Rightarrow S_n; \mathcal{U}_n\} = \{\emptyset \Rightarrow S_1, v_2 \Rightarrow S_2, \ldots, v_n \Rightarrow S_n; \mathcal{U}_2, \ldots, \mathcal{U}_n\}$. The constraint system associated with the execution $\mathfrak{exec}$, denoted by $\mathcal{C}_{\mathfrak{exec}}$, and with the initial intruder knowledge $K_{\mathcal{I}}$ is:

$$\mathcal{C}_{\mathfrak{exec}} = (E_1 \vdash v_2, \ldots, E_{n-1} \vdash v_n, \mathcal{U}) \text{ where:}$$

- E_1 is a set of ground terms in $\mathcal{T}(\mathcal{F})$ representing the initial intruder knowledge and the first sent message, $E_1 = K_{\mathcal{I}} \cup S_1$,

- for every i in the set $\{2, \ldots, n-1\}$, E_i is a set of terms included in $\mathcal{T}(\mathcal{F}, \mathcal{X})$ defined as $E_i = E_{i-1} \cup S_i$
- $\mathcal{U} = \mathcal{U}_2 \cup \ldots \cup \mathcal{U}_n$.

It is easy to see that $\mathcal{C}_{\mathfrak{exec}}$ is a constraint system in the sense of Definition 20. In the context of cryptographic protocols, the inclusion $E_i \subseteq E_{i-1}$ for every $i \in \{1, \ldots, n-1\}$ means that the knowledge of an intruder does not decrease as the protocol progresses: after receiving a message a honest agent will respond to it, this response can then be added to the knowledge of the intruder who listens to all communications. The condition on variables, $Var(E_i) \subseteq \{v_1, \ldots, v_{i-1}\}$, stems from the fact that a message sent at some step i must be built from previously received messages recorded in the variables v_j, $j < i$, and from the ground initial knowledge of the honest agents.

An execution $\mathfrak{exec}$ is said to be a *possible execution* if the correspondant constraint system $\mathcal{C}_{\mathfrak{exec}}$ is satisfiable, that is, there is a substitution σ satisfying $\mathcal{C}_{\mathfrak{exec}}$ with respect to the intruder $\mathcal{I}$.

Example 13 *We consider the instance of Needham-Schroeder symmetric key protocol given in Example 11, and we consider an execution of that instance given by the following sequence of rules*

$$\begin{array}{l} \emptyset \Rightarrow \langle Alice, \langle Bob, n_a \rangle \rangle \\ v_1 \Rightarrow enc^s(\langle \pi_2(\pi_2(v_1)), \langle \pi_1(\pi_2(v_1)), \langle K_{\pi_1(v_1)\pi_1(\pi_2(v_1))}, enc^s(\langle K_{\pi_1(v_1)\pi_1(\pi_2(v_1))}, \pi_1(v_1) \rangle, \\ \qquad K_{\pi_1(\pi_2(v_1))s} \rangle \rangle \rangle, K_{\pi_1(v_1)s} \rangle; \\ \qquad v_1 \stackrel{?}{=} \langle X_1, \langle Y_1, Z_1 \rangle \rangle \end{array}$$

The associated constraint system is defined as follows: $\mathcal{C} = (E_1 \vdash v_1, \mathcal{U})$ *and* $\mathcal{U} = \left\{ v_1 \stackrel{?}{=} \langle X_1, \langle Y_1, Z_1 \rangle \rangle \right\}$. *The intruder deduction system considered here is the Dolev-Yao intruder system with explicit destructors,* $\mathcal{I}^e_{DY}$. *It is easy to see that the constraint system given above has a solution which is given by the substitution* $\sigma = \{X_1 \mapsto Alice, Y_1 \mapsto Bob, Z_1 \mapsto n_a, x_1 \mapsto Alice, x_2 \mapsto Bob, x_3 \mapsto n_a\}$. *Thus, the execution given above is a possible execution.*

2.3.2 From insecurity problem to satisfiability of constraint systems

We are concerned here with the secrecy property.

Secrecy. The secrecy property can be expressed by requiring that the secret data s is not deducible from the messages sent on the network. Assume S to be the set of messages sent on the network, the secrecy of the message (data) s is preserved if and only if s is not deduced from S, that is $s \notin \bar{S}^{\mathcal{I}}$ where $\mathcal{I}$ is the given intruder system. Here, we are interested by the secrecy property in the following context:

"given an execution $\mathfrak{exec}$ of an instance of a protocol, does this execution preserve secrecy?"

An execution $\mathfrak{exec}$ preserves the secrecy of the data s if and only if the intruder, at the end of that execution, is not able to deduce the data s. While the execution $\mathfrak{exec} = \{\emptyset \Rightarrow S_1, v_2 \Rightarrow S_2, \ldots, v_n \Rightarrow S_n; \mathcal{U}_2, \ldots, \mathcal{U}_n\}$ is associated to the constraint system $\mathcal{C}_{\mathfrak{exec}} = (E_1 \vdash v_2, \ldots, E_{n-1} \vdash v_n, \mathcal{U})$ defined as in Section 2.3.1, the knowledge of the intruder at the end of $\mathfrak{exec}$ is $E_n = E_{n-1} \cup S_n$, and hence, the secrecy of s is preserved in that execution if and only if $s \notin \overline{E_n}^{\mathcal{I}}$. We conclude that the secrecy property of the execution can be encoded directly by adding an additional constraint $E_n \vdash v_{n+1}$ and additional equation $v_{n+1} \stackrel{?}{=}_{\mathcal{H}} s$ to $\mathcal{C}_{\mathfrak{exec}}$, and asking for the satisfiability of the new constraint system, also called *extended constraint system*. We say that a protocol preserves the secrecy property if and only if each execution of that protocol preserves the secrecy property, and to verify whether the secrecy property is preserved by a protocol for a *"bounded number of sessions"* (*i.e.* an instance of a protocol), it is then sufficient to check whether the secrecy property is preserved by each execution of that instance. In [156] and [88], the authors showed that not all executions of an instance of a protocol need to be enumerated. But anyhow, here we are only interested whether an arbitrary execution of an instance of a protocol preserves the secrecy property, and deciding whether an execution preserves the secrecy property is done by deciding the satisfiability of the correspondant extended constraint system. Hence, deciding whether a protocol under a bounded number of sessions is insecure, that is does not preserve the secrecy property (also called deciding the secrecy problem of a protocol under a bounded number of sessions) is reduced to deciding the satisfiability of constraint systems (also called decidability of reachability problem).

Example 14 *We consider the execution given in the Example 13. We recall the constraint system associated to that execution:* $\mathcal{C} = (E_1 \vdash v_1, \mathcal{U})$ *and* $\mathcal{U} = \left\{ v_1 \stackrel{?}{=} \langle X_1, \langle Y_1, Z_1 \rangle \rangle \right\}$.
Let sec *be a secret data. Deciding if the given execution does not preserve the secrecy of* sec *under the intruder* $\mathcal{I}^e_{DY}$ *is reduced to deciding if the following constraint system*

$$\mathcal{C} = (E_1 \vdash v_1, E_2 \vdash v_6, \mathcal{U}')$$

where $E_2 = E_1 \cup \{$
$enc^s(\langle \pi_2(\pi_2(v_1)), \langle \pi_1(\pi_2(v_1)), \langle K_{\pi_1(v_1)\pi_1(\pi_2(v_1))}, enc^s(\langle K_{\pi_1(v_1)\pi_1(\pi_2(v_1))}, \pi_1(v_1) \rangle,$
$K_{\pi_1(\pi_2(v_1))s}) \rangle \rangle \rangle, K_{\pi_1(v_1)s})\}$, *and* $\mathcal{U}' = \mathcal{U} \cup \left\{ v_6 \stackrel{?}{=} ses \right\}$
is $\mathcal{I}^e_{DY}$*-satisfiable.*

2.4 Conclusion

In this chapter, we have presented the basic notions that we will use in the next chapters. Mainly, we showed how cryptographic protocols are symbolically modeled, and how to reduce the insecurity problem of cryptographic protocols with bounded number of sessions to the satisfiability problem of constraint systems. In the next chapters, we will relax the *perfect cryptography assumption* by taking into account *some algebraic properties of cryptographic primitives*, and, following the symbolic approach given in this chapter, we will analyse cryptographic protocols in the presence of some algebraic operators.

Chapter 3

Analysis of protocols with collision vulnerable hash functions

In this chapter, we consider the class of cryptographic protocols that use collision vulnerable hash functions. Only a few years ago, it was intractable to compute collisions on hash functions, so they were considered to be collision resistant by cryptographers, and protocols were built upon this assumption. From the nineties on, several authors [98, 103, 199, 201] have proved the tractability of finding collision attacks over several hash functions, and some practical methods have been published to compute collisions on some commonly used hash functions.

Following the symbolic method introduced in Chapter 2, we reduce the insecurity problem of our class of cryptographic protocols to the ordered satisfiability problem for the intruder that uses the collision vulnerability property of hash functions when attacking a protocol execution. We give an algorithm that the intruder employs to compute collisions on hash functions. By this algorithm and roughly following the results obtained in [74], we conjecture that the ordered satisfiability problem for the intruder exploiting the collision vulnerability property of hash functions can be reduced to the ordered satisfiability problem for an intruder operating on words, that is with an associative symbol of concatenation, and we show the decidability of the last problem. The decidability of the ordered satisfiability problem for the intruder operating on words is interesting in its own right as it is the first decidability result that we are aware of for an intruder system for which unification is infinitary, and that permits to consider in other contexts an associative concatenation of messages instead of their pairing. The results of this Chapter have been published in the proceedings of *ASIAN 2006* conference [67].

Outline. In Section 3.1, we introduce hash functions and show some of their properties. In Section 3.2, we introduce the collision vulnerability property. In Section 3.3, we give the model that we use during this chapter. The collision vulnerability property is symbolically formalised in Section 3.4, and the decidability results are given in Section 3.5.

3.1 Hash functions

3.1.1 Definition of hash functions

Cryptographic hash functions, or simply *hash functions* play a fundamental role in modern cryptography. They take a message of arbitrary length as input and output a fixed-length message referred to as a *hash- code*, *hash-result*, *hash-value*, or simply *hash*.

Definition 34 *(hash function) Let D and R be respectively a set of messages of arbitrary length and a set of messages of fixed length. A hash function h is a function $h : D \rightarrow R$ which has, as a minimum, the following two properties:*

Compression
h maps an input x of arbitrary finite length, to an output $h(x)$ of fixed length.

Ease of computation
given h and an input x, $h(x)$ is easy to compute; actually, h is a linear function, that is for any input x, $h(x)$ is computable in time $O(\|x\|)$.

For any hash function $h : D \rightarrow R$, it is easy to see that $\|D\| > \|R\|$.

Applications. Hash functions have many applications in information security. A typical use of a cryptographic hash would be as follows: *Alice* poses a tough math problem to *Bob*, and claims she has solved it. *Bob* would like to try it himself, but would yet like to be sure that *Alice* is not bluffing. Therefore, *Alice* writes down her solution, appends a random nonce, computes its hash and tells *Bob* the hash value (whilst keeping the solution and nonce secret). This way, when *Bob* comes up with the solution himself a few days later, *Alice* can prove that she had the solution earlier by revealing the nonce to *Bob*. This is an example of a simple *commitment scheme*.

Another important application of secure hashes is verification of message integrity. Determining whether any change has been made to a message (or a file) can be accomplished by comparing the hash of the message calculated before and after transmission (or any other event). The hash of a message can also serve as a means of reliably identifying a file.

A related application is password verification [125]: passwords are usually not stored in cleartext, for obvious reasons, but instead in hashed form. To authenticate a user, the password presented by the user is hashed and compared with the stored hash.

Hash functions are also used in digital signature schemes [175, 139]: for both security and performance reasons, some signature schemes specify that only the hash of the message will be "signed", and not the entire message. Hash functions can also be used in the generation of pseudorandom bits [153].

3.1.2 Properties of hash functions

Let h be an arbitrary hash function, $h : D \rightarrow R$. We present here the three potential properties (in addition to *ease of computation* and *compression* given in Definition 34), for h [153]:

Preimage resistance
: given a hash $y \in R$, if the correspondant input is not known, it is computationally infeasible, *i.e.* it takes too long (hundreds of years) to compute using the fastest of super computers, to find any input $x \in D$ such that $h(x) = y$. This concept is related to that of *one way function*.

Second preimage resistance
: given an input $x \in D$, it is computationally infeasible to find another input $x' \in D$ such that $x' \neq x$ and $h(x') = h(x)$. This property is sometimes referred to as *weak collision resistance*.

Collision resistance
: it is computationally infeasible to find two distinct inputs $x, x' \in D$ such that $h(x') = h(x)$. This property is sometimes referred to as *strong collision resistance*.

We say that a *hash function is vulnerable to respectively preimage attacks, second preimage attacks, and collision attacks* if it lacks respectively preimage resistance, second preimage resistance and collision resistance property. We say also that a *hash function is respectively one-way, second-preimage resistant and collision resistant hash function* if it is a hash function as per definition 34 with respectively the following properties: preimage resistance, second preimage resistance and collision resistance property.

In [153], the authors showed the following relationships between properties of hash functions given above:

- collision resistance implies second-preimage resistance of hash functions;
- collision resistance does not guarantee preimage resistance;
- preimage resistance does not guarantee second-preimage resistance.

Motivations. We give now one motivation for each of the three major properties above. Consider a digital signature scheme wherein the signature is applied to the hash $h(x)$ rather than the message x. Here h should be a second-preimage resistant hash function, otherwise, a intruder $\mathcal{I}$ may observe the signature of some agent A on $h(x)$, then find an x' such that $h(x') = h(x)$, and claim that A has signed x'. If $\mathcal{I}$ is able to actually choose the message which A signs, then $\mathcal{I}$ needs only find a pair x, x' such that $h(x) = h(x')$ rather than the harder task of finding a second preimage of x; in this case, collision resistance is also necessary [153]. Less obvious is the requirement of preimage resistance for some public-key signature schemes; consider RSA [174], where agent A has public key (e, n). $\mathcal{I}$ may choose a random value y, compute $z = y^e \ (mod\ n)$, and claim that y is As signature on z. This attack is possible if $\mathcal{I}$ can find an input x such that $h(x) = z$.

3.1.3 Examples of hash functions

Let us consider the following function: $h(x) = x^2$ - $1\ (mod\ p)$, with p a prime number. This function h is a hash function as per Definition 34, but h is not a one-way hash function because finding x such that $h(x) = y$ for a given y is easy [153].

Let us consider the following function: $h(x) = x^2\ (mod\ n)$ with $n = p * q$, p, q are two randomly chosen primes. h is a one-way hash function because finding a x such that $h(x) = y$ for a given y is computationally equivalent to factoring and thus intractable, but finding a 2nd-preimage, and, therefore, collisions, is trivial (given x, we have that $h(x) = h(-x)$), and thus h is neither second-preimage resistant hash function nor collision resistant hash function [153].

3.2 Collision vulnerability property

A hash function is a function $h : D \rightarrow R$ with $\|D\| > \|R\|$ (Definition 34), that is a hash function is many-to-one. This implies that the existence of pair of inputs x, x' with $x \neq x'$ and $h(x) = h(x')$ is unavoidable, we call such pair of inputs a *collision*. However, only a few years ago, it was intractable to compute collisions on hash functions, so they were considered to be collision resistant by cryptographers, and protocols were built upon this assumption. From the nineties on, several authors [98, 103, 199, 201] have proved the tractability of finding pseudo-collision and collision attacks over several hash functions. Taking this into account, we consider in this chapter hash functions having the following properties: preimage resistance, second-preimage resistance, and collision vulnerability. From now on, we call *a collision vulnerable hash function* a hash function having these properties. The *collision vulnerability* means that the hash function is not collision resistant, *i.e.* it is computationally feasible to compute

two distinct inputs x and x' with $h(x) = h(x')$ provided that x and x' are created at the same time and independently one of the other.

To mount a collision attack, the intruder would typically begin by constructing two different messages with the same hash where one message appears legitimate or innocuous while the other serves the intruder's purposes.

3.2.1 Hash functions having this property

MD5 Hash function [173] is one of the most widely used cryptographic hash functions nowadays. It was designed in 1992 as an improvement on MD4 [172], and its security was widely studied since then by several authors. The first result was a pseudo-collision for MD5 [98]. When permitting to change the initialisation vector, another attack (free-start collision) has been found [103]. Recently, a real collision involving two 1024-bits messages was found with the standard value [199]. This first weakness was extended into a differential-like attack [202] and tools were developed [129, 130] for finding the collisions which work for any initialisation value and which are quicker than methods presented in [199]. Finally, other methods have been developed for finding new MD5 collisions [204, 183]. The development of collision-finding algorithms is not restricted to MD5 hash function. Several methods for MD4 [172] research attack have been developed [200, 104]. In [200] a method to search RIPEMD [105] collision attacks was also developed, and in [42], a collision on SHA-0 [7] has been presented. Finally, Wang *et al.* have developed in [201] another method to search for collisions for the SHA-1 [4] hash function.

3.2.2 Collision vulnerability in practice

We consider here the *story of Alice and her boss* [94]. *Alice* has been working for some time in the office of *Julius Caesar*. On her last day of work, *Caesar* gives her a letter of recommendation on paper. *Alice* decides to take advantage of this opportunity to gain access to *Caesar's* secret documents. *Caesar* uses MD5 hash function which is collision vulnerable (Section 3.2.1) for his digital signature scheme DSA [3]. When she receives her letter of recommendation on paper, *Alice* prepars two postscripts files with the same MD5 hash: one is the letter given by *Caesar* and the other is an order from *Caesar* to grant *Alice* some kind of secrecy clearance. She asks *Caesar* to digitally sign the letter and due to the hash collision, *Caesar's* signature for the letter of recommendation is also valid for the order. She then presents the order and the digital signature to the person in charge of *Caesar's* files, and finally gains access to *Caesar's* secret documents.

3.3 The model

To analyse cryptographic protocols, we follow in this chapter the symbolic model described in Chapter 2. To this end, we assume an infinite set of variables $\mathcal{X}$, an infinite set of constants $\mathcal{C}$, a set of function symbols $\mathcal{F}$. In addition to what is already intoducted in Chapter 2, we make use here of some additional notions that we will show next.

Given a term t, we recall that $Var(t)$ denotes the set of variables appearing in t, we recall also that $Cons(t)$ denotes the set of free constants appearing in t, that is, the set of constants from $\mathcal{C}$ appearing in t. Now, we extend the definition of $Cons(t)$ to take in consideration non-free constants, *i.e.* the function symbols in $\mathcal{F}$ with arity 0, appearing in t, and we defined $all.Cons(t)$ to denote the set of free constants of t together with the set of non-free constants of t. More formally, $all.Cons(t) = Cons(t) \cup \{f \in \mathcal{F}$ such that $arity(f) = 0$ and there is a position $p \in Pos(t)$ with $t_{|p} = f\}$. We define the set of atoms $Atoms$ to be the union of Var and $all.Cons$, if t is a term, $Atoms(t) = Var(t) \cup Cons(t)$.

In the rest of this chapter, we assume a complete simplification ordering $>$ on $\mathcal{T}(\mathcal{F}, \mathcal{X})$, *i.e.* $>$ is a simplification ordering on $\mathcal{T}(\mathcal{F}, \mathcal{X})$ total over ground terms $\mathcal{T}(\mathcal{F})$, and for which the minimal element is a free constant, *i.e.* a constant in $\mathcal{C}$, called c_{min}. We denote by C_{spec} the set consisting of c_{min} and of all symbols in $\mathcal{F}$ of arity 0, *i.e.* $C_{spec} = \{c_{min}\} \cup \{f \in \mathcal{F}$ such that $arity(f) = 0\}$.

3.3.1 Mode in an equational theory

The notion of $Mode$ *on a signature* $\mathcal{F}$ has been initially defined in [74]. Assume $\mathcal{H}$ is an equational theory over a signature $\mathcal{F}$. Assume also that F is partitioned into two disjoint sets $\mathcal{F}_0$ and $\mathcal{F}_1$, and that $\mathcal{X}$ is partitioned into two disjoint sets $\mathcal{X}_0$ and $\mathcal{X}_1$. We first define a signature function $Sign$ on $\mathcal{F} \cup Atoms$ in the following way:

$$\begin{array}{rcl} Sign & : & \mathcal{F} \cup Atoms \rightarrow \{0,1,2\} \\ Sign(f) & = & \begin{cases} 0 \text{ if } f \in \mathcal{F}_0 \cup \mathcal{X}_0 \\ 1 \text{ if } f \in \mathcal{F}_1 \cup \mathcal{X}_1 \\ 2 \text{ otherwise, i.e. when } f \text{ is a free constant } (f \in \mathcal{C}) \end{cases} \end{array}$$

The function $Sign$ is extended to terms by taking $Sign(t) = Sign(Top(t))$.

We also assume that there exists a $Mode$ function with arity 2, $Mode$, such that $Mode(f, i)$ is defined for every symbol $f \in \mathcal{F}$ and every integer i such that $1 \leq i \leq arityf$. For all valid f, i we have $Mode(f, i) \in \{0, 1\}$ and $Mode(f, i) \leq Sign(f)$. Thus for all $f \in \mathcal{F}_0$ and for all i we have $Mode(f, i) = 0$.

3.3.2 Well-moded equational theories

A position different from ε in a term t is *well-moded* if it can be written $p \cdot i$ (where p is a position and i a non negative integer) such that $Sign(t_{|p \cdot i}) = Mode(Top(t_{|p}), i)$. In other words the position in a term is well-moded if the subterm at that position is of the expected type with respect to the function symbol immediately above it. A term is *well-moded* if all its *non root* positions are well-moded. Note in particular that a well-moded term does not contain free constants (*i.e. constants in* $\mathcal{C}$). If a position of t is not well-moded we say it is *ill-moded* in t. The root position of a term is always ill-moded. An equational theory $\mathcal{H}$ is well-moded if for all equations $u \doteq v \in \mathcal{H}$ the terms u and v are well-moded and $Sign(u) = Sign(v)$, and similarly, a rule $u \to v$ with u and v are two terms is well-moded if u and v are well-moded and $Sign(u) = Sign(v)$. In [74], the authors proved that if an equational theory is well-moded then the application of Unfailing Knuth-Bendix completion procedure (Chapter 2, Section 2.1.6) outputs a well-moded system.

We remark also that if $\mathcal{H}$ is the union of two equational theories $\mathcal{H}_0$ and $\mathcal{H}_1$ over two disjoint signatures $\mathcal{F}_0$ and $\mathcal{F}_1$, the theory $\mathcal{H}$ is well-moded when assigning Mode i to each argument of each operator $f \in \mathcal{F}_i$, for $i \in \{0, 1\}$.

3.3.3 Subterm values

The notion of Mode also permits to define a new subterm relation in $\mathcal{T}(\mathcal{F}, \mathcal{X})$. This relation has been initially defined in [74].

We call a *subterm value* of a term t a subterm (as defined in Section 2.1.3, Chapter 2) of t that is either atomic or occurs at an ill-moded position of t. We denote $Sub_v(t)$ the set of subterm values of t. We extend as expected the notion of subterm value to a set of terms E, $Sub_v(E) = \bigcup_{e \in E} Sub_v(e)$. We denote by $factors(t)$, for a term t, the subset of the maximal and strict subterm values of t.

Example 15 *Consider two binary function symbols f and g with $Sign(f) = Sign(g) = Mode(f, 1) = Mode(g, 1) = 1$ and $Mode(f, 2) = Mode(g, 2) = 0$, and $t = f(f(g(a, b), f(c, c)), d)$. Its subterm values are a, b, $f(c, c)$, c, d, and its factors are a, b, $f(c, c)$ and d.*

In the rest of this chapter, *the notion of subterm will refer to subterm values*.

3.3.4 Intruder deduction system

As stated before, our concern in this chapter is the symbolic analysis of cryptographic protocols using collision vulnerable hash functions. In chapter 2, we show how to reduce the insecurity problem of cryptographic protocols to the satisfiability problem of constraint solving. This reduction is done assuming the intruder deduction rules are of the form:

$$x_1, \ldots, x_n \rightarrow f(x_1, \ldots, x_n)$$

with f a public function symbol in $\mathcal{F}$.

Unfortunately, such intruder deduction rules are not sufficient to represent an intruder taking advantage of the collision vulnerability property for hash functions. To this end, we introduce next another definition of intruder deduction rules. This representation of intruder deduction rules have been given initially in [72].

Definition 35 *(Intruder deduction rules) An intruder deduction rule is a rule of the form*

$$l_1, \ldots, l_n \rightarrow l$$

with

- $l_1, \ldots, l_n, l \in \mathcal{T}(\mathcal{F}, \mathcal{X})$,
- $all.Cons((l\sigma)\downarrow) \subseteq \bigcup_{i=1}^{n} all.Cons((l_i\sigma)\downarrow) \cup C_{spec}$, *for any ground substitution* σ.

This second condition in the definition above is very similar to the *origination* condition for well-definedness in [157]. It is easy to see that if the equation theory $\mathcal{H}$ verifies the property: $Var(u) = Var(v)$ for each $u \doteq v \in \mathcal{H}$, then the second condition in the definition above is verified if and only if $Var(l) \subseteq Var(\{l_1, \ldots, l_n\})$.

Example 16 *Let $\mathcal{F} = \{., \epsilon\}$ with . denotes the concatenation, and ϵ denotes the empty word, and let*

$$\mathcal{H} = \begin{cases} (x.y).z = x.(y.z) \\ x.\epsilon = x \\ \epsilon.x = x \end{cases}$$

be the associated equational theory. The following rule

$$x.y \rightarrow x$$

is an intruder deduction rule as per definition 35.

Definition 36 *(Intruder deduction system) An intruder deduction system $\mathcal{I}$, also called an intruder system, is a triple $\mathcal{I} = \langle \mathcal{F}, \mathcal{L}_{\mathcal{I}}, \mathcal{H} \rangle$ where $\mathcal{F}$ is a signature, $\mathcal{L}_{\mathcal{I}}$ is a set of intruder deduction rules (as per Definition 35), and $\mathcal{H}$ is an equational theory over $\mathcal{T}(\mathcal{F}, \mathcal{X})$.*

Given two set of terms $E, F \subseteq \mathcal{T}(\mathcal{F}, \mathcal{X})$, $E \rightarrow_{\mathcal{I}} F$ and $\mathcal{I}$-derivations are defined as in Chapter 2.

Definition 37 *(well-formed derivation) Let $\mathcal{I} = \langle \mathcal{F}, \mathcal{L}_{\mathcal{I}}, \mathcal{H} \rangle$ be an intruder deduction system, and let E (respectively t) be a set of ground terms (respectively a ground term) in normal form. A derivation $D : E \rightarrow_{\mathcal{I}} E, t_1, \rightarrow_{\mathcal{I}} \ldots \rightarrow_{\mathcal{I}} E, t_1, \ldots, t_{n-1}, t$ is well-formed if for every $1 \in \{1, \ldots, n-1\}$, $t_i \in Sub(E \cup t)$.*

Definition 38 *(locality) Let $\mathcal{I} = \langle \mathcal{F}, \mathcal{L}_{\mathcal{I}}, \mathcal{H} \rangle$ be an intruder deduction system. $\mathcal{I}$ is local if for every E, t with E a set of ground terms in normal form and t a ground term in normal form, if there is a derivation starting from E of goal t then there is a well-formed derivation starting from E of goal t.*

Example 17 *We present here an intruder operating on words who is able to concatenate messages and extract prefixes and suffixes. The intruder system is given by $\langle \mathcal{F}, \mathcal{L}_{\mathcal{I}}, \mathcal{H} \rangle$ with:*

- $\mathcal{F} = \{., \epsilon\}$; *. denotes the concatenation, and ϵ denotes the empty word*
- $\mathcal{L}_{\mathcal{I}} = \begin{cases} x, y \rightarrow x.y \\ x.y \rightarrow x \\ x.y \rightarrow y \\ \emptyset \rightarrow \epsilon \end{cases}$
- $\mathcal{H} = \begin{cases} (x.y).z = x.(y.z) \\ x.\epsilon = x \\ \epsilon.x = x \end{cases}$

Combination of intruder deduction systems

We recall in Definition 39 the notion of *disjoint combination (also called disjoint union) of intruder deduction systems*. This notion has been initially introduced by Y. Chevalier and M. Rusinowitch [73] who used this notion to show that the analysis of cryptographic protocols in presence of an intruder deduction system $\mathcal{I}$ may be reduced to the analysis of cryptographic protocols in presence of simpler intruder deduction systems $\mathcal{I}_1, \ldots, \mathcal{I}_n$ provided that $\mathcal{I}$ is the disjoint union of $\mathcal{I}_1, \ldots, \mathcal{I}_n$.

Definition 39 *Let $\mathcal{I}_1 = \langle \mathcal{F}_1, \mathcal{L}_{\mathcal{I}1}, \mathcal{H}_1 \rangle$ and $\mathcal{I}_2 = \langle \mathcal{F}_2, \mathcal{L}_{\mathcal{I}2}, \mathcal{H}_2 \rangle$ be two intruder deduction systems such that $\mathcal{F}_1$ and $\mathcal{F}_2$ are disjoints, $\mathcal{H}_1$ an equational theory on $\mathcal{F}_1$ and $\mathcal{H}_2$ an equational theory on $\mathcal{F}_2$. The following intruder deduction system $\mathcal{I} = \langle \mathcal{F}_1 \cup \mathcal{F}_2, \mathcal{L}_{\mathcal{I}1} \cup \mathcal{L}_{\mathcal{I}2}, \mathcal{H}_1 \cup \mathcal{H}_2 \rangle$ denotes the disjoint union of the two intruder deduction systems $\mathcal{I}_1$ and $\mathcal{I}_2$.*

3.3.5 Symbolic derivation

Given a intruder system $\mathcal{I} = \langle \mathcal{F}, \mathcal{L}_\mathcal{I}, \mathcal{H} \rangle$, and a protocol $\mathfrak{P}$. At *step* i, a role in $\mathfrak{P}$ receives a (possibly $\emptyset$) message, checks its well-formedness by verifying that it is as expected, and applies rules in $\mathcal{L}_\mathcal{I}$ to construct the response. We give in Definition 30 (Chapter 2) a specification of role, this specification is adequate when the intruder deduction rules are of the form $x_1, \ldots, x_n \rightarrow f(x_1, \ldots, x_n)$ with f a public function symbol in $\mathcal{F}$. Since, in this chapter, we consider a different form of the intruder deduction rules (Definition 35), we represent a role by a *symbolic derivation* [72].

Definition 40 *(Symbolic Derivations) Let $\mathcal{I} = \langle \mathcal{F}, \mathcal{L}_\mathcal{I}, \mathcal{H} \rangle$ be an intruder deduction system. A $\mathcal{I}$-symbolic derivation is a tuple $(\mathcal{V}, \mathcal{S}, \mathcal{K}, In, Out)$ where $\mathcal{V}$ is a finite sequence of variables $(x_i)_{i \in Ind}$, indexed by a linearly ordered set $(Ind, <)$, $\mathcal{K}$ is a set of ground terms (the initial knowledge), In, Out are two disjoint subsets of Ind, and $\mathcal{S}$ is a set of equations such that, for all $x_i \in \mathcal{V}$ one of the following holds:*

- $i \in In$;
- *There exists a ground term $t \in \mathcal{K}$, and an equation $x_i \stackrel{?}{=} t$ in $\mathcal{S}$;*
- *There exists a rule $l_1, \ldots, l_m \rightarrow r \in \mathcal{L}_\mathcal{I}$ such that $\mathcal{S}$ contains the equations $x_i \stackrel{?}{=} r$ and $x_{\alpha_j} \stackrel{?}{=} l_j$ for $j \in \{1, \ldots, m\}$ with $\alpha_j < i$.*

A symbolic derivation is closed *if $In = Out = \emptyset$, and in this case, it may be simply denoted $(\mathcal{V}, \mathcal{S}, \mathcal{K})$. A substitution σ satisfies a closed symbolic derivation $\mathcal{C} = (\mathcal{V}, \mathcal{S}, \mathcal{K})$, and we write $\sigma \models_\mathcal{I} \mathcal{C}$, if $\sigma \models_\mathcal{H} \mathcal{S}$.*

Let $(\mathcal{V}, \mathcal{S}, \mathcal{K}, In, Out)$ be a symbolic derivation. For simplicity, from now on, we replace every index i in the set of indices Ind by the variable x_i associated to this index, and we do not model equations $x_i \stackrel{?}{=} t \in \mathcal{S}$ for $t \in \mathcal{K}$, writing t directly when x_i is needed, but we keep x_i in $\mathcal{V}$.

Example 18 *We consider the following simple protocol:*

$$\begin{array}{ll} A \Rightarrow B: & enc^p(\langle A, N_A \rangle, Pk_B) \\ B \Rightarrow A: & enc^p(N_B, Pk_A) \end{array}$$

The role A is represented by the following symbolic derivation:

$$(\mathcal{V}_A, \mathcal{S}_A, \mathcal{K}_A, In_A, Out_A) \text{ with:}$$

- $\mathcal{V}_A = \{x_{01}^A, x_{02}^A, x_{03}^A, x_{04}^A, x_{05}^A, x_{06}^A, x_1^A, x_2^A, y_1^A, y_2^A\}$, *ordered as follows:* $x_{01}^A < x_{02}^A < x_{03}^A < x_{04}^A < x_{05}^A < x_{06}^A < x_1^A < y_1^A < x_2^A < y_2^A$
- $\mathcal{K}_A = \{A, B, N_A, K_A, K_A^{-1}, K_B\}$,

- $In_A = \{x_1^A, x_2^A\}$,
- $Out_A = \{y_1^A, y_2^A\}$,
- $\mathcal{S}_A = \left\{x_1^A \stackrel{?}{=} \emptyset, x_2^A \stackrel{?}{=} enc^p(x, Pk_A), y_1^A \stackrel{?}{=} enc^p(\langle A, N_A\rangle, Pk_B), y_2^A \stackrel{?}{=} \emptyset\right\}$

The role B is represented by the following symbolic derivation:

$$(\mathcal{V}_B, \mathcal{S}_B, \mathcal{K}_B, In_B, Out_B) \text{ with:}$$

- $\mathcal{V}_B = \{x_{01}^B, x_{02}^B, x_{03}^B, x_{04}^B, x_{05}^B, x_{06}^B, x_1^B, y_1^B\}$, *ordered as follows:* $x_{01}^B < x_{02}^B < x_{03}^B < x_{04}^B < x_{05}^B < x_{06}^B < x_1^B < y_1^B$
- $\mathcal{K}_B = \{A, B, N_B, K_B, K_B^{-1}, K_A\}$,
- $In_B = \{x_1^B\}$,
- $Out_A = \{y_1^B\}$,
- $\mathcal{S}_B = \left\{x_1^B \stackrel{?}{=} enc^p(y, Pk_B), y_1^A \stackrel{?}{=} enc^p(N_B, Pk_{\pi_1(dec^p(x_1^B, Pk_B^{-1}))})\right\}$

Composition of symbolic derivations

Given two $\mathcal{I}$-symbolic derivations, we show next how to compose them [72].

Definition 41 *Let $\mathcal{I} = \langle \mathcal{F}, \mathcal{L}_\mathcal{I}, \mathcal{H}\rangle$ be an intruder deduction system, and let $\mathcal{C}_1 = (\mathcal{V}_1, \mathcal{S}_1, \mathcal{K}_1, In_1, Out_1)$ and $\mathcal{C}_2 = (\mathcal{V}_2, \mathcal{S}_2, \mathcal{K}_2, In_2, Out_2)$ be two $\mathcal{I}$-symbolic derivations with two disjoint sets of variables and index sets $(Ind_1, <_1)$, $(Ind_2, <_2)$ respectively. Let I_1, I_2, O_1, O_2 be subsets of In_1, In_2, Out_1, Out_2 respectively. We recall that every index i in the set of indices Ind has been replaced by the variable x_i associated to this index. Assume that there is an order preserving bijection ϕ from $I_1 \cup I_2$ to $O_1 \cup O_2$ such that $\phi(I_1) = O_2$ and $\phi(I_2) = O_1$. A composition of two $\mathcal{I}$-symbolic derivations along the sets I_1, I_2, O_1, O_2 is a symbolic derivation*

$$\mathcal{C} = (\mathcal{V}, \phi(\mathcal{S}_1 \cup \mathcal{S}_2), \mathcal{K}_1 \cup \mathcal{K}_2, (In_1 \cup In_2) \setminus (I_1 \cup I_2), (Out_1 \cup Out_2) \setminus (O_1 \cup O_2))$$

$\mathcal{V}$ is a sequence of variables indexed by $Ind = (Ind_1 \setminus \{i \mid x_i \in I_1\}) \cup (Ind_2 \setminus \{i \mid x_i \in I_2\})$, ordered by a linear extension of the transitive closure of the relation:

$$<_1 \cup <_2 \cup \{(u, v) \mid v = \phi(w) \text{ and } u <_1 w \text{ or } u <_2 w\}$$

and such that the variable of index i in $\mathcal{V}$ is equal to the variable of index i in $\mathcal{V}_1$ if $i \in Ind_1$, and to the variable of index i in $\mathcal{V}_2$ if $i \in Ind_2$.

A composition of two $\mathcal{I}$-symbolic derivations is also a $\mathcal{I}$-symbolic derivation, and we can compose an arbitrary number of $\mathcal{I}$-symbolic derivations in the same way.

Example 19 *Let us consider the protocol and the two symbolic derivations of Example 18. The normal execution of that protocol corresponds to the following composition of the two symbolic derivations of the previous example:* $I_1 = \{x_2^A\}$, $O_1 = \{y_1^A\}$, $I_2 = \{x_1^B\}$, *and* $O_2 = \{y_1^B\}$. *This composition imposes that* $x_2^A = y_1^B$ *which means that the second message received by* A *is the first message send by* B, *and* $x_1^B = y_1^A$ *which means that the first message received by* B *is the first message send by* A. *We recall that the first message received by* A *is empty.*

3.3.6 Ordered satisfiability problem

In chapter 2, we show how reduce the insecurity problem of cryptographic protocols to the satisfiability problem of constraint systems. Since this reduction is based on the fact that the capacity of the intruder attacking the protocol is represented by rules of the form $x_1, \ldots, x_n \rightarrow f(x_1, \ldots, x_n)$, and since, in this chapter, the intruder deduction rules have a different form, we can not apply this reduction. To this end, we introduce next another satisfiability problem, called *Ordered satisfiability problem*, to which we reduce the insecurity problem of cryptographic protocols using collision-vulnerable hash functions. The *Ordered satisfiability problem* has been initially defined in [72].

Definition 42 *Ordered* $\mathcal{I}$*-satisfiability problem Given an intruder deduction system* $\mathcal{I} = \langle \mathcal{F}, \mathcal{L}_\mathcal{I}, \mathcal{H} \rangle$, *the ordered* $\mathcal{I}$*-satisfiability problem is defined as follows:*

Input: *A* $\mathcal{I}$*-symbolic derivation* $\mathcal{C}$, *a set of ground terms* $\mathcal{K}_i$ *representing the intruder knowledge,* $X = Var(\mathcal{C})$, $C = all.Cons(\mathcal{C})$ *and a linear ordering* $\prec$ *on* $X \cup C$.

Output: SAT *if and only if there exists a* $\mathcal{I}$*-symbolic derivation* $\mathcal{C}_i = (\mathcal{V}_i, \mathcal{S}_i, \mathcal{K}_i, In_i, Out_i)$, *a closed composition* $\mathcal{C}_a$ *of* $\mathcal{C}_i$ *and* $\mathcal{C}$, *and a substitution* σ *such that* (1) $\sigma \models_\mathcal{I} \mathcal{C}_a$ *and* (2) *for all* $x \in X$ *and* $c \in C$, $x \prec c$ *implies* $c \notin all.Cons(x\sigma)$.

3.4 Symbolic formalisation of collision vulnerability property

We show in this section the symbolic formalisation of collision vulnerability property for hash functions. We recall that the collision vulnerability property of a hash function h means that it is computationally feasible to compute two distinct inputs x and x' with $h(x) = h(x')$ provided that x and x' are created at the same time and independently one of the other.

In order to construct a couple of messages having the same hash value, we introduce two new function symbols f, g, each of them with arity 4, and we

make use of the concatenation of messages, denoted by the function symbol ".", which is an associative and unary function symbol.

We define next the intruder systems $\mathcal{I}_{AU}$, $\mathcal{I}_f$, $\mathcal{I}_g$, $\mathcal{I}_{free}$ and $\mathcal{I}_h$.

We remark that our modelisation of collision vulnerability property does not take into account the time for finding collisions, which is significantly greater than the time necessary for other operations.

3.4.1 Intruder on words with free function symbols

We recall that the goal of this section is the symbolic specification of an intruder system that takes into account the collision vulnerability property for hash functions to mount attacks on a protocol.
We assume that the algorithm employed by the intruder to find collisions, *i.e.* a couple of messages with the same hash value, is as follows:

1. Start from two messages m and m';
2. The intruder splits both messages into two parts, m in m_1, m_2 and m' in m'_1, m'_2 such that $m = m_1.m_2$ and $m' = m'_1.m'_2$ with "." denotes the concatenation;
3. Then, the intruder computes two messages n and n' such that:

$$(HC): \ h(m_1.n.m_2) = h(m'_1.n'.m'_2)$$

Next, we represent the *collision vulnerability property of hash functions* as follows:

$$\forall\, m, m', \exists\, \hat{m}, \hat{m'}, \text{ such that } h(\hat{m}) = h(\hat{m'})$$

and more formally,

$$\forall\, m_1, m_2, m'_1, m'_2, \exists\, n, n', \text{ such that } h(m_1.n.m_2) = h(m'_1.n'.m'_2)$$

and hence, by *skolemisation*,

$$\forall\, m_1, m_2, m'_1, m'_2, h(m_1.f(m_1, m_2, m'_1, m'_2).m_2) = h(m'_1.g(m_1, m_2, m'_1, m'_2).m'_2),$$
where f and g are two function symbols with $arity\ 4$.

> In the remainder of this chapter, we represent *collision vulnerability property of hash functions* as follows: $\forall\, m, m'$, the intruder can compute two messages $g(m_1, m_2, m'_1, m'_2)$ and $f(m_1, m_2, m'_1, m'_2)$ such that: $(1)\ m = m_1.m_2$, and $(2)\ m' = m'_1.m'_2$, and $(3)\ (HC): h(m_1.g(m_1, m_2, m'_1, m'_2).m_2) = h(m'_1.f(m_1, m_2, m'_1, m'_2).m'_2)$.

We remark that the function symbols f and g denote the (complex) algorithm being used to find collisions starting from two different messages m and m'.

We first define the *intruder operating on words*, which we denote by $\mathcal{I}_{AU}$. This intruder is able to concatenate messages and extract *prefixes* and *suffixes*. We define $\mathcal{I}_{AU}$ as follows:

$$\mathcal{I}_{AU} = \langle \mathcal{F}_{AU}, \mathcal{L}_{\mathcal{I}AU}, \mathcal{H}_{AU} \rangle$$

with:

- $\mathcal{F}_{AU} = \{., \epsilon\}$; . denotes the concatenation, and ϵ denotes the empty word
- $\mathcal{L}_{\mathcal{I}AU} = \begin{cases} x, y \rightarrow x.y \\ x.y \rightarrow x \\ x.y \rightarrow y \\ \emptyset \rightarrow \epsilon \end{cases}$
- $\mathcal{H}_{AU} = \begin{cases} (x.y).z = x.(y.z) \\ x.\epsilon = x \\ \epsilon.x = x \end{cases}$

We then extend the $\mathcal{I}_{AU}$ intruder with the two function symbols f and g. We first define the intruder $\mathcal{I}_g$ who is an intruder able to compose messages using the function symbol g. $\mathcal{I}_g$ is given as follows:

$$\mathcal{I}_g = \langle \{g\}, \{x_1, x_2, x_3, x_4 \rightarrow g(x_1, x_2, x_3, x_4)\}, \emptyset \rangle$$

and similarly $\mathcal{I}_f$ is given as follows:

$$\mathcal{I}_f = \langle \{f\}, \{x_1, x_2, x_3, x_4 \rightarrow f(x_1, x_2, x_3, x_4)\}, \emptyset \rangle$$

We also define the intruder system $\mathcal{I}_{free}$ as the disjoint union of $\mathcal{I}_{AU}$, $\mathcal{I}_f$ and $\mathcal{I}_g$, and we have:

$$\mathcal{I}_{free} = \langle \mathcal{F}_{free}, \mathcal{L}_{\mathcal{I}free}, \mathcal{H}_{free} \rangle.$$

with:

- $\mathcal{F}_{free} = \{., \epsilon, f, g\}$;
- $\mathcal{L}_{\mathcal{I}free} = \begin{cases} x, y \rightarrow x.y \\ x.y \rightarrow x \\ x.y \rightarrow y \\ \emptyset \rightarrow \epsilon \\ x_1, x_2, x_3, x_4 \rightarrow f(x_1, x_2, x_3, x_4) \\ x_1, x_2, x_3, x_4 \rightarrow g(x_1, x_2, x_3, x_4) \end{cases}$
- $\mathcal{H}_{free} = \begin{cases} (x.y).z = x.(y.z) \\ x.\epsilon = x \\ \epsilon.x = x \end{cases}$

We remark that f and g do not appear in the equational theory $\mathcal{H}_{free}$: they are free in that equational theory.

3.4.2 Hash-colliding intruder

We gave in Section 3.4.1 the algorithm employed by the intruder to find collisions. A consequence of our model is that in order to build collisions starting from two messages m and m' the intruder must know these two messages. A side effect is that it is not possible to build three (or more) different messages with the same hash value by iterating the research for collisions.

In a more comprehensive model we might moreover want to model that collisions cannot always be found using attacks published in the literature, but instead that given a deadline, the probability p of success of an attack is strictly below 1. This would imply that the application of this rule by the intruder would, assuming independence of collision attacks, reduce the likelihood of the symbolic attack found. In this setting our model would account for attacks with a non-negligible probability of success as is shown in [34].

Leaving probabilities aside, we express intruder's application of a hash function in our setting by adding the rule $x \rightarrow h(x)$ to the deduction rules of the $\mathcal{I}_{free}$ intruder system. As a consequence, the previous description of the $\mathcal{I}_{free}$ intruder system enables us to model a collision-capable intruder. We describe $\mathcal{I}_h$ as follows:

$$\mathcal{I}_h = \langle \mathcal{F}_h, \mathcal{L}_{\mathcal{I}h}, \mathcal{H}_h \rangle$$

with:

- $\mathcal{F}_h = \mathcal{F}_{AU} \cup \{f, g\} \cup \{h\}$
- $\mathcal{L}_{\mathcal{I}h} = \mathcal{L}_{\mathcal{I}free} \cup \{x \rightarrow h(x)\}$
- $\mathcal{H}_h = \mathcal{H}_{free} \cup \{h(x_1.g(x_1, x_2, x_1', x_2').x_2) = h(x_1'.f(x_1, x_2, x_1', x_2').x_2')\}$

The function symbols f and g are not free anymore in this equational theory.

3.4.3 Properties on $\mathcal{I}_{free}$ and $\mathcal{I}_h$ intruder deduction systems

In Figure 3.1, we define the functions $Mode$ and $Sign$ on $\mathcal{F}_h$.

It is easy to see that the equational theories $\mathcal{H}_h$, $\mathcal{H}_f$, $\mathcal{H}_g$, $\mathcal{H}_{AU}$, and $\mathcal{H}_{free}$ are well-moded.

Lemma 13 *Let $R(\mathcal{H}_h)$ be the system resulting from the application of the Unfailing Knuth-Bendix procedure on $\mathcal{H}_h$, and let $l = r \in R(\mathcal{H}_h)$. If $l \in \mathcal{X}$ then $l \in Var(r)$.*

Figure 3.1 $Mode$ and $Sign$ on $\mathcal{I}_h$

$Mode$:

$Mode(\cdot, 1) = Mode(\cdot, 2) = 0$
$Mode(g, i) = Mode(f, i) = 0, \quad \forall i \in \{1, \ldots, 4\}$
$Mode(h, 1) = 0$

$Sign$:

$Sign(\cdot) = Sign(\epsilon) = Sign(f) = Sign(g) = 0$
$Sign(h) = 1$

PROOF.

Let $l = r \in R(\mathcal{H}_h)$ and suppose that $l \in \mathcal{X}$ and $l \notin Var(r)$. Let t_1 and t_2 be two different terms in $\mathcal{T}(\mathcal{F}_h, \mathcal{X})$ and let σ_1 and σ_2 be two substitutions such that $\sigma_1(l) = t_1,\ \sigma_2(l) = t_2$ and $\sigma_1(r) = \sigma_2(r)$. Then, $t_1 =_{\mathcal{H}_h} t_2$. We deduce that if $l \in \mathcal{X}$ and $l \notin Var(r)$ for a rule $l = r \in R(\mathcal{H}_h)$, all terms in $\mathcal{T}(\mathcal{F}_h, \mathcal{X})$ are equals modulo $\mathcal{H}_h$ which is impossible. Then for any rule $l = r \in R(\mathcal{H}_h)$, if $l \in \mathcal{X}$, we have $l \in Var(r)$. ■

Lemma 14 *Let $t \in \mathcal{T}(\mathcal{F}_h, \mathcal{X})$, we have:*

1. *If $t' \in Sub(t)$ and $Sign(t') = 1$ then $t' \in Sub_v(t)$;*
2. *If $Sign(t) = 1$ then $Sign((t)\downarrow) = 1$.*

PROOF.

1. Let $t \in \mathcal{T}(\mathcal{F}_h, \mathcal{X})$ and $t' \in Sub(t)$ such that $Sign(t') = 1$, let us prove that $t' \in Sub_v(t)$. Since $t' \in Sub(t)$, we have two cases:
   - $t' = t$, then $t' \in Sub_v(t)$.
   - t' is a strict subterm of t, then there exists an integer $p \geq 0$, an integer $i \geq 1$ such that $t_{|p.i} = t'$. We have $Sign(t_{|p.i}) = 1$ and by definition of $\mathcal{I}_h$ theory, $Mode(Top(t_{|p}), i) = 0$ then $Mode(Top(t_{|p}), i) \neq Sign(t_{|p.i})$. Thus t' is in ill-moded position in t, which implies that $t' \in Sub_v(t)$.
2. Let t be a ground term in $\mathcal{T}(\mathcal{F}_h)$ such that $Sign(t) = 1$. We have a finite sequence of rewritings starting from t leading to $(t)\downarrow$: $t \rightarrow_{R(\mathcal{H}_h)} \ldots \rightarrow_{R(\mathcal{H}_h)} t_i \rightarrow_{R(\mathcal{H}_h)} t_{i+1} \rightarrow_{R(\mathcal{H}_h)} \ldots \rightarrow_{R(\mathcal{H}_h)} (t)\downarrow$. Suppose that $Sign(t_i) = 1$, and let us prove that $Sign(t_{i+1}) = 1$. Let $l = r$ be the rule applied in the step i. By definition of rewriting, there exists a ground substitution σ, a position p such that $t_{i|p} = l\sigma$, $t_{i+1} = t_i[p \leftarrow r\sigma]$ and $l\sigma > r\sigma$. We have two cases:

- If $p \neq \varepsilon$, then $Top(t_{i+1}) = Top(t_i)$ and thus by $Sign(t_i) = 1$. We have $Sign(t_{i+1}) = 1$.
- If $p = \epsilon$, then $t_i = l\sigma$. Since $Sign(l\sigma) = 1$ and $l\sigma$ is ground, we have $Top(l\sigma) = h$. Since $l\sigma > r\sigma$ and by lemma 13, we have $l \notin \mathcal{X}$, and thus $l = h(l')$ for some $l' \in \mathcal{T}(\mathcal{F}_h, \mathcal{X})$. Since $R_{\mathcal{H}_h}$ is well-moded and $Sign(l) = 1$, we have $Sign(r) = 1$. We have three cases:
  - r is a non-free constant. Since the only non-free constant in $\mathcal{H}_h$ theory is ϵ and $Sign(\epsilon) = 0$, this case is impossible.
  - r is a variable. By lemma 13, we have $r \in Var(l)$, and thus $r \in Sub(l)$. Since l is well-moded in $\mathcal{H}_h$ theory, we haven $Sign(r) = 0$, which contradicts $Sign(t) = Sign(r)$.
  - $r = h(r')$ for $r' \in \mathcal{T}(\mathcal{F}_h, \mathcal{X})$. This implies that we have $r\sigma = h(r'\sigma)$, and therefor $Sign(r\sigma) = 1 = Sign(t_{i+1})$.

For all $i \in \{1, \ldots, n-1\}$, we have $Sign(t_i) = 1$ implies $Sign(t_{i+1}) = 1$, which proves the second point of the lemma.

■

Lemma 15 *Let t be a ground term in $\mathcal{F}_h$ with all its factors in normal form. We have: $Sub_v(t) \setminus \{\epsilon, t\} \subseteq Sub_v((t)\downarrow)$.*

PROOF.

Let t be a ground term in $\mathcal{F}_h$. There exists a finite sequence of rewritings starting from t leading to $(t)\downarrow$: $t \rightarrow_{R(\mathcal{H}_h)} \ldots \rightarrow_{R(\mathcal{H}_h)} t_i \rightarrow_{R(\mathcal{H}_h)} t_{i+1} \rightarrow_{R(\mathcal{H}_h)} \ldots \rightarrow_{R(\mathcal{H}_h)} (t)\downarrow$. Let us prove the lemma by contradiction and assume that $u \in Sub_v(t_i) \setminus \{\epsilon, t_i\}$ and $u \notin Sub_v(t_{i+1})$. Since $u \in Sub_v(t_i) \setminus \{\epsilon, t_i\}$, there exists an integer $q \geq 1$ such that $t_{i|q} = u$. Let $l = l'$ be the rule applied on t_i. There exists an integer $p \geq 0$, a ground substitution σ such that $t_{i|p} = l\sigma$ and $t_{i+1} = t_i[p \leftarrow l'\sigma]$ with $l\sigma > l'\sigma$.

- If $u \notin Sub_v(l\sigma)$ then $u \in Sub_v(t_{i+1})$.
- If $u \in Sub_v(l\sigma)$, by the fact that l is well-moded, u is in normal form and $u \neq \epsilon$, there exists $x \in Var(l)$ such that $u \in Sub_v(x\sigma)$. Since $Var(l) = Var(l')$, we have $u \in Sub_v(t_{i+1})$.

In the two cases, we lead to a contradiction with $u \notin Sub_v(t_{i+1})$. This concludes the proof of the lemma. ■

Lemma 16 *The intruder deduction system $\mathcal{I}_h$ satisfies the following criterion: for any set of ground terms in normal form E, if $E \rightarrow_{\mathcal{L}_{\mathcal{I}h}} E, r \rightarrow_{\mathcal{L}_{\mathcal{I}h}} E, r, t$ and $r \notin Sub_v(E, t) \cup C_{spec}$ then there is a set of ground terms in normal form F such that $E \rightarrow^*_{\mathcal{L}_{\mathcal{I}free}} F \rightarrow_{\mathcal{L}_{\mathcal{I}h}} F, t$.*

PROOF.

Let E be a set of ground terms in normal forms satisfying the following derivation: $E \rightarrow_{\mathcal{L}_{\mathcal{I}_h}} E, r \rightarrow_{\mathcal{L}_{\mathcal{I}_h}} E, r, t$ such that $r \notin Sub_v(E, t) \cup C_{spec}$. In order to prove that there exists a set of ground terms in normal form F such that $E \rightarrow^*_{\mathcal{L}_{\mathcal{I}free}} F \rightarrow_{\mathcal{L}_{\mathcal{I}_h}} F, t$, it suffices to prove that $E \rightarrow_{\mathcal{L}_{\mathcal{I}_h}} E, t$. We have $E \rightarrow_{\mathcal{L}_{\mathcal{I}_h}} E, r$ and the only $\mathcal{L}_{\mathcal{I}_h}$ rule is $x \rightarrow h(x)$. By definition, there exists a normal ground substitution σ such that $x\sigma \in E$ and $r = (h(x\sigma))\downarrow$. Since $Sign(h(x\sigma)) = 1$ by Lemma 14, we have $Sign(r) = 1$. Since $E, r \rightarrow_{\mathcal{L}_{\mathcal{I}_h}} E, r, t$, there exists a normal ground substitution σ' such that $x\sigma' \in E, r$ and $t = (h(x\sigma'))\downarrow$. If $x\sigma' = r$, we have $t = (h(r))\downarrow$. $h(r)$ is in normal form, since all its factors are in normal form and $r \in Sub_v(h(r)) \setminus \{h(r), \epsilon\}$, by Lemma 15 $r \in Sub_v(t)$, which contradicts the hypothesis $r \notin Sub_v(E, t) \cup C_{spec}$. By contradiction, we have $x\sigma' \in E$ and thus $E \rightarrow_{\mathcal{L}_{\mathcal{I}_h}} E, t$. ■

In the following lemma, $t =^1_{HC} t'$ denotes that there exists a one step rewriting between t and t' using (HC) equation.

Lemma 17 *Let $t_0, t, t' \in \mathcal{T}(\mathcal{F}_\langle, \mathcal{X})$ such that $t_0 =_{\mathcal{H}_{AU}} t =^1_{HC} t'$ and $t_0 = h(t_1 \cdot f(t_1, t_2, t_3, t_4) \cdot t_2)$. We have: $t' =_{\mathcal{H}_{AU}} h(t_3 \cdot g(t_1, t_2, t_3, t_4) \cdot t_4)$.*

PROOF.

Let $h(m_1 \cdot f/g(m_1, m_2, m_3, m_4) \cdot m_2) = h(m_3 \cdot g/f(m_1, m_2, m_3, m_4) \cdot m_4)$ be the ground instance of (HC) used between t and t'. Let us prove that $m_1 =_{\mathcal{H}_{AU}} t_1$. If $m_1 \neq_{\mathcal{H}_{AU}} t_1$, we have either m_1 is a prefix modulo $h_\cdot$ of t_1 or t_1 is a prefix modulo $\mathcal{H}_{AU}$ of m_1. Let us review these two cases:

- m_1 is a prefix modulo $\mathcal{H}_{AU}$ of t_1: then $t_1 = m_1 \cdot x$ and $x \neq_{\mathcal{H}_{AU}} \epsilon$, then $f/g(m_1, m_2, m_3, m_4) \in Sub(t_1)$, then $m_2 \in Sub(t_1)$. And we have $m_2 = y \cdot t_2$ with $y \neq_{\mathcal{H}_{AU}} \epsilon$, then $f(t_1, t_2, t_3, t_4) \in Sub(m_2)$ then $t_1 \in Sub(m_2)$. We conclude that t_1 is a strict subterm of m_2 and m_2 is a strict subterm of t_1 which is impossible.

- t_1 is a prefix modulo $\mathcal{H}_{AU}$ of m_1: by reasoning as above on t_2 which is a suffix of m_2, we can also prove that this case is impossible.

Thus we have $m_1 =_{\mathcal{H}_{AU}} t_1$, and thus $f/g(m_1, m_2, m_3, m_4) =_{\mathcal{H}_{AU}} f(t_1, t_2, t_3, t_4)$, that is $m_i =_{\mathcal{H}_{AU}} t_i$ for $i \in \{1, 2, 3, 4\}$ and $t' =_{\mathcal{H}_{AU}} h(t_3 \cdot g(t_1, t_2, t_3, t_4) \cdot t_4)$. ■

Lemma 18 *Let $h(m), h(m')$ be two pure terms and σ be ground substitution such that $\sigma \models_{\mathcal{H}_h} h(m) \stackrel{?}{=} h(m')$. Then one of the following holds:*

- $\sigma \models_{\mathcal{H}_{AU}} m \stackrel{?}{=} m'$
- $\sigma \models_{\mathcal{H}_{AU}} \left\{ m \stackrel{?}{=} x_1 \cdot g(x_1, x_2, y_1, y_2) \cdot x_2, m' \stackrel{?}{=} y_1 \cdot f(x_1, x_2, y_1, y_2) \cdot y_2 \right\}$

with x_1, x_2, y_1, y_2 new variables (modulo the commutativity of $\stackrel{?}{=}$).

PROOF.

Let $m_1, m_2, m_3 \in \mathcal{T}(\mathcal{F}_h, \mathcal{X})$ such that $h(m_1) =^1_{HC} h(m_2) =^1_{HC} h(m_3)$. If $m_1 =_{\mathcal{H}_{AU}} t_1 \cdot f(t_1, t_2, t_3, t_4) \cdot t_2$ then, by Lemma 17 we have

$$\begin{cases} m_2 =_{\mathcal{H}_{AU}} t_3 \cdot g(t_1, t_2, t_3, t_4) \cdot t_4 \\ m_3 =_{\mathcal{H}_{AU}} t_1 \cdot f(t_1, t_2, t_3, t_4) \cdot t_2 \end{cases}$$

Let $S_{m_1} = \{m|\ h(m) =_{\mathcal{H}_h} h(m_1)\}$ then, by Lemma 17 we have $S_{m_1} = \{m|\ m =_{\mathcal{H}_{AU}} m_1\} \cup \{m|\ m =_{\mathcal{H}_{AU}} t_3 \cdot g(t_1, t_2, t_3, t_4) \cdot t_4\}$.
We have $\sigma \models_{\mathcal{H}_h} h(m) \stackrel{?}{=} h(m')$ that is $h(m\sigma) =_{\mathcal{H}_h} h(m'\sigma)$, and thus $m'\sigma \in S_{m\sigma}$ which implies that either $m\sigma =_{\mathcal{H}_{AU}} m'\sigma$ and then $\sigma \models_{\mathcal{H}_{AU}} m \stackrel{?}{=} m'$ or $m\sigma =_{\mathcal{H}_{AU}} x_1\sigma \cdot f(x_1\sigma, x_2\sigma, y_1\sigma, y_2\sigma) \cdot x_2\sigma$ and $m'\sigma =_{\mathcal{H}_{AU}} y_1\sigma \cdot g(x_1\sigma, x_2\sigma, y_1\sigma, y_2\sigma) \cdot y_2\sigma$ and then $\sigma \models_{\mathcal{H}_{AU}} \left\{m \stackrel{?}{=} x_1 \cdot g(x_1, x_2, y_1, y_2) \cdot x_2, m' \stackrel{?}{=} y_1 \cdot f(x_1, x_2, y_1, y_2) \cdot y_2\right\}$. ■

Lemma 19 *Let E be a set of ground terms in normal form. If $E \rightarrow^*_{\mathcal{I}_{free}} f(t_1, t_2, t'_1, t'_2)$ and $f(t_1, t_2, t'_1, t'_2) \notin Sub(E)$ then $E \rightarrow^*_{\mathcal{I}_{free}} t_1, t_2, t'_1, t'_2$.*

PROOF.

We have $E \rightarrow^*_{\mathcal{I}_{free}} f(t_1, t_2, t'_1, t'_2)$ that is, there exists a finite sequence of rewritings starting from E leading to $f(t_1, t_2, t'_1, t'_2)$: $E \rightarrow_{\mathcal{I}_{free}} E_1 \rightarrow_{\mathcal{I}_{free}} \ldots \rightarrow_{\mathcal{I}_{free}} E_{n-1} \rightarrow_{\mathcal{I}_{AU}} E_{n-1}, f(t_1, t_2, t'_1, t'_2)$. By hypothesis, we have $f(t_1, t_2, t'_1, t'_2) \in Sub(E_n) \setminus Sub(E)$. Let E_i be the smallest set in the derivation such that $f(t_1, t_2, t'_1, t'_2) \in Sub(E_i) \setminus Sub(E_{i-1})$ [$i \geq 1$]. By $\mathcal{L}_{\mathcal{I}free}$ and $\mathcal{H}_{free}$, the rule applied in the step i of the derivation is $x_1, x_2, y_1, y_2 \rightarrow f(x_1, x_2, y_1, y_2)$. This implies that there exists a normal ground substitution σ such that $t_i = x_i\sigma$ and $t'_i = y_i\sigma$ for $i \in \{1, 2\}$ and $t_1, t_2, t'_1, t'_2 \in E_{i-1}$. We deduce that $E \rightarrow^*_{\mathcal{I}_{free}} t_1, t_2, t'_1, t'_2$. ■

Lemma 20 *Let $E \subseteq \mathcal{T}(\mathcal{F}_h)$ be a set of ground terms in normal form, and assume that E does not contain terms having the form $f(t_1, t_2, t_3, t_4)$ or $g(t_1, t_2, t_3, t_4)$ for some terms $t_1, \ldots, t_4$. Let m_1, m_2 be two terms in $\mathcal{T}(\mathcal{F}_h, \mathcal{X})$ such that $(h(m_1) \stackrel{?}{=} h(m_2))$ is $\mathcal{H}_h$-satisfiable. Let σ be a ground substitution which satisfies $(h(m_1) \stackrel{?}{=}_{\mathcal{H}_h} h(m_2))$. We have:*

$$E \rightarrow^*_{\mathcal{I}_{AU}} (m_1\sigma)\downarrow \text{ if and only if } E \rightarrow^*_{\mathcal{I}_{AU}} (m_2\sigma)\downarrow$$

PROOF.

By symmetry, it suffices to prove that $E \rightarrow^*_{\mathcal{I}_{AU}} (m_1\sigma)\downarrow$ implies $E \rightarrow^*_{\mathcal{I}_{AU}} (m_2\sigma)\downarrow$. Since $\sigma \models_{\mathcal{H}_h} (h(m_1) \stackrel{?}{=} h(m_2))$, by Lemma 18 we have two cases:

- If $\sigma \models_{\mathcal{H}_{AU}} m_1 \stackrel{?}{=} m_2$ then the result is obvious.

- If $\sigma \models_{\mathcal{H}_{AU}} \left\{ m \stackrel{?}{=} x_1 \cdot g(x_1, x_2, y_1, y_2) \cdot x_2, m' \stackrel{?}{=} y_1 \cdot f(x_1, x_2, y_1, y_2) \cdot y_2 \right\}$ then

$$\begin{cases} m_1\sigma =_{\mathcal{H}_{AU}} x_1\sigma \cdot f(x_1\sigma, x_2\sigma, y_1\sigma, y_2\sigma) \cdot x_2\sigma \\ m_2\sigma =_{\mathcal{H}_{AU}} y_1\sigma \cdot g(x_1\sigma, x_2\sigma, y_1\sigma, y_2\sigma) \cdot y_2\sigma \end{cases}$$

  Since $E \rightarrow^*_{\mathcal{I}_{AU}} (m_1\sigma)\downarrow$, we have $E \rightarrow^*_{\mathcal{I}_{AU}} (x_1\sigma \cdot f(x_1\sigma, x_2\sigma, y_1\sigma, y_2\sigma) \cdot x_2\sigma)\downarrow$ and thus, $E \rightarrow^*_{\mathcal{I}_{AU}} (x_1\sigma)\downarrow \cdot f((x_1\sigma)\downarrow, (x_2\sigma)\downarrow, (y_1\sigma)\downarrow, (y_2\sigma)\downarrow) \cdot (x_2\sigma)\downarrow$ which implies that $E \rightarrow^*_{\mathcal{I}_{AU}} (x_1\sigma)\downarrow, (x_2\sigma)\downarrow, f((x_1\sigma)\downarrow, (x_2\sigma)\downarrow, (y_1\sigma)\downarrow, (y_2\sigma)\downarrow)$.
  By hypothesis, $f((x_1\sigma)\downarrow, (x_2\sigma)\downarrow, (y_1\sigma)\downarrow, (y_2\sigma)\downarrow) \notin SubE$, Lemma 19 implies $E \rightarrow^*_{\mathcal{I}_{AU}} (x_1\sigma)\downarrow, (x_2\sigma)\downarrow, (y_1\sigma)\downarrow, (y_2\sigma)\downarrow$ and thus $E \rightarrow^*_{\mathcal{I}_{AU}} g((x_1\sigma)\downarrow, (x_2\sigma)\downarrow, (y_1\sigma)\downarrow, (y_2\sigma)\downarrow)$ which implies that $E \rightarrow^*_{\mathcal{I}_{AU}} (y_1\sigma)\downarrow \cdot g((x_1\sigma)\downarrow, (x_2\sigma)\downarrow, (y_1\sigma)\downarrow, (y_2\sigma)\downarrow) \cdot (y_2\sigma)\downarrow$. We conclude that $E \rightarrow^*_{\mathcal{I}_{AU}} (m_2\sigma)\downarrow$.

■

3.5 Decidability results

We show next the decidability of respectively ordered $\mathcal{I}_{AU}$, ordered $\mathcal{I}_f$, ordered $\mathcal{I}_g$, ordered $\mathcal{I}_{free}$-satisfiability problems, and we conjecture the decidability of the ordered $\mathcal{I}_h$-satisfiability problem.

3.5.1 Decidability of ordered $\mathcal{I}_{AU}$-satisfiability problem

We state below some basic facts on $\mathcal{I}_{AU} = \langle \{.\}, \mathcal{L}_{\mathcal{I}AU}, \mathcal{H}_{AU} \rangle$.

Lemma 21 *Let E and t be respectively a set of terms and a term in normal form with respect to the equational theory $\mathcal{H}_{AU}$. We have:*

1. $all.Cons(E) \subseteq \overline{E}^{\mathcal{I}_{AU}}$,
2. $E \subseteq \overline{all.Cons(E)}^{\mathcal{I}_{AU}}$,
3. $\overline{E}^{\mathcal{I}_{AU}} = \overline{all.Cons(E)}^{\mathcal{I}_{AU}}$,
4. $t \in \overline{E}^{\mathcal{I}_{AU}}$ *if and only if* $all.Cons(t) \subseteq all.Cons(E)$.

PROOF.

Let E and t be respectively a set of ground terms and a ground term in normal form with respect to the equational theory $\mathcal{H}_{AU}$.

1. Let $c \in all.Cons(E)$ be a constant, and let e be a term in E such that $c \in all.Cons(e)$. We have therefore $e = e_1.c.e_2$. By associativity of ., we have $e = e_1.c.e_2 = (e_1.c).e_2 = e_1.(c.e_2)$. Thus $E \rightarrow_{\mathcal{I}_{AU}} E, (e_1.c) \rightarrow_{\mathcal{I}_{AU}} E, (e_1.c), c$. This implies $all.Cons(E) \subseteq \overline{E}^{\mathcal{I}_{AU}}$.

2. By definition of $\mathcal{I}_{AU}$, it is easy to see that $E \subseteq \overline{all.Cons(E)}^{\mathcal{I}_{AU}}$.

3. (1) and (2) implies easily (3).

4. We have $t \in \overline{E}^{\mathcal{I}_{AU}}$ is equivalent to $\overline{E}^{\mathcal{I}_{AU}} = \overline{(E,t)}^{\mathcal{I}_{AU}}$, which is equivalent to $\overline{all.Cons(E)}^{\mathcal{I}_{AU}} = \overline{all.Cons(E,t)}^{\mathcal{I}_{AU}}$ by (3). By contradiction, assume that $t \in \overline{E}^{\mathcal{I}_{AU}}$ and there is a constant $c \in all.Cons(t) \setminus all.Cons(E)$. $c \notin all.Cons(E)$ implies that $c \notin \overline{all.Cons(E)}^{\mathcal{I}_{AU}}$, and $c \in all.Cons(t)$ implies that $c \in \overline{all.Cons(E,t)}^{\mathcal{I}_{AU}}$. This contradicts the equality $\overline{all.Cons(E)}^{\mathcal{I}_{AU}} = \overline{all.Cons(E,t)}^{\mathcal{I}_{AU}}$, and hence we deduce that $all.Cons(t) \subseteq all.Cons(E)$. Now, we prove the converse. We have $all.Cons(t) \subseteq all.Cons(E)$, this implies $all.Cons(t) \subseteq \overline{all.Cons(E)}^{\mathcal{I}_{AU}}$. By (2), we have $t \in \overline{all.Cons(t)}^{\mathcal{I}_{AU}}$, and hence, $t \in \overline{all.Cons(E)}^{\mathcal{I}_{AU}}$. By (3), we conclude that $t \in \overline{E}^{\mathcal{I}_{AU}}$.

■

The consequence of Lemma 21 is that only the set of constants appearing or not in the initial knowledge and the goal of a supposed derivation are relevant to decide its feasibility. This has the important implication, with respect to decidability, that it is not necessary to know the exact instance of intruder's knowledge (his initial knowledge and the messages in the output of the symbolic derivation up to this point) and the goal (the next input message of the symbolic derivation) to decide whether a derivation exists. It suffices to know the guessable sets of constants of the knowledge and of the goal.

We give in Figure 3.2 the algorithm that solve ordered $\mathcal{I}_{AU}$-satisfiability problem and then, show its correctness, completeness and termination.

The definition of satisfiability of symbolic derivations allows us to conclude automatically the completeness of the algorithm:

Lemma 22 *(Completeness) The algorithm described in Figure 3.2 is complete.*

PROOF.

Let $\mathcal{C} = (\mathcal{V}, \mathcal{S}, \mathcal{K}, In, Out)$ be a $\mathcal{I}_{AU}$ symbolic derivation, $\mathcal{K}_{\mathcal{I}}$ be the initial intruder knowledge, and $\prec$ be a linear ordering on the variables and constants of $\mathcal{C}$. Assume that $\mathcal{C}$ is satisfiable, this implies, by definition, that there exists a $\mathcal{I}_{AU}$-symbolic derivation $\mathcal{C}_{\mathcal{I}} = (\mathcal{V}_{\mathcal{I}}, \mathcal{S}_{\mathcal{I}}, \mathcal{K}_{\mathcal{I}}, In_{\mathcal{I}}, Out_{\mathcal{I}})$, a closed composition $\mathcal{C}_a$ of $\mathcal{C}_{\mathcal{I}}$ and $\mathcal{C}$, and a substitution σ such that (1) $\sigma \models_{\mathcal{I}_{AU}} \mathcal{C}_a$ and (2) for all x variable in $\mathcal{C}$ and c constant in $\mathcal{C}$, $x \prec c$ implies $c \notin all.Cons(x\sigma)$. Hence, the intruder $\mathcal{I}$ receives all messages sent by the protocol participants, and every message they receive is send by the intruder. We have that $x\sigma$ is defined for all $x \in \mathcal{V}$, we let $P_x = all.Cons(x\sigma)$. Furthermore, we have that for all $x \in In$, $(x\sigma)\downarrow \in \overline{\mathcal{K}'_{\mathcal{I}}\sigma}^{\mathcal{I}_{AU}}$ with $\mathcal{K}'_{\mathcal{I}} = \mathcal{K}_{\mathcal{I}} \cup \{x' \mid x' \in Out \text{ and } x' \prec x\}$, and by Lemma 21, we have

Figure 3.2 Ordered satisfiability for $Intruder_{.}$ deduction system

Input

- A $\mathcal{I}_{AU}$ symbolic derivation $\mathcal{C} = (\mathcal{V}, \mathcal{S}, \mathcal{K}, In, Out)$;
- a finite set of ground terms $\mathcal{K}_{\mathcal{I}}$ representing the initial knowledge;
- an ordering $\prec$ on $Var(\mathcal{C}) \cup all.Cons(\mathcal{C})$, $\prec$ is compatible with $<$ defined on Ind, *i.e.* $i < j$ implies $x_i \prec x_j$.

Step 1:
For each variable $x \in \mathcal{V}$, guess the set of constants $P_x = \{a_1, \ldots, a_k\}$ that may occur in the solution, where $a_i \prec x$ for $i \in \{1, \ldots, k\}$;

Step 2:
Check the satisfiability of $\mathcal{S}$ with these ordering constraints.

Step 3:
Check for each variable $x \in In$, that $P_x \subseteq all.Cons(\mathcal{K}_{\mathcal{I}}) \cup \bigcup_{x' \in Out, x' < x} P_{x'}$.

Step 4:
If both checks are successful return Sat else Fail.

$all.Cons(x\sigma) \subseteq all.Cons(\mathcal{K}'_{\mathcal{I}}) = all.Cons(\mathcal{K}_{\mathcal{I}}) \cup \bigcup_{x' \in Out\,,\, x' \prec x} all.Cons(x'\sigma) = all.Cons(\mathcal{K}_{\mathcal{I}}) \cup \bigcup_{x' \in Out\,,\, x' \prec x} P_{x'}$. Given σ, it is easy to see that the algorithm of Figure 3.2 outputs sat, and hence, we conclude the proof. ■

Lemma 23 *(Correctness) The algorithm described in Figure 3.2 is correct.*

PROOF.

Let $\mathcal{C} = (\mathcal{V}, \mathcal{S}, \mathcal{K}, In, Out)$ be a $\mathcal{I}_{AU}$ symbolic derivation, $\mathcal{K}_{\mathcal{I}}$ be the initial intruder knowledge, and $\prec$ be a linear ordering on the variable and constants of $\mathcal{C}$. Assume that the algorithm in Figure 3.2 outputs sat. By the description of the algorithm and by Lemma 21, we deduce that there exists a $\mathcal{I}_{AU}$-symbolic derivation $\mathcal{C}_{\mathcal{I}} = (\mathcal{V}_{\mathcal{I}}, \mathcal{S}_{\mathcal{I}}, \mathcal{K}_{\mathcal{I}}, In_{\mathcal{I}}, Out_{\mathcal{I}})$, a closed composition $\mathcal{C}_a$ of $\mathcal{C}_{\mathcal{I}}$ and $\mathcal{C}$, and a substitution σ such that (1) $\sigma \models_{\mathcal{I}_{AU}} \mathcal{C}_a$ and (2) for all x variable in $\mathcal{C}$ and c constant in $\mathcal{C}$, $x \prec c$ implies $c \notin all.Cons(x\sigma)$. Hence, we conclude the proof. ■

Lemma 24 *(Termination) Let $\mathcal{C} = (\mathcal{V}, \mathcal{S}, \mathcal{K}, In, Out)$ be an $\mathcal{I}_{AU}$ symbolic derivation, $\mathcal{K}_{\mathcal{I}}$ be the initial intruder knowledge, and $\prec$ be a linear ordering on the variable and constants of $\mathcal{C}$. The algorithm described in Figure 3.2 terminates.*

PROOF.

Since $\mathcal{V}$ is finite (by definition) and the unification modulo $\mathcal{H}_{AU}$ is with constant restrictions decidable [186], it is easy to conclude that the algorithm terminates. ■

By Lemmas 22, 23 and 24, we conclude the following theorem:

Theorem 2 *The ordered $\mathcal{I}_{AU}$ satisfiability problem is decidable.*

Related results. The decidability result obtained in this section is interesting in its own right as it is the first decidability result that we are aware of for an intruder system for which unification is infinitary, and that permits to consider an associative concatenation of messages instead of their pairing.

A related decidability result has been obtained by Y. Chevalier *et al.* [72] for an intruder on multi-set of terms, that is an intruder with an associative-commutative-unit equational theory. In [58], S. Bursuc *et al.* obtain another decidability result in presence of associative and commutative symbols under some restrictions. This result is more recent than ours and does not cover the case of [72].

3.5.2 Decidability of ordered $\mathcal{I}_f$ and $\mathcal{I}_g$ satisfiability problems

Theorem 3 *The ordered $\mathcal{I}_f$ satisfiability problem is decidable.*

PROOF.

We have $\mathcal{I}_f = \langle \{f\}, \{x_1, x_2, x_3, x_4 \rightarrow f(x_1, x_2, x_3, x_4)\}, \emptyset \rangle$. It is easy to see that $\mathcal{I}_f$ is local (Definition 38). Since the equational theory in $\mathcal{I}_f$ is empty, by [70], we deduce that given E, t with E a set of ground terms and t a ground term, $t \in \overline{E}^{\mathcal{I}_f}$ is decidable, and hence, we can see easily that the ordered $\mathcal{I}_f$ satisfiability problem is decidable. ■

Following the same reasoning as before, we can deduce the following theorem.

Theorem 4 *The ordered $\mathcal{I}_g$ satisfiability problem is decidable.*

3.5.3 Decidability of ordered $\mathcal{I}_{free}$ satisfiability problem

Theorem 5 *The ordered $\mathcal{I}_{free}$ satisfiability problem is decidable.*

PROOF.

$\mathcal{I}_{free}$ is the disjoint union of $\mathcal{I}_{AU}$, $\mathcal{I}_g$ and $\mathcal{I}_f$ intruder deduction systems. We have that ordered-$\mathcal{I}_{AU}$ (respectively $\mathcal{I}_g$ and $\mathcal{I}_f$) satisfiability problem, is decidable (respectively Theorem 2, Theorem 4, and Theorem 3). The result obtained in [72] prove that the ordered satisfiability problem for the disjoint union of decidable intruder deduction systems is also decidable. Thus ordered $\mathcal{I}_{free}$ satisfiability problem is decidable. ■

3.5.4 Decidability of ordered $\mathcal{I}_h$-satisfiability problem

Following the results obtained in [74], and by Lemmas 13, 14, 15, 16, 17, 18, 19, and 20, we conjecture that we can reduce the ordered $\mathcal{I}_h$ satisfiability problem to the ordered $\mathcal{I}_{free}$ satisfiability problem. And hence, we conjecture that the ordered $\mathcal{I}_h$ satisfiability problem is decidable.

3.6 Conclusions

We have analysed here the class of cryptographic protocols that use collision vulnerable hash functions. In order to model such hash functions we have introduced new symbols to denote the ability to create messages with the same hash value. This introduction amounts to the skolemisation of the equational property describing the existence of collisions. We believe that this construction can be extended to model the more complex and game-based properties that appear when relating a symbolic and a concrete model of cryptographic primitives. It is well-known that hash functions are widely used in the digital signature schemes which is the topic of the next chapter.

Chapter 4

Analysis of protocols with vulnerable digital signature schemes

The idea of *digital signature schemes* first appeared in *W. Diffie and M.E. Hellman's* seminal paper [101]. Digital signatures have many applications in information security, including authentication, data integrity, and non-repudiation. In [117], the authors showed the different flaws of digital signature schemes. Furthermore, the authors defined what is *a secure digital signature scheme*: a digital signature scheme is considered to be secure if it is existential unforgeable against adaptative chosen message attacks. Unfortunately, while this security notion is adequate to the single-user setting, it is not adequate for the multi-user setting [152]. In this chapter, we are interested by the *security of signature schemes in the multi-user setting*, and to this end, we consider two properties of digital signature schemes: the *constructive exclusive ownership vulnerability property* and the *destructive exclusive ownership vulnerability property*. We show the decidability of the insecurity problem for two classes of cryptographic protocols where the signature schemes employed have respectively these two properties. This result has been published in the proceedings of *FSTTCS 2007* conference [66].

Outline. In Section 4.1 we present the signature schemes, in Section 4.2 we define the *constructive exclusive ownership vulnerability property* (also called *duplicate-signature key selection property*), and the *destructive exclusive ownership vulnerability property* is defined in Section 4.3. In Section 4.4, we define the symbolic model of these two perperties (Section 4.4.1 and Section 4.4.2), we prove the decidability of $\mathcal{H}_{DSKS}$ and $\mathcal{H}_{DEO}$ unifiability problems (Section 4.4.3), and we prove the decidability of $\mathcal{I}_{DSKS}$ and $\mathcal{I}_{DEO}$ reachability problems (Section 4.4.5).

4.1 Signature schemes

4.1.1 Definition of signature schemes

The idea of *digital signature* first appeared in *W. Diffie and M.E. Hellman's* seminal paper [101]. A *digital signature* of a message is a number dependent on some secret known only to the signer, and, additionally, on the content of the message being signed. Signatures must be verifiable; if a dispute arises as to whether a party signed a document (caused by either a lying signer trying to repudiate a signature it did create, or a fraudulent claimant), an unbiased third party should be able to resolve the matter equitably, without requiring access to the signers secret information (secret key). To this end, each agent A possesses a pair of keys: a *public key*, pk_A and a *secret key*, sk_A. The agent A uses his secret key sk_A to sign messages, and pk_A is the key used by the other agents to verify A's signatures. The secret key of an agent is called *signing key*, and his public key is called *verifying key*.

Formally, we define a *key generation algorithm* G to be an algorithm that takes an agent's name A and a random number as arguments and returns a pair of public and secret keys, respectively denoted by $Pk(A)$ and $Sk(A)$, corresponding to that agent. We define a *digital signature generation algorithm (or signature generation algorithm)*, denoted by sig, to be an algorithm that takes a secret key (also called *signing key*), sk, and a message, m, as inputs, and generates a message, $sig(m, sk)$, representing the digital signature of m using the secret key sk. We define a *digital signature verification algorithm (or verification algorithm)*, denoted by ver, to be an algorithm testing whether a message s is a valid signature of a message m using the public key pk (also called *verifying key*).

Definition 43 *(digital signature scheme) A digital signature scheme (or simply a signature scheme) is defined by three algorithms: the signature generation algorithm "sig", the verification algorithm "ver", the key generation algorithm "G".*

Example 20 *(Example of signature schemes) We give here a classical digital signature scheme: the signature scheme proposed by W. Diffie and M. E. Hellman [101]. To create a signature scheme, W. Diffie and M. E. Hellman proposed to use a "trap-door function" f (informally, a "trap-door function" f is a function for which it is easy to evaluate $f(x)$ for any argument x but for which, given only $f(x)$, it is computationally infeasible to find any y with $f(y) = f(x)$ without the secret "trap-door" information). In their signature scheme, an agent A publishes the "trap-door function" f and anyone can validate any $A's$ signature by checking that $f(signature) = message$. Only the agent A possesses the "trap-door" information allowing him to invert f, and compute a signature y such that $f(y) = x$ where x is the message to sign.*

4.1.2 Classification of signature schemes

There exists two general classes of digital signature schemes, which can be briefly summarised as follows:

- *Digital signature schemes with appendix* require the message that has been signed (also called the *original message*) as input to the verification algorithm.
- *Digital signature schemes with message recovery* do not require the original message as input to the verification algorithm. In this case, the original message is recovered from the signature itself.

In what follows, we make use of $\mathcal{A}$ to denote the set of agents, K to denote the set of keys, $\mathcal{R}$ to denote a set of numbers, $\mathcal{M}$ to denote the message space, that is, the set of messages to which the signature generation algorithm may be applied, and $\mathcal{S}$ to denote the signature space, that is the set of messages to which belong the signatures.

Digital signature schemes with appendix

Digital signature schemes with appendix denote the schemes which require the original message as input to the verification algorithm. Examples of mechanisms providing digital signature schemes with appendix are the DSA [3], ElGamal [116], and Schnorr [185] signature schemes. We describe below the three algorithms: the *signature generation*, the *verification*, and the *key generation* algorithms.

Key generation algorithm.
Each agent A creates a pair of keys, a secret key and a correspondant public key. The agent A will use his secret key to sign messages, and his public key will be used by other agents for verifying $A's$ signatures. The key generation algorithm is defined as follows:

$$G: \ \mathcal{A} \times \mathcal{R} \rightarrow K * K$$

We remark that the second argument of the function G represents a randomly generated number, the use of this random number allows any agent A to have a different pair of keys for each session, and that using the same key generation function. Furthermore, we remark that the two functions Sk and Pk denote the two parts of the key generation algorithm G, and we denote respectively by $Sk(A)$ and $Pk(A)$ the secret and public keys of $A \in \mathcal{A}$;

Signature generation and verification algorithms.

An agent A produces a signature s for a message m, which can later be verified by any agent B. The signature generation algorithm is defined as follows:

$$sig: \mathcal{M} \times K \rightarrow \mathcal{S}$$

The verification algorithm is defined as follows:

$$ver: \mathcal{M} \times \mathcal{S} \times K \rightarrow \{1\}$$

such that for any m, s and pk we have: $ver(m, s, pk) = \{ 1, if\ s = sig(m, sk),\ sk = Sk(A),\ and\ pk = Pk(A)\ for\ some\ agent\ A$
We assume each time an agent generates a pair of secret and public keys, (sk, pk), the secret key sk *matches* the public key pk, that is, for any message m, we have $ver(m, sig(m, sk), pk) = 1$.

Digital signature schemes with message recovery

Digital signature schemes with message recovery denote the schemes for which a priori knowledge of the message that has been signed is not required for the verification algorithm. In this case, the original message can be recovered from the signature itself. Examples of schemes providing digital signatures with message recovery are RSA [174], Rabin [171], and Nyberg-Rueppel [18] public-key signature schemes. We describe below the three algorithms: the *signature generation*, the *verification*, the *public key generation*, and the *key generation* algorithms.

Key generation algorithms.

Each agent A creates a pair of keys, a secret key and a correspondant public key. The agent A will use his secret key to sign messages, and his public key will be used by other agents for verifying $A's$ signatures. The key generation algorithm is defined as follows:

$$G: \mathcal{A} \times \mathcal{R} \rightarrow K * K$$

We remark that the second argument of the function G represents a randomly generated number, the use of this random number allows any agent A to have a different pair of keys for each session, and that using the same key generation function. Furthermore, we remark that the two functions Sk and Pk denote the two parts of the key generation algorithm G, and we denote respectively by $Sk(A)$ and $Pk(A)$ the secret and public keys of $A \in \mathcal{A}$;

Signature generation and verification algorithms.
An agent A produces a signature s for a message m, which can later be verified by any agent B. The signature generation algorithm is defined as follows:

$$sig: \mathcal{M} \times K \rightarrow \mathcal{S}$$

The verification algorithm is defined as follows:

$$ver: \mathcal{S} \times K \rightarrow \mathcal{M}$$

such that for any s and pk we have: $ver(s, pk) = \{ m, if\ s = sig(m, sk),\ sk = Sk(A),\ and\ pk = Pk(A)\ for\ some\ agent\ A$
We assume each time an agent generates a pair of secret and public keys, (sk, pk), the secret key sk *matches* the public key pk, that is, for any message m, we have $ver(sig(m, sk), pk) = m$.

In the Handbook of Applied Cryptography, at Chapter 11 [153], it is shown that any digital signature scheme with message recovery can be turned into a digital signature scheme with appendix.

4.1.3 Attacks and breaks on signature schemes

Description of the attacks. There are two basic attacks against digital signature schemes described as follows [117, 153]:

- *key-only attacks:* in these attacks, an intruder knows only the signers public key.
- *message attacks:* here an intruder is able to examine signatures corresponding either to known or chosen messages before his attempt to break the scheme. We identify the following four kinds of message attacks, which are characterised by how the messages whose signatures the intruder sees are chosen.
  - *Known message attack:* an intruder has signatures for a set of messages which are known to the intruder but not chosen by him.
  - *Chosen message attack:* an intruder obtains from an agent A valid signatures for a chosen list of messages before attempting to break the signature scheme. This attack is non-adaptive in the sense that messages are chosen before any signatures are seen. Furthermore, these messages are independent of $A's$ public key.

- *Directed chosen message attack:* this attack is similar to chosen message attack, except that the list of messages for which the intruder obtains $A's$ signatures may be created after seeing $A's$ public key but before any signatures are seen.
- *Adaptive chosen message attack:* an intruder is allowed to use the signer A as an oracle, that is, not only the intruder may request from the agent A signatures of messages which depend on As public key but he may also request signatures of messages which depend additionally on previously obtained signatures.

The above attacks are listed in order of increasing severity, with the *adaptative chosen message attack* being the most severe natural attack an intruder can mount.

Description of the breaks. The different notions of *break a signature scheme* we give below were initially introduced in [117, 153].

We say that an intruder *forges a signature* if he is able to produce a *new signature* which will be accepted as one of some other agent.

One might say that the intruder has *broken* a signature scheme if his attack allows him to do one of the following with a non-negligible probability:

- *Total break:* an intruder is able to compute the secret key information of an agent.
- *Universal forgery:* an intruder is able to find an efficient signing algorithm functionally equivalent to an agent's signing algorithm.
- *Selective forgery:* an intruder is able to forge a signature for a particular message or class of messages chosen a priori. Creating the signature does not directly involve the legitimate signer.
- *Existential forgery:* an intruder is able to forge a signature for at least one message. The intruder has little or no control over the message whose signature is obtained, and the legitimate signer may be involved in the deception.

The kinds of "breaks" are listed above in order of decreasing severity, the least the intruder might hope for is to succeed with an existential forgery. We say that a signature scheme is respectively *totally breakable*, *universally forgeable*, *selectively forgeable*, or *existentially forgeable* if it is breakable in one of the above senses.

From now on, we are interested only by signature schemes with appendix, and for simplicity, we write "signature schemes" instead of "signature schemes with appendix".

4.2 Duplicate-signature key selection (DSKS) property

4.2.1 Description of DSKS property

Digital signature schemes are primarily employed to authenticate documents, *i.e.* if a public key is known to be associated with an agent A, then a valid signature of some data according to $A's$ public key proves that the agent A endorses the content of the signed message, and approves the association between the message that has been signed and his public key. This assumes that no other agent has access to the secret key. But digital signature schemes can also be employed to authenticate the origin of a message, *i.e.* ensure that only one person could have signed it, or more formally, given a signed message s and $A's$ public key pk, if s is a valid signature of m according to pk then s has been produced using $A's$ secret key.

The analysis of digital signature algorithms has however focused on the former authentication property. Formally speaking, the yardstick security notion for assessing the robustness of a digital signature scheme is the existential unforgeability against adaptative chosen-message attacks [117, 93]. This notion states that, given a pair of signing and verifying keys, it is infeasible for someone ignorant of the signing key to forge a signature for at least one message, and this even when messages devised by the attacker are signed beforehand. The security goal provided by this property is the impossibility (within given computing bounds) to impersonate a legitimate user (*i.e.* one that does not reveal its signature key) when signing a message.

We note that this robustness does not address the issue of the identification of the origin of a message. However, this latter concept is also pertaining to digital signatures when they are employed in a non-repudiation protocol. While one would not differentiate the two properties (message authentication and origin authentication) at first glance, they are different since the first authentication property requires the existence of the participation of the signer in the creation of the message, while the latter mandates the unicity of a possible creator of a message.

The two notions of message authentication and origin authentication collapse in the *single-user* setting when there exists only one pair of signing and verifying keys (or only one legitimate agent who can sign data, authenticate data). They may however be different in the *multi-user* setting.

There has been some works defining and proving security of some of the cryptographic primitives in the *multi-user* setting. *M. Bellare and P. Rogaway* [37], *S. Black-Wilson et al.* [43], *R. Canetti and H. Krawczyk* [59] and *V. Shoup* [189] gave security definitions for symmetric key entity authentication schemes and key transport schemes in the *multi-user* setting, M. Bellare *et al.* [36] presented security definitions for public key encryption in the *multi-user* setting, and more

recently, *A. Menezes and N. Smart* [152] gave security definitions for signature schemes in the *multi-user* setting.

In this chapter, we are interested by the origin authentication goal of the signature schemes with appendix in the *multi-user* setting, and we consider in this section the property called *duplicate-signature key selection property* while the *destructive exclusive ownership property* is considered in Section 4.3. *Duplicate-signature key selection property* (also called *key substitution property*) can be informally described as follow:

Definition 44 *(Duplicate-signature key selection property) Let $S = (G, sig, ver)$ be a digital signature scheme. S possesses the duplicate-signature key selection property (or key substitution property) if, knowing $A's$ public key, where A is an agent, and $A's$ signature s_A on a message m, the intruder is able to construct with a non-negligible probability a new pair of keys $(Sk_{\mathcal{I}}, Pk_{\mathcal{I}})$ such that $Sk_{\mathcal{I}}$ matches $Pk_{\mathcal{I}}$, and s_A is also $\mathcal{I}'s$ signature on the message m, that is $ver(m, s_A, Pk_{\mathcal{I}}) = 1$.*

This property was first introduced by S. Blake-Wilson and A. Menezes [45]. In [45], *S. Blake-Wilson and A. Menezes* showed the existence of a flaw on a variant of the Station-to-Station protocol [102] and which is due to the fact that the used signature scheme has the *duplicate-signature key selection property*. That flaw is given by the possibility to confuse a participant into thinking he shares a key with another person than the actual one. This *duplicate-signature key selection property* was also discussed in [152], and in [170].

In [45], the authors showed that, in certain circumstances, RSA [174], DSA [3], ECDSA [5] and ElGamal [116] signature schemes have *duplicate-key substitution property*. And in [170], the authors showed that DSS [6] also has this property, an in [27] the authors showed that Schnorr signature scheme has this property.

4.2.2 DSKS property in practice

We present here the *unknown-key share* attack, given by *J. Baek et al.* [27], on the *Key Agreement protocol* proposed by *Hirose* and *Yoshida* (also called *KAP-HY protocol*) in [119]. This attack exploits the duplicate-signature key selection property of signature schemes.

An *unknown key share (UKS)* attack on a key agreement protocol is an attack whereby an agent A finishes believing she shares the key with B, and although this is in fact the case, B mistakenly believes that he shares the key with an agent $E \neq A$ [27, 45, 44, 102].

Presentation of the KAP-HY protocol. The signature scheme used in this protocol is the Redundant signature scheme, which is a variant of Schnorr signature scheme. In what follows and for simplicity, we denote by $sig_A(m)$ the signature

of a message m by an agent A. Abstracting the details of the Diffie-Hellman key construction with messages u_A and u_B, and of the signature scheme, the protocol reads as follows:

$$\begin{cases} 1 \;\; A \Rightarrow B: \;\; u_A, id_A \\ 2 \;\; B \Rightarrow A: \;\; u_B, sig_B(u_A), id_B \\ 3 \;\; A \Rightarrow B: \;\; sig_A(u_B) \end{cases}$$

In the description given above, id_A (respectively id_B) denotes the identification of the agent A (respectively B) which contains $A's$ (respectively $B's$) public key, $u_A = g^{-k_A}(modp)$, and $u_B = g^{-k_B}(modp)$, where k_A (respectively k_B) is a secret value known only by A (respectively by B), p and g are public values. At the end of the protocol, A computes $K_A = u_B^{-k_A} mod\, p$ and B computes $K_B = u_A^{-k_B} mod\, p$. K_A and K_B verify the property $K_A = K_B$. Actually, $K_A = u_B^{-k_A}(mod\ p) = g^{-k_B{}^{-k_A}}(mod\ p) = g^{(-k_B)*(-k_A)}(mod\ p) = g^{-k_A{}^{-k_B}}(mod\ p) = u_A^{-k_B}(mod\ p) = K_B$. And hence, K_A becomes the session key between A and B.

Description of the UKS attack on the KAP-HY protocol. In [27], *J. Baek et al.* showed that the redundant signature scheme employed in the KAP-HY protocol possesses the duplicate-signature key selection property, and elaborate on this to show that the KAP-HY is vulnerable to a UKS attack. In this attack, we have two honest agents: Alice playing the role A, Bob playing the role B, and the intruder $\mathcal{I}$. In this attack, the intruder $\mathcal{I}$ waits that $Alice$ initiates a session with him: The attack is described as follows:

$$\begin{cases} 1 \;\; Alice \Rightarrow \mathcal{I}: \;\; u_a, id_a \\ 2 \;\; \mathcal{I} \Rightarrow Bob: \;\; u_a, id_a \\ 3 \;\; Bob \Rightarrow \mathcal{I}(Alice): \;\; u_b, sig_b(u_a), id_b \\ 4 \;\; \mathcal{I} \Rightarrow Alice: \;\; u_b, sig_b(u_a), id_{\mathcal{I}} \\ 5 \;\; Alice \Rightarrow \mathcal{I}: \;\; sig_a(u_b) \\ 6 \;\; \mathcal{I}(Alice) \Rightarrow Bob: \;\; sig_a(u_b) \end{cases}$$

In this attack, the intruder $\mathcal{I}$ records, but passes unchanged, the first message, and initiates a session as $Alice$ with Bob. It then intercepts the message sent by Bob to $Alice$, and, exploiting the duplicate-signature key selection property of the redundant signature scheme, builds from the public key of Bob and from the message $sig_b(u_a)$ a pair of signing and verifying keys, and registers this key pair. $\mathcal{I}$ then sends the signature to $Alice$, but this time accompanied by his identity (4). The main point is that when $Alice$ checks the signature of the incoming message, she accepts it on the ground that it seems to originate from $\mathcal{I}$. At the end of this execution, $Alice$ believes that the key is shared with $\mathcal{I}$ whereas it is actually shared with Bob.

4.3 Destructive exclusive ownership property

4.3.1 Description of DEO property

In [170], *T. Pornin and J. P. Stern* studied the duplicate-signature key selection property. They introduced the notion of *conservative exclusive ownership* defined as follows: a signature scheme with appendix is said to provide conservative exclusive ownership if it does not have duplicate-signature key selection property.

T. Pornin and J.P. Stern [170] also introduced the notion of *destructive exclusive ownership* defined as follows: a signature scheme with appendix is said to provide destructive exclusive ownership if it is computationally infeasible for the intruder, given an agent public key Pk_A, a message m and $A's$ signature s of m, to produce a new pair of secret and public keys $(Sk_{\mathcal{I}}, Pk_{\mathcal{I}})$, a new message $m' \neq m$, such that $Sk_{\mathcal{I}}$ matches $Pk_{\mathcal{I}}$, and $ver(m', s, Pk_{\mathcal{I}}) = 1$.

Example 21 *In [170], the authors showed that, in certain circumstances, RSA [174] and DSS [6] do not provide destructive exclusive ownership.*

In what follows, we make use of "signature schemes vulnerable to constructive exclusive ownership property", and "signature schemes vulnerable to destructive exclusive ownership property". We say that *a signature scheme is vulnerable to conservative exclusive ownership property* if it does not provide that property, that is if it has duplicate-signature key selection property, and similarly, we say that *a signature scheme is vulnerable to destructive exclusive ownership property* if it does not provide that property.

4.4 Decidability results

In this section, we show that the insecurity problem for the class of cryptographic protocols using signature schemes vulnerable to conservative exclusive ownership property (respectively for the class of cryptographic protocols using signature schemes vulnerable to destructive exclusive ownership property) is decidable. To get this decidability result, we follow the symbolic approach described in Chapter 2, that is, we reduce the insecurity problem for our two classes of cryptographic protocols to the satisfiability problem of respectively two classes of constraint systems, and we give a procedure to decide the satisfiability problem for these two classes of constraint systems.

4.4.1 Symbolic model for constructive exclusive ownership vulnerability property

In order to symbolically analyse the class of cryptographic protocols using digital signature schemes vulnerable to the constructive exclusive ownership property (also called digital signature schemes having duplicate signature key selection property), we consider the following signature $\mathcal{F}_{DSKS} = \{Sk, Pk, sig, ver, Sk', Pk', 1\}$ where

- Sk, denoting the secret key generation function which is a part of the key generation algorithm G, is a function with arity 1,
- Pk, denoting the public key generation function which is a part of the key generation algorithm G, is a function with arity 1,
- sig, denoting the signature generation algorithm, is a function with arity 2,
- ver, denoting the verification algorithm, is a function with arity 3,
- Sk', denoting the "special" intruder secret key generation function, is a function with arity 2,
- Pk', denoting the "special" intruder public key generation function, is a function with arity 2,
- 1, denoting a possible output of ver, is a function with arity 0,

The functions Sk, Pk (respectively sig and ver) given above abstract the two parts of the key generation algorithm G (respectively the signature generation algorithm, and the verification algorithm) in a signature scheme. The key generation algorithm employs randomly generated number to perform its computation. We assume that this number is kept secret and that it is destroyed at the end of the computation. We abstract this situation by assuming the functions modelling this algorithm are private. The special functions Sk', Pk' abstract the ability of the intruder, knowing an agent's public key (pk), and the agent's signature s on a message m, to construct a new pair of secret and public keys $(Pk'(pk, s), Sk'(pk, s))$ such that the verification of s with respect to m and the new public key succeeds. We assume that $\mathcal{F}_{DSKS} = \mathcal{F}_{DSKS_{pub}} \cup \mathcal{F}_{DSKS_{pri}}$ where $\mathcal{F}_{DSKS_{pub}} = \{sig, ver, Sk', Pk', 1\}$ and $\mathcal{F}_{DSKS_{pri}} = \{Sk, Pk\}$.

The constructive exclusive ownership vulnerability property is represented by the following equational theory, denoted by $\mathcal{H}_{\mathcal{DSKS}}$:

$$\mathcal{H}_{\mathcal{DSKS}} = \begin{cases} ver(x, sig(x, Sk(y)), Pk(y)) = 1 \\ ver(x, sig(x, Sk'(y_1, y_2)), Pk'(y_1, y_2)) = 1 \\ sig(x, Sk'(Pk(y), sig(x, Sk(y)))) = sig(x, Sk(y)) \end{cases}$$

The intruder system we consider to analyse our class of cryptographic protocols is given as follows:

$$\mathcal{I}_{DSKS} = \langle \mathcal{F}_{\mathcal{DSKS}}, \mathcal{T}_{\mathcal{DSKS}}, \mathcal{H}_{\mathcal{DSKS}} \rangle$$

with:

$$\begin{cases} \mathcal{F}_{\mathcal{DSKS}} &= \mathcal{F}_{DSKS_{pub}} \cup \mathcal{F}_{DSKS_{pri}} \\ \mathcal{F}_{DSKS_{pub}} &= \{sig, ver, Sk', Pk', 1\} \\ \mathcal{F}_{DSKS_{pri}} &= \{Sk, Pk\} \\ \mathcal{T}_{\mathcal{DSKS}} &= \{sig(x,y), ver(x,y,z), Sk'(x,y), Pk'(x,y), 1\} \end{cases}$$

The associated set of intruder deduction rules, denoted by $\mathcal{L}_{DSKS}$ is given as follows:

$$\mathcal{L}_{DSKS} = \begin{cases} x, y \rightarrow sig(x,y) \\ x, y, z \rightarrow ver(x,y,z) \\ x, y \rightarrow Sk'(x,y) \\ x, y \rightarrow Pk'(x,y) \\ \emptyset \rightarrow 1 \end{cases}$$

In what follows, we introduce the rewrite system, $\mathcal{R}_{DSKS}$, generating the equational theory $\mathcal{H}_{DSKS}$, and we prove that $\mathcal{R}_{DSKS}$ is convergent. The rewrite system $\mathcal{R}_{DSKS}$ is obtained by applying Knuth-Bendix completion procedure [131] on $\mathcal{H}_{\mathcal{DSKS}}$. This completion procedure is described in Chapter 2, at Section 2.1.6.

Lemma 25 *$\mathcal{H}_{\mathcal{DSKS}}$ is generated by the convergent rewriting system:*

$$\mathcal{R}_{\mathcal{DSKS}} = \begin{cases} ver(x, sig(x, Sk(y)), Pk(y)) \rightarrow 1 \\ ver(x, sig(x, Sk'(y_1, y_2)), Pk'(y_1, y_2)) \rightarrow 1 \\ ver(x, sig(x, Sk(y)), Pk'(Pk(y), sig(x, Sk(y)))) \rightarrow 1 \\ sig(x, Sk'(Pk(y), sig(x, Sk(y)))) \rightarrow sig(x, Sk(y)) \end{cases}$$

PROOF.

The application of the *Knuth-Bendix* completion procedure [131] on $\mathcal{H}_{\mathcal{DSKS}}$ terminates successfully and outputs the rewrite system $\mathcal{R}_{\mathcal{DSKS}}$, which is a convergent rewrite system generating $\mathcal{H}_{DSKS}$ (Chapter 2, Section 2.1.6). ■

Lemma 26 *Any $\mathcal{R}_{DSKS}$-narrowing derivation starting from any term t terminates.*

PROOF.

In Lemma 25, we proved that $\mathcal{R}_{DSKS}$ is a convergent rewrite system. Following the definition of basic narrowing (Definition 13, Chapter 2), it is easy to see that any right member of any $\mathcal{R}_{DSKS}$ rule is not $\mathcal{R}_{DSKS}$-basic narrowable, and hence, by Theorem 1 (Chapter 2) , we conclude that any $\mathcal{R}_{DSKS}$-narrowing derivation starting from any term terminates.■

Lemma 27 *$\mathcal{H}_{\mathcal{DSKS}}$ has the finite variant property.*

PROOF.

In Lemma 25, we proved that $\mathcal{H}_{DSKS}$ is generated by a convergent rewrite system $\mathcal{R}_{DSKS}$, and in lemma 26, we proved that any $\mathcal{R}_{DSKS}$-narrowing derivation starting from any term terminates. In [86], the authors showed that each convergent rewrite system $\mathcal{R}$ such that any $\mathcal{R}$-basic narrowing derivation starting from any term terminates has the finite variant property. This allows us to conclude that $\mathcal{R}_{DSKS}$ and hence $\mathcal{H}_{DSKS}$ has the finite variant property. ■

4.4.2 Symbolic model for destructive exclusive ownership vulnerability property

In order to symbolically analyse the class of cryptographic protocols using digital signature schemes vulnerable to the destructive exclusive ownership property, we consider the following signature $\mathcal{F}_{DEO} = \{Sk, Pk, sig, ver, Sk", Pk", f, 1\}$ where

- Sk, denoting the secret key generation function which is a part of the key generation algorithm G, is a function with arity 1,
- Pk, denoting the public key generation function which is a part of the key generation algorithm G, is a function with arity 1,
- sig, denoting the signature generation algorithm, is a function with arity 2,
- ver, denoting the verification algorithm, is a function with arity 3,
- $Sk"$, denoting the "special" intruder secret key generation function, is a function with arity 2,
- $Pk"$, denoting the "special" intruder public key generation function, is a function with arity 2,
- f, denoting the "special" algorithm applied by the intruder to generate a "special" message, is a function with arity 2,
- 1, denoting a possible output of ver, is a function with arity 0.

The functions Sk, Pk (respectively sig and ver) given above abstract the two parts of the key generation algorithm G(respectively the signature generation algorithm, and the verification algorithm) in a signature scheme. Following the same reasons as in Section 4.4.1, Sk and Pk are private function symbols. The special functions $Sk", Pk", f$ abstract the ability of the intruder, knowing an agent's public key (pk), and the agent's signature s on a message m, to construct

a new pair of secret and public keys ($Sk"(pk, s)$, $Pk"(pk, s)$), a new message m' ($m' = f(pk, s)$) such that the verification of s with respect to the new message m' and the new public key succeeds. We assume that $\mathcal{F}_{DEO} = \mathcal{F}_{DEO_{pub}} \cup \mathcal{F}_{DEO_{pri}}$ where $\mathcal{F}_{DEO_{pub}} = \{sig, ver, Sk", Pk", f, 1\}$ and $\mathcal{F}_{DEO_{pri}} = \{Sk, Pk\}$.

In addition to the signature $\mathcal{F}_{DEO}$, and as described in Chapter 2, we make use of an infinite set of variables $\mathcal{X}$ and an infinite set of constants $\mathcal{C}$.

The destructive exclusive ownership vulnerability property is represented by the following equational theory, denoted by $\mathcal{H}_{\mathcal{DEO}}$:

$$\mathcal{H}_{\mathcal{DEO}} = \begin{cases} ver(x, sig(x, Sk(y)), Pk(y)) = 1 \\ ver(x, sig(x, Sk"(y_1, y_2)), Pk"(y_1, y_2)) = 1 \\ sig(f(Pk(y), sig(x, Sk(y))), Sk"(Pk(y), sig(x, Sk(y))) = sig(x, Sk(y)) \end{cases}$$

The intruder system we consider to analyse our class of cryptographic protocols is given as follows:

$$\mathcal{I}_{DEO} = \langle \mathcal{F}_{\mathcal{DEO}}, \mathcal{T}_{\mathcal{DEO}}, \mathcal{H}_{\mathcal{DEO}} \rangle$$

with:

$$\begin{cases} \mathcal{F}_{\mathcal{DEO}} &= \mathcal{F}_{DEO_{pub}} \cup \mathcal{F}_{DEO_{pri}} \\ \mathcal{F}_{DEO_{pub}} &= \{sig, ver, Sk", Pk", f, 1\} \\ \mathcal{F}_{DEO_{pri}} &= \{Sk, Pk\} \\ \mathcal{T}_{\mathcal{DEO}} &= \{sig(x, y), ver(x, y, z), Sk"(x, y), Pk"(x, y), f(x, y), 1\} \end{cases}$$

The associated set of intruder deduction rules, denoted by $\mathcal{L}_{DEO}$ is given as follows:

$$\mathcal{L}_{DEO} = \begin{cases} x, y \rightarrow sig(x, y) \\ x, y, z \rightarrow ver(x, y, z) \\ x, y \rightarrow Sk"(x, y) \\ x, y \rightarrow Pk"(x, y) \\ x, y \rightarrow f(x, y) \\ \emptyset \rightarrow 1 \end{cases}$$

In what follows, we introduce the rewrite system, $\mathcal{R}_{DEO}$, generating the equational theory $\mathcal{H}_{DEO}$, and we prove that $\mathcal{R}_{DEO}$ is convergent. The rewrite system $\mathcal{R}_{DEO}$ is obtained by applying Knuth-Bendix completion procedure [131] on $\mathcal{H}_{\mathcal{DEO}}$. This completion procedure is described in Chapter 2, at Section 2.1.6.

Lemma 28 *$\mathcal{H}_{\mathcal{DEO}}$ is generated by the convergent rewriting system:*

$$\mathcal{R}_{DEO} = \begin{cases} ver(x, sig(x, Sk(y)), Pk(y)) \rightarrow 1 \\ ver(x, sig(x, Sk"(y_1, y_2)), Pk"(y_1, y_2)) \rightarrow 1 \\ ver(f(Pk(y), sig(x, Sk(y))), sig(x, Sk(y)), Pk"(Pk(y), sig(x, Sk(y)))) \rightarrow 1 \\ sig(f(Pk(y), sig(x, Sk(y))), Sk"(Pk(y), sig(x, Sk(y)))) \rightarrow sig(x, Sk(y)) \end{cases}$$

PROOF.
The application of the *Knuth-Bendix* completion procedure [131] on $\mathcal{H}_{\mathcal{DEO}}$ terminates successfully and outputs the rewrite system $\mathcal{R}_{\mathcal{DEO}}$, which is a convergent rewrite system generating $\mathcal{H}_{DEO}$ (Chapter 2, Section 2.1.6). ■

Lemma 29 *Any $\mathcal{R}_{DEO}$-narrowing derivation starting from any term t terminates.*

PROOF.
In Lemma 28, we proved that $\mathcal{R}_{DEO}$ is a convergent rewrite system. Following the definition of basic narrowing (Definition 13, Chapter 2), it is easy to see that any right member of any $\mathcal{R}_{DEO}$ rule is not $\mathcal{R}_{DEO}$-basic narrowable, and hence, by Theorem 1 (Chapter 2) , we conclude that any $\mathcal{R}_{DSKS}$-narrowing derivation starting from any term terminates.■

Lemma 30 *$\mathcal{H}_{\mathcal{DEO}}$ has the finite variant property.*

PROOF.
In Lemma 28, we proved that $\mathcal{H}_{DEO}$ is generated by a convergent rewrite system $\mathcal{R}_{DEO}$, and in lemma 29, we proved that any $\mathcal{R}_{DEO}$-narrowing derivation starting from any term terminates. In [86], the authors showed that each convergent rewrite system $\mathcal{R}$ such that any $\mathcal{R}$-basic narrowing derivation starting from any term terminates has the finite variant property. This allows us to conclude that $\mathcal{R}_{DEO}$ and hence $\mathcal{H}_{DEO}$ has the finite variant property. ■

4.4.3 Decidability of $\mathcal{H}_{DSKS}$- and $\mathcal{H}_{DEO}$-unifiability problems

In this section, we prove the decidability of $\mathcal{H}_{DSKS}$- and $\mathcal{H}_{DEO}$-unifiability problems. Given an arbitrary equational theory $\mathcal{H}$, we recall the definition of $\mathcal{H}$-unifiability problem (Chapter 2, Definition 3).
The $\mathcal{H}$-unifiability problem is defined as follows:

Input: A $\mathcal{H}$-unification system $\mathcal{U}$.
Output: SAT if and only if there exists a substitution σ such that $\sigma \models_{\mathcal{H}} \mathcal{U}$.

In Chapter 2 (Section 2.1.8), we present a complete, sound and finite $\mathcal{H}$-unification algorithm where $\mathcal{H}$ is an equational theory having the finite variant property.

Since $\mathcal{R}_{DSKS}$ and $\mathcal{R}_{DEO}$ have the finite variant property (Lemma 27 and Lemma 30 respectively), we conclude the following theorem.

Theorem 6 *The $\mathcal{H}_{DSKS}$- and $\mathcal{H}_{DEO}$-unifiability problems are decidable.*

We prove below that $\mathcal{H}_{DSKS}$- (respectively $\mathcal{H}_{DEO}$-) unifiability problem is in fact in $NPtime$.

Complexity of $\mathcal{H}_{DSKS}$- and $\mathcal{H}_{DEO}$-unification

Theorem 7 *The $\mathcal{H}_{\mathcal{DSKS}}$-unifiability problem (respectively. $\mathcal{H}_{\mathcal{DEO}}$-unifiability problem) can be decided in $NPtime$.*

PROOF.

Let us prove the theorem for $\mathcal{H}_{\mathcal{DSKS}}$-unifiability problem. We have $\mathcal{H}_{DSKS}$ is an equational theory generated by a convergent rewrite system $\mathcal{R}_{DSKS}$ such that the right hand side of every rule in $\mathcal{R}_{DSKS}$ is not $\mathcal{R}_{DSKS}$-narrowable.

We proved in Theorem 6 the decidability of $\mathcal{H}_{DSKS}$-unifiability problem, and the unification algorithm that solves that problem is the algorithm proposed in Chapter 2, at Section 2.1.8. Let us prove that this algorithm runs in $NPtime$. Let s and t be two terms, and let $M = g(s,t)$, (g is a new function symbol representing the cartesian product), and $m = \|M\|_{dag} = \|s\|_{dag} + \|t\|_{dag} + 1$ be the length of M. Following the unification algorithm we use to solve this problem, if there is a $\mathcal{H}_{DSKS}$-unifier of s and t, then there is a variant substitution θ of M such that $(s\theta)\downarrow$ and $(t\theta)\downarrow$ are syntactically unifiable. Given any variant θ' of M, lemma 5 (Chapter 2) proved that we can guess in $NPtime$ another variant substitution θ of M such that θ is more general modulo $\mathcal{H}$ than θ'. Assuming that we always guess the right variant substitution θ' for which there is another variant substitution θ, computable in $NPtime$, that makes $(s\theta)\downarrow$ and $(t\theta)\downarrow$ syntactically unifiable, and since the unification algorithm modulo $\emptyset$-theory runs in $Ptime$, we conclude that $\mathcal{H}_{\mathcal{DSKS}}$-unifiability can be decided in $NPtime$.

By the same reasoning, we prove that $\mathcal{H}_{\mathcal{DEO}}$-unifiability can be decided in $NPtime$. ■

Remark. Given a H_{DSKS} (respectively H_{DEO})-unification system $\mathcal{U}$ such that $\mathcal{U}$ is H_{DSKS} (respectively H_{DEO})-unifiable. There is a non deterministic algorithm that compute a H_{DSKS} (respectively H_{DEO})-unifier of $\mathcal{U}$.

4.4.4 Saturation of $\mathcal{I}_{DSKS}$ and $\mathcal{I}_{DEO}$ deduction rules

Construction of the saturation Let $\mathcal{I} = \langle \mathcal{F}, \mathcal{T}_{\mathcal{I}}, \mathcal{H} \rangle$ be an intruder deduction system (Definition 16) and $\mathcal{H}$ be the considered equational theory. Assume that $\mathcal{H}$ is generated by a convergent rewrite system $\mathcal{R}$ and has the finite variant property. Let $\mathcal{L}_{\mathcal{I}}$ be the intruder deduction rules associated with $\mathcal{I}$. We recall that rules in $\mathcal{L}_{\mathcal{I}}$ are of the form $x_1, \ldots, x_n \rightarrow f(x_1, \ldots, x_n)$ and f is a public function symbol in $\mathcal{F}$.

In what follows, we make use of the notation $\widetilde{l}$ to mean a set of terms $l_1, \ldots, l_n$ for $n \geq 1$.

The saturation of the set of deduction rules $\mathcal{L}_{\mathcal{I}}$ modulo the equational theory $\mathcal{H}$ is given in Figure 4.1.

Figure 4.1 System of saturation rules

Input:

The intruder deduction rules $\mathcal{L}_{\mathcal{I}}$.

Initialisation:

Let $\mathcal{I}' = \langle \mathcal{F}, \mathcal{L}_{\mathcal{I}'}, \emptyset \rangle$ be the variant of the intruder deduction system $\mathcal{I}$ (Definition 19, Chapter 2). We recall that

$$\mathcal{L}_{\mathcal{I}'} = \bigcup_{\substack{x_1,\ldots,x_n \rightarrow f(x_1,\ldots,x_n) \in \mathcal{L}_{\mathcal{I}} \\ \theta \text{ variant substitution of } f(x_1,\ldots,x_n)}} x_1\theta,\ldots,x_n\theta \rightarrow (f(x_1,\ldots,x_n)\theta)\downarrow$$

Step 1. Start with $\mathcal{L}_{\mathcal{I}}" = \mathcal{L}_{\mathcal{I}'}$. Apply on $\mathcal{L}_{\mathcal{I}}"$ the rules below until any added rule is subsumed by a rule already present in $\mathcal{L}_{\mathcal{I}}"$.

$$\text{Subsumption :} \quad \frac{\widetilde{l_1} \rightarrow r \in \mathcal{L}_{\mathcal{I}}" \quad \widetilde{l_2} \rightarrow r \in \mathcal{L}_{\mathcal{I}}"}{\mathcal{L}_{\mathcal{I}}" \leftarrow \mathcal{L}_{\mathcal{I}}" \setminus \left\{\widetilde{l_2} \rightarrow r\right\}} \; \widetilde{l_1} \subseteq \widetilde{l_2}$$

$$\text{Closure :} \quad \frac{\widetilde{l_1} \rightarrow r_1 \in \mathcal{L}_{\mathcal{I}}", \quad t, \widetilde{l_2} \rightarrow r_2 \in \mathcal{L}_{\mathcal{I}}"}{\mathcal{L}_{\mathcal{I}}" \leftarrow \mathcal{L}_{\mathcal{I}}" \cup \left\{(\widetilde{l_1}, \widetilde{l_2} \rightarrow r_2)\sigma\right\}} \quad \begin{array}{c} t \notin \mathcal{X} \\ \sigma = mgu_{\emptyset}(r_1, t) \end{array}$$

Output:

Output $\mathcal{L}_{\mathcal{I}"}$.

We define two new deduction systems $\mathcal{I}' = \langle \mathcal{F}, \mathcal{L}_{\mathcal{I}'}, \emptyset \rangle$ and $\mathcal{I}" = \langle \mathcal{F}, \mathcal{L}_{\mathcal{I}"}, \emptyset \rangle$. We remark that $\mathcal{I}$ satisfies the definition of intruder deduction system as given in Definition 16 (Chapter 2) and in [73], and the intruder deduction systems $\mathcal{I}', \mathcal{I}"$ satisfy the definition of intruder deduction system as given in [72].

Saturation of $\mathcal{I}_{DSKS}$ and $\mathcal{I}_{DEO}$ The application of the saturation given in Figure 4.1 on $\mathcal{L}_{\mathcal{DSKS}}$ terminates, and yields the following two sets of rules, each corresponding to a step of the saturation algorithm (respectively the *Initialisation* and the *first* step):

$$\mathcal{L}'_{\mathcal{DSKS}} = \mathcal{L}_{\mathcal{DSKS}} \cup \begin{cases} x, Sk'(Pk(y), sig(x, Sk(y))) \rightarrow sig(x, Sk(y)) \\ x, sig(x, Sk(y)), Pk(y) \rightarrow 1 \\ x, sig(x, Sk'(y_1, y_2)), Pk'(y_1, y_2) \rightarrow 1 \\ x, sig(x, Sk(y)), Pk'(Pk(y), sig(x, Sk(y))) \rightarrow 1 \end{cases}$$

$$\mathcal{L}"_{\mathcal{DSKS}} = \mathcal{L}_{\mathcal{DSKS}} \cup \begin{cases} x, Sk(y) \rightarrow sig(x, Sk(y)) \\ x, Sk'(Pk(y), sig(x, Sk(y))) \rightarrow sig(x, Sk(y)) \end{cases}$$

and, the application of the saturation given in Figure 4.1 on $\mathcal{L}_{\mathcal{DEO}}$ terminates, and yields the following two sets of rules, each corresponding to a step of the saturation algorithm (respectively the *Initialisation* and the *first* step):

$$\mathcal{L}'_{\mathcal{DEO}} = \mathcal{L}_{\mathcal{DEO}} \cup \begin{cases} x, sig(x, Sk(y)), Pk(y) \rightarrow 1 \\ x, sig(x, Sk"(y_1, y_2)), Pk"(y_1, y_2) \rightarrow 1 \\ f(Pk(y), sig(x, Sk(y))), sig(x, Sk(y)), Pk"(Pk(y), sig(x, Sk(y))) \rightarrow 1 \\ f(Pk(y), sig(x, Sk(y))), Sk"(Pk(y), sig(x, Sk(y))) \rightarrow sig(x, Sk(y)) \end{cases}$$

$$\mathcal{L}"_{\mathcal{DEO}} = \mathcal{L}_{\mathcal{DEO}} \cup \{ f(Pk(y), sig(x, Sk(y))), Sk"(Pk(y), sig(x, Sk(y))) \rightarrow sig(x, Sk(y)) \}$$

From the intruder systems $\mathcal{I}_{DSKS}$ and $\mathcal{I}_{DEO}$, we define four new intruder systems:
$\mathcal{I}'_{DSKS} = \langle \mathcal{F}_{\mathcal{DSKS}}, \mathcal{L}'_{\mathcal{DSKS}}, \emptyset \rangle$,
$\mathcal{I}"_{DSKS} = \langle \mathcal{F}_{\mathcal{DSKS}}, \mathcal{L}"_{\mathcal{DSKS}}, \emptyset \rangle$,
$\mathcal{I}'_{DEO} = \langle \mathcal{F}_{\mathcal{DEO}}, \mathcal{L}'_{\mathcal{DEO}}, \emptyset \rangle$ and,
$\mathcal{I}"_{DEO} = \langle \mathcal{F}_{\mathcal{DEO}}, \mathcal{L}"_{\mathcal{DEO}}, \emptyset \rangle$.

We remark that $\mathcal{I}_{DSKS}$ and $\mathcal{I}_{DEO}$ satisfy the definition of intruder deduction system as given in Definition 16 (Chapter 2) and in [73], and the intruder deduction systems $\mathcal{I}'_{DSKS}, \mathcal{I}'_{DEO}, \mathcal{I}"_{DSKS}, \mathcal{I}"_{DEO}$ satisfy the definition of intruder deduction system as given in [72].

In the rest of this chapter, we assume that $\mathcal{H}$, $\mathcal{R}$, $\mathcal{L}$, $\mathcal{L}'$, $\mathcal{L}"$, $\mathcal{I}$, $\mathcal{I}'$, and $\mathcal{I}"$ to be either respectively $\mathcal{H}_{\mathcal{DSKS}}$, $\mathcal{R}_{\mathcal{DSKS}}$, $\mathcal{L}_{\mathcal{DSKS}}$, $\mathcal{L}'_{\mathcal{DSKS}}$, $\mathcal{L}"_{DSKS}$, $\mathcal{I}_{DSKS}$, $\mathcal{I}'_{DSKS}$, $\mathcal{I}"_{DSKS}$ or respectively $\mathcal{H}_{\mathcal{DEO}}$, $\mathcal{R}_{\mathcal{DEO}}$, $\mathcal{L}_{\mathcal{DEO}}$, $\mathcal{L}'_{\mathcal{DEO}}$, $\mathcal{L}"_{DEO}$, $\mathcal{I}_{DEO}$, $\mathcal{I}'_{DEO}$, $\mathcal{I}"_{DEO}$.

Properties of the saturation Let us first prove that the intruder deduction system obtained after the saturation gives exactly the same deductive power to the initial intruder deduction system.

Given a set of terms E and a term t, from now on, we make use of the notation $E \rightarrow_{\mathcal{I}}^{*} t$ to means that there exists a $\mathcal{I}$-derivation starting from E of goal t, that is $t \in \overline{E}^{\mathcal{I}}$.

Lemma 31 *For any set of normal ground terms E and any normal ground term t we have: $t \in \overline{E}^{\mathcal{I}} t$ if and only if $t \in \overline{E}^{\mathcal{I}"}$.*

PROOF.

Lemma 8 (Chapter 2) showed that for any two sets of ground terms in normal form E and F, we have $E \rightarrow_{\mathcal{I}} F$ if and only if $E \rightarrow_{\mathcal{I}'} F$. This implies that $t \in \overline{E}^{\mathcal{I}}$ if and only if $t \in \overline{E}^{\mathcal{I}'}$ for any set of ground terms E in normal form and any ground term t in normal form. Now, we prove that $t \in \overline{E}^{\mathcal{I}'}$ if and only if $t \in \overline{E}^{\mathcal{I}"}$. First, let us assume that $t \in \overline{E}^{\mathcal{I}'}$, that is, there exists a $\mathcal{I}'$-derivation D starting from E of goal t, and let us prove that there exists a $\mathcal{I}"$-derivation starting from E of goal t. If there exists a step in the derivation D which uses a rule $\widetilde{l} \rightarrow r \in \mathcal{L}_{\mathcal{I}'}$ but not in $\mathcal{L}_{\mathcal{I}}"$, then, by construction of $\mathcal{L}_{\mathcal{I}}"$, there exists another rule $\widetilde{l_1} \rightarrow r$ in $\mathcal{L}_{\mathcal{I}}"$ such that $\widetilde{l_1} \subseteq \widetilde{l}$ and thus that can be applied instead of $\widetilde{l} \rightarrow r$. We conclude that $t \in \overline{E}^{\mathcal{I}"}$.

For the reciprocal, let us assume that there exists a $\mathcal{I}"$-derivation starting from E of goal t, and let us prove that there exists a $\mathcal{I}'$-derivation starting from E of goal t. We begin by defining an arbitrary order on the rules of $\mathcal{L}_{\mathcal{I}'}$, and we extend this order to the rules of $\mathcal{L}_{\mathcal{I}}" \setminus \mathcal{L}_{\mathcal{I}'}$ as follows: the rules of $\mathcal{L}_{\mathcal{I}'}$ are smaller than the rules of $\mathcal{L}_{\mathcal{I}}" \setminus \mathcal{L}_{\mathcal{I}'}$ and the rules of $\mathcal{L}_{\mathcal{I}}" \setminus \mathcal{L}_{\mathcal{I}'}$ are ordered according to the order of their construction during the saturation. Let $\mathrm{M}(D)$ be the multiset of deduction rules applied in D. Let $\Omega(E,t) = \{D \mid D : E(\rightarrow_{\mathcal{I}'}^{*} \cup \rightarrow_{\mathcal{I}"})^{*} t\}$. Since $t \in \overline{E}^{\mathcal{I}"}$, $\Omega(E,t) \neq \emptyset$. Let D be a derivation in $\Omega(E,t)$ having the minimal $\mathrm{M}(D)$, and let us prove that D does not use rules in $\mathcal{L}_{\mathcal{I}}" \setminus \mathcal{L}_{\mathcal{I}'}$. By contradiction, suppose that D uses a rule $\widetilde{l} \rightarrow r \in \mathcal{L}_{\mathcal{I}}" \setminus \mathcal{L}_{\mathcal{I}'}$. Since $\widetilde{l} \rightarrow r \notin \mathcal{L}_{\mathcal{I}'}$, it has been constructed according to the closure rule given in Figure **??**. This implies that there exists two rules $\widetilde{l_1} \rightarrow r_1$ and $s, \widetilde{l_2} \rightarrow r_2$ in $\mathcal{L}_{\mathcal{I}"}$ such that $\mu = mgu(r_1, s)$, $s \notin \mathcal{X}$, $\widetilde{l} = (\widetilde{l_1}, \widetilde{l_2})\mu$ and $r = r_2\mu$. suppose that $\widetilde{l} \rightarrow r$ is applied on the set of terms F, $F \rightarrow_{\widetilde{l} \rightarrow r} F, g$. Since $\widetilde{l}\sigma \subseteq F$ and $r\sigma = g$ for a substitution σ, we have $\widetilde{l_1}\mu\sigma \subseteq F$ and $(s, \widetilde{l_2})\mu\sigma \subseteq F \cup r_1\mu\sigma$, this implies that $F \rightarrow_{\widetilde{l_1} \rightarrow r_1} F, r_1\mu\sigma \rightarrow_{\widetilde{l_2}, s \rightarrow r_2} F, r_1\mu\sigma, g$. Let D' be the derivation where $\widetilde{l_1} \rightarrow r_1$ and $s, \widetilde{l_2} \rightarrow r_2$ replace $\widetilde{l} \rightarrow r$. D' is in $\Omega(E,t)$. Since $\widetilde{l_1} \rightarrow r_1$ and $s, \widetilde{l_2} \rightarrow r_2$ have an order smaller than the order of $\widetilde{l} \rightarrow r$, we have $\mathrm{M}(D') < \mathrm{M}(D)$ which contradicts the minimality of $\mathrm{M}(D)$.

We conclude that D does not use rules in $\mathcal{L}_{\mathcal{I}"} \setminus \mathcal{L}_{\mathcal{I}'}$. This yields the reciprocal of the lemma. ■

The next lemma proves that, for any set of terms E in normal form and any term t in normal form, if $t \in \overline{E}^{\mathcal{I}"}$ then there is a $\mathcal{I}"$-derivation starting from E of goal t such that for any rule $\widetilde{l} \to r$ applied in the derivation with the substitution σ, the instances, with respect to σ, of the non variable terms in $\widetilde{l}$ belong to E, and hence are not the result of another deduction rules.

We make use of $Cons(D)$ to denote the set of terms constructed during the derivation D, more formally, let E and t be respectively a set of terms and a term in normal form and let D be a derivation starting from E of goal t, $D : E = E_0 \to E_0, t_1 \to \ldots \to E_{n-2}, t_{n-1} \to E_{n-1}, t$, $Cons(D) = E_0 \cup \{t_1, \ldots, t_{n-1}\}$.

Lemma 32 *Let E (respectively t) be a set of terms (respectively a term) in normal form. If $t \in \overline{E}^{\mathcal{I}"}$, then for every $\mathcal{I}"$-derivation D starting from E of goal t, we have either D satisfies the following property*

$$prop : \textit{for all } \mathcal{I}" \textit{ rules } \widetilde{l} \to r \textit{ applied with substitution } \sigma, \textit{for all } s \in \widetilde{l} \setminus \mathcal{X}, \textit{we have } s\sigma \in E$$

or there exists another $\mathcal{I}"$-derivation D' starting from E of goal t such that $Cons(D) = Cons(D')$ and D' satisfies the property $prop$.

PROOF.

We have $t \in \overline{E}^{\mathcal{I}"}$ implies that the set $\Omega(E, t)$ of $\mathcal{I}"$-derivations starting from E of goal t is not empty. Let $D \in \Omega(E, t)$, $D : E = E_0 \to E_1 \to \ldots \to E_{n-1}, t$, and suppose that D does not satisfy the property $prop$. we denote $\widetilde{l_i} \to r_i$ the rule applied at step i with the substitution σ. Let us (pre-)order derivations in $\Omega(E, t)$ with a measure M such that M(D') for a derivation D' is a multiset of integers constructed as follows: starting with $M(D') = \emptyset$, for all steps k, $1 \leq k \leq n$, for every term $u \in l_k\sigma$ obtained by former rule, add k to $M(D')$. Since this pre-order is well-founded, there exists a derivation $d \in \Omega(E, t)$ such that $M(d)$ is minimum, and $Cons(d) = Cons(D)$. Let us prove that d satisfies the property $prop$. By contradiction, assume that d does not satisfy $prop$ and let j be the first step in d such that $\widetilde{l_j} \to r_j$ is the rule applied with substitution σ and there is a term $u \in \widetilde{l_j} \setminus \mathcal{X}$ obtained by a former rule, let $\widetilde{l_h} \to r_h$ be this rule. Since $u \notin \mathcal{X}$, *Closure* can be applied on $\widetilde{l_h} \to r_h$ and $\widetilde{l_j} \to r_j$ and the resulting rule can be applied at step j instead of $\widetilde{l_j} \to r_j$ yielding also E_j. Let d' be the derivation obtained after this replacement, $d' \in \Omega(E, t)$ and $Cons(d') = Cons(d)$. Since $h < j$ and by definition of M, we have $M(d') < M(d)$ which contradicts the minimality of $M(d)$. We deduce that d satisfies $prop$ and then we have the lemma. ■

4.4.5 Decidability of $\mathcal{I}_{DSKS}$ and $\mathcal{I}_{DEO}$ reachability problems

In what follows, we prove the decidability of $\mathcal{I}_{DSKS}$ and $\mathcal{I}_{DEO}$ reachability problems.

Theorem 8 *The $\mathcal{I}_{DSKS}$-reachability (respectively $\mathcal{I}_{DEO}$-reachability) problem is decidable.*

The rest of this section is devoted to the presentation of an algorithm for solving $\mathcal{I}_{DSKS}$-reachability (respectively $\mathcal{I}_{DEO}$-reachability) problems and to a proof scheme of its completeness, correctness and termination. This decision procedure comprises three different steps.

We recall that we make use of $\mathcal{H}$, $\mathcal{R}$, $\mathcal{L}$, $\mathcal{L}'$, $\mathcal{L}$", $\mathcal{I}$, $\mathcal{I}'$, and $\mathcal{I}$" to denote either $\mathcal{H}_{\mathcal{DSKS}}$, $\mathcal{R}_{\mathcal{DSKS}}$, $\mathcal{L}_{\mathcal{DSKS}}$, $\mathcal{L}'_{\mathcal{DSKS}}$, $\mathcal{L}"_{\mathcal{DSKS}}$, $\mathcal{I}_{DSKS}$, $\mathcal{I}'_{DSKS}$, $\mathcal{I}"_{DSKS}$ respectively or $\mathcal{H}_{\mathcal{DEO}}$, $\mathcal{R}_{\mathcal{DEO}}$, $\mathcal{L}_{\mathcal{DEO}}$, $\mathcal{L}'_{\mathcal{DEO}}$, $\mathcal{L}"_{DEO}$, $\mathcal{I}_{DEO}$, $\mathcal{I}'_{DEO}$, $\mathcal{I}"_{DEO}$ respectively.

Let $\mathcal{C}$ be an $\mathcal{I}$-constraint system, $\mathcal{C} = (E_1 \vdash v_1, \ldots, E_n \vdash v_n, \mathcal{U})$.

First step: Guess of a variant constraint system Guess a variant $\mathcal{C}' = (E'_1 \rhd t'_1, \ldots, E'_n \rhd t'_n)$ $\mathcal{I}$-constraint system associated to $\mathcal{C}$. $\mathcal{C}'$ is constructed from $\mathcal{C}$ as given in Definition 28 (Chapter 2).

Lemma 33 *Let $\mathcal{C}$ be a $\mathcal{I}$-constraint system. If σ is a substitution in normal form such that $\sigma \models_{\mathcal{I}} \mathcal{C}$, then there exists a choice of $\mathcal{C}'$ obtained at the end of $Step$ 1 and a substitution σ' in normal form such that $\sigma' \models_{\mathcal{I}"} \mathcal{C}'$.*

PROOF.
Let $\mathcal{C} = (E_1 \vdash v_1, \ldots, E_n \vdash v_n, \mathcal{U})$ be a $\mathcal{I}$-constraint system, and let σ be a substitution in normal form such that $\sigma \models_{\mathcal{I}} \mathcal{C}$. This implies that there exists a substitution θ in normal form and a substitution τ in normal form such that $\tau \models_{\mathcal{I}} ((E_1\theta)\downarrow \rhd (v_1\theta)\downarrow, \ldots, (E_n\theta)\downarrow \rhd (v_n\theta)\downarrow)$, and $((E_1\theta)\downarrow \rhd (v_1\theta)\downarrow, \ldots, (E_n\theta)\downarrow \rhd (v_n\theta)\downarrow)$ is a variant $\mathcal{I}$-constraint system associated to $\mathcal{C}$ (Lemma 12 at Chapter 2). By Lemma 31, we deduce that $\tau \models_{\mathcal{I}} "((E_1\theta)\downarrow \rhd (v_1\theta)\downarrow, \ldots, (E_n\theta)\downarrow \rhd (v_n\theta)\downarrow)$, which concludes the proof. ■

Lemma 34 *Let $\mathcal{C}$ (respectively $\mathcal{C}'$) be a $\mathcal{I}$- (respectively $\mathcal{I}$"-) constraint system. Assume that $\mathcal{C}'$ is obtained from $\mathcal{C}$ at the end of $Step$ 1. If $\mathcal{C}'$ is $\mathcal{I}$"-satisfiable then $\mathcal{C}$ is $\mathcal{I}$-satisfiable.*

PROOF.
Let $\mathcal{C}$ (respectively $\mathcal{C}'$) be a $\mathcal{I}$- (respectively $\mathcal{I}$"-) constraint system and assume that $\mathcal{C}'$ is obtained from $\mathcal{C}$ at then end of $Step$ 1. This implies that $\mathcal{C} = (E_1 \vdash v_1, \ldots, E_n \vdash v_n, \mathcal{U})$ and $\mathcal{C} = ((E_1\theta)\downarrow \rhd (v_1\theta)\downarrow, \ldots, (E_n\theta)\downarrow \rhd (v_n\theta)\downarrow)$

while θ is a substitution in normal form (the construction of θ is shown in Definition 28). Since $\mathcal{C}'$ is $\mathcal{I}$"-satisfiable there exists a normal substitution σ' such that $(v_i\theta)\downarrow\sigma' \in \overline{(E_i\theta)\downarrow\sigma'}^{\mathcal{I}"}$ and thus $(v_i\theta\sigma')\downarrow \in \overline{(E_i\theta\sigma')\downarrow}^{\mathcal{I}"}$, which implies that $(v_i\theta\sigma')\downarrow \in \overline{(E_i\theta\sigma')\downarrow}^{\mathcal{I}}$ (Lemma 31). Let $\sigma = (\theta\sigma')\downarrow$, we have that $(v_i\sigma)\downarrow \in \overline{(E_i\sigma)\downarrow}^{\mathcal{I}}$, and by construction of θ, we have that $\theta \models_{\mathcal{H}} \mathcal{U}$, and hence $\sigma \models_{\mathcal{H}} \mathcal{U}$ which concludes the proof. ■

Figure 4.2 System of transformation rules.

Apply :

$$\frac{\mathcal{C}_\alpha, E \rhd t, \mathcal{C}_\beta}{(\mathcal{C}_\alpha, (E \rhd y)_{y\in l_x}, \mathcal{C}_\beta)\sigma} \quad \begin{array}{c} l_x, l_1, \ldots, l_n \to r \in \mathcal{L}_{\mathcal{I}"} \text{ and } l_x \subseteq \mathcal{X}, t \notin \mathcal{X} \\ e_1, \ldots, e_n \in E \text{ and } \sigma = mgu(\left\{(e_i \stackrel{?}{=}_\emptyset l_i)_i, r \stackrel{?}{=}_\emptyset t\right\}) \end{array}$$

Unif :

$$\frac{\mathcal{C}_\alpha, E \rhd t, \mathcal{C}_\beta}{(\mathcal{C}_\alpha, \mathcal{C}_\beta)\sigma} \quad \begin{array}{c} u, t \notin \mathcal{X} \\ u \in E,\ \sigma = mgu(u \stackrel{?}{=}_\emptyset t) \end{array}$$

Second step: Transformation in solved form. We give now the rules that simplify a modified constraint system. These rules are given in Figure 4.2. Our goal is to transform $\mathcal{C}'$, the modified constraint system obtained from $\mathcal{C}$ at the end of $Step\ 1$, into a modified constraint system in solved form.

The next Lemma shows that the application of a rule from Figure 4.2 on a modified constraint system outputs a modified constraint system.

Lemma 35 *Let $\mathcal{C}'$ be a modified constraint system. The application of $Apply$ and $Unif$ rules on $\mathcal{C}'$ outputs a modified constraint system.*

PROOF.

Let $\mathcal{C}' = (E_1 \rhd t_1, \ldots, E_n \rhd t_n)$ be a modified constraint system. Definition 22 (Chapter 2) implies that $E_i \subseteq E_{i+1}$ and $Var(E_i) \subseteq Var(\{t_1, \ldots, t_{i-1}\})$ for $i \in \{1, \ldots, n\}$. Assume $\mathcal{C}' = (E_1 \rhd x_1, \ldots, E_{i-1} \rhd x_{i-1}, E_i \rhd t_i, E_{i+1} \rhd t_{i+1}, \ldots, E_n \rhd t_n)$ with $t_i \notin \mathcal{X}$, and let us prove that the application of $Apply$ and $Unif$ rules on $\mathcal{C}'$ outputs a modified constraint system.

Unif rule. The application of *Unif* on $\mathcal{C}'$ outputs $\mathcal{C}" = (E_1\sigma \rhd x_1\sigma, \ldots, E_{i-1}\sigma \rhd x_{i-1}\sigma, E_{i+1}\sigma \rhd t_{i+1}\sigma, \ldots, E_n\sigma \rhd t_n\sigma)$ with $\sigma = mgu(u \stackrel{?}{=}_\emptyset t_i)$, $u \in E_i \setminus \mathcal{X}$. $E_j \subseteq E_{j+1}$ implies $E_j\sigma \subseteq E_{j+1}\sigma$. We prove next that $Var(E_j\sigma) \subseteq Var(x_1\sigma, \ldots, x_{i-1}\sigma, t_{i+1}\sigma, \ldots, t_{j-1}\sigma)$. Actually, we have $Var(E_j) \subseteq Var(x_1, \ldots, x_{i-1}, t_i, \ldots, t_{j-1})$. This implies that $Var(E_j\sigma) \subseteq Var(x_1\sigma, \ldots, x_{i-1}\sigma, t_i\sigma, t_{i+1}\sigma, \ldots, t_{j-1}\sigma)$. We have that $t_i\sigma = u\sigma$ and $u \in E_i$, thus $Var(t_i\sigma) = Var(u\sigma) \subseteq Var(E_i\sigma) \subseteq Var(\{x_1\sigma, \ldots, x_{i-1}\sigma\})$. This implies that $Var(E_j\sigma) \subseteq Var(x_1\sigma, \ldots, x_{i-1}\sigma, t_{i+1}\sigma, \ldots, t_{j-1}\sigma)$, and hence, the application of $Unif$ rule on $\mathcal{C}'$ outputs a modified constraint system.

Apply rule. The application of *Apply* on $\mathcal{C}'$ outputs $\mathcal{C}" = (E_1\sigma \rhd x_1\sigma, \ldots, E_{i-1}\sigma \rhd x_{i-1}\sigma, (E_i\sigma \rhd y\sigma)_{y\in l_x}, E_{i+1}\sigma \rhd t_{i+1}\sigma, \ldots, E_n\sigma \rhd t_n\sigma)$ with $\sigma = mgu(r \stackrel{?}{=}_\emptyset t_i, (u_k \stackrel{?}{=}_\emptyset l_k)_{1\leq k\leq m})$, $u_k \in E_i \setminus \mathcal{X}$, and $l_x, l_1, \ldots, l_m \rightarrow r \in \mathcal{L}_{\mathcal{I}"}$. $E_j \subseteq E_{j+1}$ implies $E_j\sigma \subseteq E_{j+1}\sigma$. Let $j \leq i$, we have that $Var(E_j) \subseteq Var(\{x_1, \ldots, x_{j-1}\})$ and hence, $Var(E_j\sigma) \subseteq Var(\{x_1\sigma, \ldots, x_{j-1}\sigma\})$. Let $j > i$, and let us prove that $Var(E_j\sigma) \subseteq Var(\{x_1\sigma, \ldots, x_{i-1}\sigma\} \cup \{y\sigma\}_{y\in l_x} \cup \{t_{i+1}\sigma, \ldots, t_{j-1}\sigma\})$. We have that $Var(E_j\sigma) \subseteq Var(\{x_1\sigma, \ldots, x_{i-1}\sigma, t_i\sigma, t_{i+1}\sigma, \ldots, t_{j-1}\sigma\})$, and $Var(t_i\sigma) = Var(r\sigma) \subseteq Var(l_x\sigma) \cup Var(\{l_1\sigma, \ldots, l_m\sigma\}) \subseteq Var(l_x\sigma) \cup Var(E_i\sigma) \subseteq Var(l_x\sigma) \cup Var(\{x_1\sigma, \ldots, x_{i-1}\sigma\})$. We conclude that $Var(E_j\sigma) \subseteq Var(\{x_1\sigma, \ldots, x_{i-1}\sigma\} \cup \{y\sigma\}_{y\in l_x} \cup \{t_{i+1}\sigma, \ldots, t_{j-1}\sigma\})$, and hence, the application of *Apply* rule on $\mathcal{C}'$ outputs a modified constraint system.

■

We prove below (Lemma 36 and Lemma 37) that the simplification of the modified constraint system $\mathcal{C}'$ using rules from Figure 4.2 terminates in the case of $\mathcal{I}"_{DSKS}$ and $\mathcal{I}"_{DEO}$.

Lemma 36 *Let $\mathcal{C} = (E_1 \rhd t_1, \ldots, E_n \rhd t_n)$ be a modified $\mathcal{I}$-constraint system. The application of transformation rules of the* algorithm *on $\mathcal{C}$ using $\mathcal{L}"_{\mathcal{DSKS}}$ rules terminates.*

PROOF.

Let nbv($\mathcal{C}$) be the number of variables in $\mathcal{C}$, and $\mathcal{M}(\mathcal{C})$ denotes the multiset of the right-hand side of deduction constraints in $\mathcal{C}$. Let us prove that after any application of a transformation rule on a modified $\mathcal{I}$-constraint system $\mathcal{C} = (\mathcal{C}_\alpha, E \rhd t)$ (where $\mathcal{C}_\alpha$ is in solved form), either $nbv(\mathcal{C})$ decreases strictly, or the identity substitution is applied on $\mathcal{C}$ during the transformation and $\mathcal{M}(\mathcal{C})$ strictly decreases.

The first point will ensure that after some point in a sequence of transformations the number of variables will be stable, and thus from this point on $\mathcal{M}(\mathcal{C})$ will strictly decrease. The fact that no more unification will be applied and that the extension of the subterm ordering on multisets is well-founded will then imply that there is only a finite sequence of different modified $\mathcal{I}$-constraint systems, and thereby the termination of the constraint solving algorithm.

This fact is obvious if the Unif rule is applied, since it amounts to the unification of two subterms of $\mathcal{C}$. It is then well-known that if the two subterms are not syntactically equal, the number of variables in their most general unifier is strictly less than the union of their variables, which is included in $Var(\mathcal{C})$. If they are syntactically equal, then no substitution is applied, and thus denoting $\mathcal{C}'$ the result of the transformation, we have $\mathcal{M}(\mathcal{C}) = \mathcal{M}(\mathcal{C}') \cup \{t\}$, and thus $\mathcal{M}(\mathcal{C}') < \mathcal{M}(\mathcal{C})$.

Let us now consider the case of the Apply rule, and let $\mathcal{C}'$ be the obtained modified $\mathcal{I}$-constraint system. If the underlying intruder deduction rule is in $\mathcal{L}_{\mathcal{DSKS}}$, the fact that t is not a variable implies that the variables of the right-hand side of the rule will be instantiated by the strict maximal subterms $t_1, \ldots, t_k$ of t. We will thus have:

$$\mathcal{M}(\mathcal{C}') = \mathcal{M}(\mathcal{C}) \cup \{t_1, \ldots, t_n\} \setminus \{t\}$$

and thus $\mathcal{M}(\mathcal{C}') < \mathcal{M}(\mathcal{C})$.

It now suffices to prove the Lemma for the two rules in $\mathcal{L}"_{\mathcal{DSKS}} \setminus \mathcal{L}_{\mathcal{DSKS}}$:

rule $x, Sk(y) \rightarrow sig(x, Sk(y))$**:** The substitution σ applied is the most general unifier of the unification system $\left\{sig(x, Sk(y)) \stackrel{?}{=}_\emptyset t, Sk(y) \stackrel{?}{=}_\emptyset u\right\}$ for some $u \in E$. Since this is syntactic unification and since we can assume neither u (by Lemma 9) nor t (by definition of the Apply rule) are variables, we must have $u = Sk(u')$ and $t = sig(t_1, t_2)$. The second equation thus yields $y = u'$, with $u' \in Sub(\mathcal{C})$. Replacing in the first equation, σ is the most general unifier of the equation $sig(x, Sk(u')) \stackrel{?}{=}_\emptyset sig(t_1, t_2)$, which reduces into the set of equations $\left\{x \stackrel{?}{=}_\emptyset t_1, Sk(u') \stackrel{?}{=}_\emptyset t_2\right\}$. The first equation implies that x is instantiated by a strict subterm t_1 of t. If the second equation is trivial we have $\mathcal{M}(\mathcal{C}') = \mathcal{M}(\mathcal{C}) \cup \{t_1\} \setminus \{t\}$, and thus $\mathcal{M}(\mathcal{C}') < \mathcal{M}(\mathcal{C})$. Otherwise, since $Var(Sk(u')) \cup Var(t_2) \subseteq Var(\mathcal{C})$ we have $nbv(\mathcal{C}') < nbv(\mathcal{C})$.

rule $x, Sk'(Pk(y), sig(x, Sk(y))) \rightarrow sig(x, Sk(y))$**:** The substitution σ applied is the most general unifier of the unification system $\left\{sig(x, Sk(y)) \stackrel{?}{=}_\emptyset t, Sk'(Pk(y), sig(x, Sk(y))) \stackrel{?}{=}_\emptyset u\right\}$ for some $u \in E$. Since this is syntactic unification and since we can assume neither u (by Lemma 9) nor t (by definition of the Apply rule) are variables, we must have $u = Sk'(u'_1, u'_2)$ and $t = sig(t_1, t_2)$. If σ is the identity on $\mathcal{C}$, we are done, since in this case we have $\mathcal{M}(\mathcal{C}') = \mathcal{M}(\mathcal{C}) \cup \{t_1\} \setminus \{t\}$ and thus $\mathcal{M}(\mathcal{C}') < \mathcal{M}(\mathcal{C})$. Otherwise let us examine how the unification system is solved. It is first transformed into:

$$\left\{x \stackrel{?}{=}_\emptyset t_1, Sk(y) \stackrel{?}{=}_\emptyset t_2, Pk(y) \stackrel{?}{=}_\emptyset u'_1, sig(x, Sk(y)) \stackrel{?}{=}_\emptyset u'_2\right\}$$

Resolving the first equation yields (note that $x \notin Var(\mathcal{C})$):

$$\left\{Sk(y) \stackrel{?}{=}_\emptyset t_2, Pk(y) \stackrel{?}{=}_\emptyset u'_1, sig(t_1, Sk(y)) \stackrel{?}{=}_\emptyset u'_2\right\}$$

Let us consider two cases, depending on whether both u'_1 and t_2 are variables:

- If they are both variables, then solving the first equation removes t_2 from $Var(\mathcal{C})$ but adds a variable y. The second equation will also remove u'_1, but since the variable y is already present, it will not add another variable. Since $Pk(y)$ and $Sk(y)$ are not unifiable, we note that we must have $t_2 \neq u'_1$, and thus we have removed two variables and added one by solving the two first equations. The remaining equation contains only variables of the "intermediate" modified $\mathcal{I}$-constraint system, and thus will not add any new variable. In conclusion, in this case, the number of variables of $\mathcal{C}$ decreases by at least 1.
- If t_2 is not a variable, and thus $t_2 = Sk(t'_2)$, with $t'_2 \in sub(\mathcal{C})$. Resolving the first equation and injecting the solution in the remaining equations yields the unification system:

$$\left\{ Pk(t'_2) \stackrel{?}{=}_{\emptyset} u'_1, sig(t_1, Sk(t'_2)) \stackrel{?}{=}_{\emptyset} u'_2 \right\}$$

  Note that up to this point the substitution σ that we built does not affect any variable of $\mathcal{C}$. If this remaining unification system is trivial, then the substitution applied on $\mathcal{C}$ is the identity, we are done (see above). Otherwise, since all the variables in this system are in $Var(\mathcal{C})$, it strictly reduces $nbv(\mathcal{C})$. This terminates the proof of this case.

Thus, if this rule is applied, either no substitution is applied on $\mathcal{C}$ and $\mathcal{M}(\mathcal{C})$ strictly decreases, or the number of variables in the resulting modified $\mathcal{I}$-constraint system $\mathcal{C}'$ is strictly smaller than the number of variables in $\mathcal{C}$.

■

Lemma 37 *Let $\mathcal{C} = (E_1 \rhd t_1, \ldots, E_n \rhd t_n)$ be a modified $\mathcal{I}$-constraint system. The application of transformation rules of the* algorithm *on $\mathcal{C}$ using $\mathcal{L}$"$_{\mathcal{DEO}}$ rules terminates.*

PROOF.

Let $\mathcal{C} = (E_1 \rhd t_1, \ldots, E_n \rhd t_n)$ be a modified $\mathcal{I}$-constraint system not in solved form and let the complexity of $\mathcal{C}$ be a couple ordered lexicographically with the following components:

- $nbv(\mathcal{C})$, the number of distinct variables in $\mathcal{C}$,
- $\mathcal{M}(\mathcal{C})$ the multiset of the right-hand side of deduction constraints in $\mathcal{C}$.

We have to show that each rule reduces the complexity. The fact is obvious if the $Unif$ rule is applied, since it amounts to the unification of two subterms of $\mathcal{C}$. If is then well-known that if two subterms are not syntactically equal, then the number of variables in their most general unifier is strictly less than the union of their variables, which is included in $Var(\mathcal{C})$. If their are syntactically equal, then

no substitution is applied, and thus denoting $\mathcal{C}'$ the result of transformation, we have $\mathcal{M}(\mathcal{C}') < \mathcal{M}(\mathcal{C})$.

Let us now consider the case of $Apply$ rule, and let $\mathcal{C}'$ be the obtained modified $\mathcal{I}$-constraint system. If the underlying intruder deduction rule is in $\mathcal{L}_{DEO}$, the fact that t is not a variable implies that the right-hand side of the rule will be instantiated by the strict maximal subterms $t_1, \ldots, t_k$ of t. we will thus have $\mathcal{M}(\mathcal{C}') = \mathcal{M}(\mathcal{C}) \cup \{t_1, \ldots, t_k\} \setminus \{t\}$ and thus $\mathcal{M}(\mathcal{C}') < \mathcal{M}(\mathcal{C})$.

It is now suffices to prove the Lemma for the rule in $\mathcal{L}"_{\mathcal{DEO}} \setminus \mathcal{L}_{\mathcal{DEO}}$:

the applied rule is: $f(Pk(y), sig(x, Sk(y))), Sk"(Pk(y), sig(x, Sk(y))) \rightarrow sig(x, Sk(y))$.

The substitution σ is the most general unifier of the unification system $\left\{t \stackrel{?}{=}_\emptyset sig(x, Sk(y)), e_1 \stackrel{?}{=}_\emptyset f(Pk(y), sig(x, Sk(y))), e_2 \stackrel{?}{=}_\emptyset Sk"(Pk(y), sig(x, Sk(y)))\right\}$ for some $e_1,\ e_2 \in E.$ since it is syntactic unification and since we can assume neither e_1, neither e_2 (by Lemma 9) nor t (by definition of the Apply rule) are variables, we must have $t = sig(t_1, t_2),\ e_1 = f(v_1, v_2)$, and $e_2 = Sk"(v_3, v_4)$.

- If $t_2 \in \mathcal{X}$, we have $\sigma(x) = t_1,\ \sigma(t_2) = Sk(y)$ and the unification system is then transformed into:
  $\left\{v_1 \stackrel{?}{=}_\emptyset Pk(y), v_2 \stackrel{?}{=}_\emptyset sig(t_1, Sk(y)), v_3 \stackrel{?}{=}_\emptyset Pk(y), v_4 \stackrel{?}{=}_\emptyset sig(t_1, Sk(y))\right\}$.
  By the fact that t_3 is replaced by y, $x, y \notin Var(\mathcal{C})$, and the number of variables in σ is strictly less than the union of variables of the unification system, we deduce that $nbv(\mathcal{C}') < nbv(\mathcal{C})$.

- If $t_2 \notin \mathcal{X}$ then $t_2 = Sk(t_3)$. We have $\sigma(x) = t_1,\ \sigma y = t_3$ and the unification system is then transformed into:
  $\left\{v_1 \stackrel{?}{=}_\emptyset Pk(t_3), v_2 \stackrel{?}{=}_\emptyset sig(t_1, Sk(t_3)), v_3 \stackrel{?}{=}_\emptyset Pk(t_3), v_4 \stackrel{?}{=}_\emptyset sig(t_1, Sk(t_3))\right\}$.
  If the unification system is obvious, that is σ is the identity substitution, we have $\mathcal{C}' = \mathcal{C} \setminus (E \rhd t)$, and then $\mathcal{M}(\mathcal{C}') = \mathcal{M}(\mathcal{C}) \setminus t$ which implies that $\mathcal{M}(\mathcal{C}') < \mathcal{M}(\mathcal{C})$. Else, we have $nbv(\mathcal{C}') < nbv(\mathcal{C})$.

This concludes the proof. ■

Lemma 38 *If $\mathcal{C} = (E_1 \rhd t_1, \ldots, E_n \rhd t_n)$ is a modified $\mathcal{I}$-constraint system satisfied by a substitution σ with respect to $\mathcal{I}"$ ($\sigma \models_{\mathcal{I}"} \mathcal{C}$), then it can be transformed into a constraint system in solved form by the rules of Figure 4.2.*

PROOF.

Let $\mathcal{C} = (E_1 \rhd t_1, \ldots, E_n \rhd t_n)$ be a modified $\mathcal{I}$-constraint system not in solved form and let i be the smallest integer such that $t_i \notin \mathcal{X}$, then $\mathcal{C} = (\mathcal{C}_\alpha, E_i \rhd t_i, \mathcal{C}_\beta)$ where $\mathcal{C}_\alpha$ is in solved form, and $t \notin \mathcal{X}$. Let σ be a substitution such that

$\sigma \models_{\mathcal{I}"} \mathcal{C}$, and let us prove that $\mathcal{C}$ can be reduced to another satisfiable modified $\mathcal{I}$-constraint system $\mathcal{C}'$ by applying the transformation rules given in the algorithm. $\sigma \models_{\mathcal{I}"} \mathcal{C}$, then $\sigma \models_{\mathcal{I}"} (\mathcal{C}_\alpha, E_i \setminus \mathcal{X} \rhd t_i, \mathcal{C}_\beta)$ (Lemma 9) and then, $(E_i \setminus \mathcal{X})\sigma \rightarrow^*_{\mathcal{I}_\emptyset} t_i\sigma$. We have two cases:

- If $t_i\sigma \in (E_i \setminus \mathcal{X})\sigma$, there exists a term $u \in (E_i \setminus \mathcal{X})$ such that $u\sigma = t_i\sigma$. Let μ be the most general unifier of u and t_i, then $\sigma = \theta\mu$, and we can simplify $\mathcal{C}$ by applying the first transformation rule Unif, $\mathcal{C} \Longrightarrow \mathcal{C}' = (\mathcal{C}_\alpha\mu, \mathcal{C}_\beta\mu)$. We have $\sigma \models_{\mathcal{I}"} \mathcal{C}_\alpha$ and $\sigma \models_{\mathcal{I}"} \mathcal{C}_\beta$, then $\theta \models_{\mathcal{I}"} \{\mathcal{C}_\alpha\mu, \mathcal{C}_\beta\mu\}$.
- If $t_i\sigma \notin (E_i \setminus \mathcal{X})\sigma$ there exists a derivation starting from $(E_i \setminus \mathcal{X})\sigma$ of goal $t_i\sigma$, and then from $E_i\sigma$ of goal $t_i\sigma$. By lemma 32, there exists a derivation starting from $E_i\sigma$ of goal $t_i\sigma$ such that for all steps in the derivation such that $l \rightarrow r$ is the applied rule with the substitution σ, for all $s \in l$ and $s \notin \mathcal{X}$, we have $s\sigma \subseteq E_i\sigma$. This implies that we can reduce $\mathcal{C}$ to $\mathcal{C}'$ by applying the Apply rule of transformation and $\theta \models_{\mathcal{I}"} \mathcal{C}'$.

We deduce that for all satisfiable modified $\mathcal{I}$-constraint systems $\mathcal{C}$ such that $\mathcal{C}$ is not in solved form, $\mathcal{C}$ can be reduced to another satisfiable modified $\mathcal{I}$-constraint system $\mathcal{C}'$ by applying the transformation rules. When applying the transformation rules to a modified $\mathcal{I}$-constraint system, we reduce its complexity (Lemmas 36 and 37), this implies that when we reduce $\mathcal{C}$, we will obtain at some step a satisfiable modified $\mathcal{I}$-constraint system which can not be reducible, this modified $\mathcal{I}$-constraint system is in solved form. This concludes the proof. ■

Lemma 39 *Let $\mathcal{C}$ and $\mathcal{C}'$ be two modified $\mathcal{I}"$-constraint systems such that $\mathcal{C}'$ is obtained from $\mathcal{C}$ by applying a transformation rule from figure 4.2. If $\mathcal{C}'$ is $\mathcal{I}"$-satisfiable then so is $\mathcal{C}$.*

PROOF.

Let $\mathcal{C}$ and $\mathcal{C}'$ be two modified $\mathcal{I}"$-constraint systems such that $\mathcal{C}'$ is obtained from $\mathcal{C}$ by applying a transformation rule and suppose that $\mathcal{C}'$ is $\mathcal{I}"$-satisfiable. Let σ' be a solution of $\mathcal{C}'$ and let us prove that $\mathcal{C}$ is $\mathcal{I}"$-satisfiable. Since a transformation rule can be applied on $\mathcal{C}$, $\mathcal{C}$ can't be in solved form. Suppose that $\mathcal{C} = (\mathcal{C}_\alpha, E \rhd t, \mathcal{C}_\beta)$ where $\mathcal{C}_\alpha$ is in solved form and $t \notin \mathcal{X}$.

- If $\mathcal{C}'$ is obtained from $\mathcal{C}$ by applying *Unif* rule then, there exists a term $u \in E \setminus \mathcal{X}$ such that u and t are syntactically unifiable. Let μ be the most general $\emptyset$-unifier then $\mathcal{C}' = (\mathcal{C}_\alpha\mu, \mathcal{C}_\beta\mu)$. Since $\sigma' \models_{\mathcal{I}"} \mathcal{C}'$, we have $\sigma'\mu \models_{\mathcal{I}"} (\mathcal{C}_\alpha, \mathcal{C}_\beta)$ and by the fact that μ is the most general $\emptyset$-unifier of t and a term in E we have $\sigma'\mu \models_{\mathcal{I}"} E \rhd t$. We deduce that $\sigma'\mu \models_{\mathcal{I}"} \mathcal{C}$.
- If $\mathcal{C}'$ is obtained from $\mathcal{C}$ by applying *Apply* then there exists a rule $l_x, l_1, \ldots, l_n \rightarrow r \in \mathcal{L}"$, a set of terms $e_1, \ldots, e_n$ in $E \setminus \mathcal{X}$ such that

$\left\{r \stackrel{?}{=}_{\emptyset} t, (l_i \stackrel{?}{=}_{\emptyset} e_i)_{1\leq i\leq n}\right\}$ has solution. Let μ be the most general $\emptyset$-unifier. $\mathcal{C}' = (\mathcal{C}_\alpha, (E \rhd x)_{x\in l_x}, \mathcal{C}_\beta)\mu$. Since $\sigma' \models_{\mathcal{I}"} \mathcal{C}'$ and by definition of μ, we have $\sigma'\mu \models_{\mathcal{I}"} \mathcal{C}$.

■

From Lemmas 33, 34, 35, 38, 39 and Lemma 36 (respectively Lemma 37), we deduce respectively the following two corollaries:

Corollary 1 *The $\mathcal{I}_{DSKS}$-reachability problem is decidable.*

Corollary 2 *The $\mathcal{I}_{DEO}$-reachability problem is decidable.*

4.5 Conclusion

In this chapter, we study the constructive exclusive ownership vulnerability and the destructive exclusive ownership vulnerability properties for digital signature schemes, and we show the decidability of the insecurity problem for the two classes of cryptographic protocols using signature schemes vulnerable respectively to constructive exclusive ownership and destructive exclusive ownership properties: we give an algorithm that solves $\mathcal{I}_{DSKS}$- and $\mathcal{I}_{DEO}$-reachability problems. This algorithm can be summarised as follows: first, we saturate $\mathcal{I}_{DSKS}$ and $\mathcal{I}_{DEO}$ intruder deduction rules, this saturation terminates, and yields two new intruder deduction systems $\mathcal{I}"_{DSKS}$ and $\mathcal{I}"_{DEO}$. Second, we simplify $\mathcal{I}_{DSKS}$ and $\mathcal{I}_{DEO}$ constraint systems, using two steps, in order to get respectively a modified $\mathcal{I}"_{DSKS}$ and $\mathcal{I}"_{DEO}$ constraint system in solved form. This simplification is sound and complete. We proved also the termination of this simplification, which is obtained due to the termination of saturation, and to the fact that each simplification step reduces the complexity of the constraint system on which the simplification is applied.

Unfortunately, the termination of such procedure is specific to our cases of cryptographic protocols.

In [95], S. Delaune and F. Jacquemard showed that the $\mathcal{I}$-reachability problem is decidable in non-deterministic polynomial time providing that the equational theory is generated by a convergent public-collapsing rewrite system. We remark that both h_{DSKS} and h_{DEO} are generated by convergent public-collapsing rewrite systems, and hence, the result of [95] can be applied on our classes of cryptographic protocols. We also remark that in [31], M. Baudet obtained a decidability result to the reachability problem for the class of subterm convergent equational theory. Since both h_{DSKS} and h_{DEO} are subterm convergent equational thoeries, we can apply the result of [31] to our classes of cryptographic protocols. The method we used in this chapter is different than the methods used in [95, 31], and can be generalised to other classes of deduction systems than can not be considered by [95, 31].

In the next chapter, we generalise this decidability result to the class of cryptographic protocols that use cryptographic primitives represented by equational theories generated by convergent rewrite systems having the finite variant property. That decidability result is obtained by generalising the proofs of this chapter: we show that the termination of the saturation is sufficient to obtain the decidability of the ground reachability problems, and we add another condition to obtain the decidability of the non ground reachability problems.

Chapter 5

Decidability results for saturated deduction systems

In this chapter, we consider the class of cryptographic protocols where the cryptographic primitives are represented by equational theories generated by convergent rewrite systems satisfying the *finite variant property*, which symbolically has been introduced in [86]. *The finite variant property* allows one to compute all possible normal forms of the instances of a term t. Such a property is claimed to be a key property for decidability results in cryptographic protocols verification in presence of algebraic properties [80], and many common equational theories have been proved to have this property, for example, the Dolev-Yao theory with explicit destructors, the Abelian group theory and others. A first contribution on this chapter is the decidability of the ground reachability problems for our class of deduction systems. Following the description given in Chapter 2, we employ the finite variant property to reduce reachability problems modulo an equational theory to reachability problems modulo the empty theory. We then partially compute a transitive closure of the possible deductions. We prove that the termination of this computation implies the decidability of the ground reachability problems. We conjecture that the overall construction amounts to proving that the deduction system is F-local [40]. We then give a new criterion that permits us to reduce general reachability problems to ground reachability problems. This criterion is based on counting the number of variables in a reachability problem before and after a deduction is guessed, and is a generalisation of the one employed for the specific case of the DSKS intruder model. The intuition behind this criterion is that a deduction rule has to provide more relations between existing fact than it introduces new unknown. We give an example showing that such an

additional criterion is needed, in the sense that there exists deduction systems on which the saturation algorithm terminates, but for which the general reachability problems are undecidable. Another contribution of this chapter is a decidability result of the ground reachability problems for the theory of blind signature [136] using the initial definition of subterm introduced in [10, 31], a similar result was given in [9] using an extended definition of subterm. In [91], another decidability result has been obtained for the class of cryptographic protocols with blind signature schemes. This result has been obtained using a different technique than the one followed in this chapter: in [91], the authors showed the decidability of a fragment of first order logic and used this result to obtain the decidability for the class of cryptographic protocols using blind signature schemes. In addition we give a decidability result to the general reachability problems for a class of subterm convergent equational theories, while a more general result was given in [31], the proof given in this chapter for our special case is much shorter.

Outline. In section 5.1, we introduce the basic notions we use in this chapter. We introduce the finite variant property in section 5.2.1, and in section 5.2.2 we give some examples of equational theories having finite variant property. We introduce the saturation algorithm in Section 5.3.1 and show its properties in section 5.3.2. We give in section 5.4 an algorithm to solve reachability problems. We show the decidability of the ground reachability problem in section 5.5.1, and the decidability of the general reachability problem in section 5.5.3. Some application of our results are shown in section 5.6. We show in section 5.7 the decidability of the ground reachability problems for the blind signature theory, and in section 5.8, we show the decidability of the ground reachability problems for the subterm convergent theories.

5.1 The model

To analyse cryptographic protocols, we follow in this chapter the symbolic model described in Chapter 2. To this end, we assume an infinite set of variables $\mathcal{X}$, an infinite set of constants $\mathcal{C}$, a set of function symbols $\mathcal{F}$. In addition to what is already intoducted in Chapter 2, we make use here of some additional notions that we will show next. Given an equational theory $\mathcal{H}'$, and rewrite system $\mathcal{R}$, *rewriting modulo* $\mathcal{H}'$, also called *equational rewriting*, is the relation $\rightarrow_{\mathcal{H}'\setminus\mathcal{R}}$ defined as follows: given two terms s, t, we have $s \rightarrow_{\mathcal{H}'\setminus\mathcal{R}} t$ if and only if there

exists a position $p \in Pos(s)$ such that $s_{|p} =_{\mathcal{H}'} l\sigma$ and $t = s[r\sigma]_p$ for some substitution σ and rule $l \to r \in \mathcal{R}$.

A rewrite system $\mathcal{R}$ is $\mathcal{H}'$-confluent if and only if for any terms t, u, v such that $t \to^*_{\mathcal{H}\backslash\mathcal{R}} u$ and $t \to^*_{\mathcal{H}\backslash\mathcal{R}} v$, there exists a term w verifying $u \to^*_{\mathcal{H}\backslash\mathcal{R}} w$ and $v \to^*_{\mathcal{H}\backslash\mathcal{R}} w$. it said to be $\mathcal{H}'$-convergent if, in addition, $=_{\mathcal{H}'} \circ \to_{\mathcal{R}} \circ =_{\mathcal{H}'}$ is well-founded. The notion of $\mathcal{R}$-normal form defined in Chapter 2 is extended to $\mathcal{H}'\backslash\mathcal{R}$-normal form as expected. These notions are initially given in [100].

We assume in the following two equational theories $\mathcal{H}$ and $\mathcal{H}'$, and an $\mathcal{H}'$-convergent rewrite system $\mathcal{R}$ generating $\mathcal{H}$. Furthermore, we assume a complete simplification order $\succ$ over $\mathcal{T}(\mathcal{F}, \mathcal{X})$, that is $\succ$ is well-founded, monotone, stable, subterm, and total over ground terms ($\mathcal{T}(\mathcal{F})$). We extend the strict order $\succ$ over $\mathcal{T}(\mathcal{F}, \mathcal{X})$ to the order $\succeq$ over $\mathcal{T}(\mathcal{F}, \mathcal{X})$ as follows: we have $t_1 \succeq t_2$ if and only if $t_1 \succ t_2$ or $t_1 = t_2$ for any terms $t_1, t_2 \in \mathcal{T}(\mathcal{F}, \mathcal{X})$.

5.2 Finite variant property

In what follows, we make use of "AC" to means the associativity and commutativity axioms.

5.2.1 Definition of finite variant property

In [84, 87], the authors came twice through the following problem:

Given an AC-convergent rewrite system $\mathcal{R}$, is it possible (and how) to compute from any term t a finite set of instances $t\sigma_1, \ldots, t\sigma_n$ such that

$$\{(t\sigma)\downarrow \mid \sigma \in \Sigma\} = \bigcup_{i=1}^{n} \{(t\sigma_i)\downarrow\theta \mid \theta \in \Sigma\}$$

where Σ is the set of normalised substitutions and $(s)\downarrow$ is the AC-normal form of s with respect to $\mathcal{R}$.

In other words, the reductions in $t\sigma$ only depend on reductions in finitely many (fixed) instances of t. This is typically what it is called in [86] the *finite variant property*: compute in advance all possible normal forms of an instance of t, independently of that instance.

Definition 45 *($\mathcal{H}$-variants modulo $\mathcal{H}'$) Given two terms $t, t' \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, t' is an $\mathcal{H}$-variant of t if there is a substitution θ such that $t\theta =_{\mathcal{H}} t'$. A complete set of $\mathcal{H}$-variants modulo $\mathcal{H}'$ of t is a set V of $\mathcal{H}$-variants of t such that, for every substitution σ, there is a term $t' \in V$ and a substitution θ such that $(t\sigma)\downarrow_{\mathcal{R}} =_{\mathcal{H}'} t'\theta$.*

Definition 46 *(finite variant property) The pair of equational theories $(\mathcal{H}, \mathcal{H}')$ has the finite variant property if for every term t, we can effectively compute a finite complete set of $\mathcal{H}$-variants modulo $\mathcal{H}'$. Sometimes and for simplicity, we will simply say variants and complete set of variants when $\mathcal{H}$ and $\mathcal{H}'$ are clear from the context.*

When $\mathcal{R}$ is a $\mathcal{H}'$-convergent rewrite system generating $\mathcal{H}$, we have that $(\mathcal{R}, \mathcal{H}')$ satisfies *the finite variant property* if and only if $(\mathcal{H}, \mathcal{H}')$ satisfies the *finite variant property*.

Definition 47 *$(\mathcal{R}, \mathcal{H}')$ satisfies the finite variant property if for any term t, there is a finite set of variants $t_1, \ldots, t_n$, effectively computable, such that, for every substitution σ, there is an index i and a substitution θ such that $(t\sigma)\downarrow_{\mathcal{H}' \setminus \mathcal{R}} =_{\mathcal{H}'} t_i\theta$.*

In [86], the authors showed that if $(\mathcal{R}, \mathcal{H}')$ has the finite variant property, we may not only compute in advance some instances t_i of t such that $(t\sigma)\downarrow$ is always an instance modulo $\mathcal{H}'$ of some t_i , but actually compute in advance substitutions θ_i such that $t_i = (t\theta_i)\downarrow$ is a complete set of variants and every normalised substitution can be factorised through θ_i. This result is summarised by the following lemma.

Lemma 40 *$(\mathcal{R}, \mathcal{H}')$ has the finite variant property if and only if for any term t, there is a finite set of substitutions $\Sigma(t)$ such that for any substitution σ, there exists a substitution $\theta \in \Sigma(t)$, and a substitution τ verifying $(\sigma)\downarrow =_{\mathcal{H}'} \theta\tau$ and $(t\sigma)\downarrow =_{\mathcal{H}'} (t\theta)\downarrow\tau$*

In [86], S. Delaune and H. Comon-Lundh define the *boundness property* as follows:

Definition 48 *(boundedness property) $(\mathcal{R}, \mathcal{H}')$ satisfies the boundedness property if for every term t, there exists an integer n such that for every normalised substitution σ, the normal form of $t\sigma$ is reachable by a derivation whose length can be bounded by n:*

$$\forall t,\ \exists n,\ \forall \sigma,\ t((\sigma)\downarrow) \rightarrow^{\leq n}_{\mathcal{H}' \setminus \mathcal{R}} (t\sigma)\downarrow$$

and then, S. Delaune and H. Comon-Lundh showed the relationships between the boundness property and the finite variant property by proving the following theorem.

Theorem 9 *$(\mathcal{R}, \mathcal{H}')$ satisfies the boundedness property if and only if $(\mathcal{R}, \mathcal{H}')$ satisfies the finite variant property.*

5.2.2 Equational theories having finite variant property

In [86], S. Delaune and H. Comon-Lundh showed that for any equational theory $\mathcal{H}$ generated by a ($\emptyset$-) convergent rewrite system $\mathcal{R}$, if any $\mathcal{R}$-basic narrowing

starting from any term t terminates then $(\mathcal{R}, \emptyset)$ satisfies the boundness property and hence, by theorem 9, $(\mathcal{R}, \emptyset)$ satisfies the finite variant property.

Since the termination of basic narrowing derivations starting from right hand sides of rules of any convergent rewrite system $\mathcal{R}$ implies the termination of basic narrowing derivations starting from any term t with respect to $\mathcal{R}$ [123] (Chapter 2, theorem 1), the result obtained in [86] allows us to conclude that finite variant property holds for any equational theory $\mathcal{H}$ generated by a convergent rewriting system $\mathcal{R}$ such that any basic narrowing derivation starting from right hand sides of rules of $\mathcal{R}$ terminates.

Furthermore, it is shown in [86] that the finite variant property holds for any equational theory $\mathcal{H}$ generated by a convergent optimally reducing [162] rewriting system $\mathcal{R}$.

In practice, many equational theories, which are relevant to cryptographic protocols, have the finite variant property. We give in the follow some of them:

Dolev-Yao theory with explicit destructors The Dolev-Yao theory with explicit destructors is given by the following equational theory:

$$\mathcal{H}_{DY} : \begin{cases} \pi_1(< x, y >) = x \\ \pi_2(< x, y >) = y dec^s(enc^s(x, y), y) = x \\ dec^p(enc^p(x, y), y^{-1}) = x \\ dec^p(enc^p(x, y^{-1}), y) = x \end{cases}$$

The application of Knuth-Bendix completion procedure [131] on $\mathcal{H}_{DY}$ terminates successfully and outputs the rewrite system $\mathcal{R}_{DY}$

$$\mathcal{R}_{DY} : \begin{cases} \pi_1(< x, y >) \rightarrow x \\ \pi_2(< x, y >) \rightarrow y \\ dec^s(enc^s(x, y), y) \rightarrow x \\ dec^p(enc^p(x, y), y^{-1}) \rightarrow x \\ dec^p(enc^p(x, y^{-1}), y) \rightarrow x \end{cases}$$

Following the results obtained in [21], $\mathcal{R}_{DY}$ is a convergent rewrite system generating $\mathcal{H}_{DY}$. Furthermore, the right hand sides in all rules in $\mathcal{R}_{DY}$ are not basic narrowable and hence, we conclude that the finite variant property holds for $\mathcal{H}_{DY}$.

Exclusive Or theory The Exclusive Or theory $\mathcal{H}_{EO}$ given by the equations

$$\begin{aligned} x + x &= 0 \\ x + 0 &= x \\ x + x + y &= y \end{aligned}$$

and the associativity and commutativity axioms for $+$. It is proven that $\mathcal{R}_{EO} = \{x + x \rightarrow 0, x + 0 \rightarrow x, x + x + y \rightarrow y\}$ is a AC-convergent rewrite

system generating the exclusive or theory, and such that $(\mathcal{R}, AC)$ satisfies the finite variant property [86].

Abelian group theory The Abelian group theory $\mathcal{H}_{AG}$ is defined by the following set of equations:

$$\mathcal{R}_{AG} : \begin{cases} x * (y * z) = (x * y) * z \\ x * y = y * x \\ x * x^{-1} = 1 \\ x * 1 = x \end{cases}$$

The following rewrite system

$$\mathcal{R}_{AG} : \begin{cases} x * 1 \to x \\ 1^{-1} \to 1 \\ x * x^{-1} \to 1 \\ x^{-1} * y^{-1} \to (x * y)^{-1} \\ (x * y)^{-1} * y \to x^{-1} \\ (x^{-1})^{-1} \to x \\ (x^{-1} * y)^{-1} \to x * y^{-1} \\ x * (x^{-1} * y) \to y \\ x^{-1} * (y^{-1} * z) \to (x * y)^{-1} * z \\ (x * y)^{-1} * (y * z) \to x^{-1} * z \end{cases}$$

is an AC-convergent system for abelian group theory [124], and $(\mathcal{R}_{AG}, AC)$ satisfies the finite variant property [86].

In this chapter, we consider only equational theories $\mathcal{H}$ generated by convergent rewrite system $\mathcal{R}$, or more formally, we consider only equational theories $\mathcal{H}$ for which there exists a rewrite system $\mathcal{R}$ such that $\mathcal{R}$ is an $\emptyset$-convergent rewrite system generating $\mathcal{H}$.

From now on, and for the aim of simplicity, we call an equational theory $\mathcal{H}$ has the *finite variant property* if $\mathcal{H}$ is generated by a convergent rewrite system $\mathcal{R}$ such that $(\mathcal{R}, \emptyset)$ has the finite variant property, that is for any term t, one can compute a finite set of substitutions $\Sigma(t) = \{\theta_1, \ldots, \theta_n\}$ such that, for any substitution σ there exists $i \in \{1, \ldots, n\}$ and a substitution τ such that $(\sigma)\downarrow = \theta_i\tau$ and $(t\sigma)\downarrow = (t\theta_i)\downarrow\tau$. We call the substitutions θ_i *variant substitutions* of t and the terms $(t\theta_i)\downarrow$ *variants* of t. It is easy to see that the variant substitutions of any term t are in normal form.

The finite variant property ensures that it is possible to compute a complete set of most general unifiers between two terms t and t'. Indeed, it suffices to compute for these two terms the respective sets of variant substitutions $\{\theta_i\}_{i\in\{1,\ldots,m\}}$, $\{\theta'_j\}_{j\in\{1,\ldots,n\}}$, and to (try to) unify in the empty theory every pair of

terms $(t\theta_i)\downarrow = (t'\theta'_j)\downarrow$. Based on this technique, a complete, sound and terminating $\mathcal{H}$-unification algorithm has been given in Section 2.1.8 (Chapter 2) when $\mathcal{H}$ is an equational theory generated by a convergent rewrite system having the finite variant property.

5.3 Saturation of intruder deduction rules

In the rest of this chapter we assume that $\mathcal{F}$ is a signature, $\mathcal{H}$ represents an equational theory generated by a convergent rewriting system $\mathcal{R}$, and such that $(\mathcal{R}, \emptyset)$ satisfies the finite variant property. Furthermore, we assume $\mathcal{I}_0 = \langle \mathcal{F}, \mathcal{T}_{\mathcal{I}_0}, \mathcal{H} \rangle$ to be an intruder deduction system (Definition 16, Chapter 2), $\mathcal{L}_0 = \mathcal{L}_{\mathcal{I}_0}$ to be intruder deduction rules associated to $\mathcal{I}_0$, that is, rules in $\mathcal{L}_{\mathcal{I}_0}$ are of the form $x_1, \ldots, x_n \rightarrow f(x_1, \ldots, x_n)$ where f is a public function symbol in $\mathcal{F}$ with arity n.

Definition 49 *(Increasing and decreasing deduction rule) Let $\widetilde{l} \rightarrow r$ be a deduction rule with $\widetilde{l}$ is a set of terms and r is a term. $\widetilde{l} \rightarrow r$ is said to be* decreasing *rule if there is a term $s \in \widetilde{l}$ such that $s \succeq r$ and $\widetilde{l} \rightarrow r$ is said to be* increasing *otherwise.*

From now on, if $\mathcal{L}$ is a set of deduction rules, we denote by $\mathcal{L}_{inc}$ the set of increasing rules in $\mathcal{L}$ and by $\mathcal{L}_{dec}$ the set of decreasing rules in $\mathcal{L}$. By definition of *increasing* and *decreasing* rules, we have $\mathcal{L} = \mathcal{L}_{inc} \cup \mathcal{L}_{dec}$.

Definition 50 *Let E (respectively t) be a set of terms (respectively a term) in normal form and let D be a derivation starting from E of goal t, $D : E = E_0 \rightarrow E_0, t_1 \rightarrow \ldots \rightarrow E_{n-2}, t_{n-1} \rightarrow E_{n-1}, t$. We let $Cons(D)$ be the sequence of terms constructed during the derivation D, $Cons(D) = E_0 \cup \{t_1, \ldots, t_{n-1}\}$.*

Definition 51 *(well-formed derivation) Let E (respectively t) be a set of terms (respectively a term) and let D be a derivation starting from E of goal t, $D : E = E_0 \rightarrow E_0, t_1 \rightarrow \ldots \rightarrow E_{n-2}, t_{n-1} \rightarrow E_{n-1}, t$. The derivation D is said to* well-formed *if for all rules $\widetilde{l} \rightarrow r$ applied with substitution σ, for all $u \in \widetilde{l} \setminus \mathcal{X}$ we have either $u\sigma \in E$ or $u\sigma$ was deduced by a former decreasing rule.*

Definition 52 *(Strongly order locality) Let $\mathcal{I}_1 = \langle \mathcal{F}, \mathcal{T}_{\mathcal{I}_1}, \mathcal{H} \rangle$ be an intruder deduction system (Definition 16), we recall that $\mathcal{T}_{\mathcal{I}_1} = \{f(x_1, \ldots, x_n)$ such that x_i is variable for every i, and $f \in \mathcal{F}_{pub}$ with arity n $\}$. Let $\mathcal{I}_2 = \langle \mathcal{F}, \mathcal{L}_{\mathcal{I}_2}, \emptyset \rangle$ be another intruder deduction system with $\mathcal{L}_{\mathcal{I}_2}$ a set of intruder deduction rules of the form $l_1, \ldots, l_n \rightarrow r$ and l_i, r are terms in $\mathcal{T}(\mathcal{F}, \mathcal{X})$. $\mathcal{I}_2$ follows the definition of intruder deduction systems of [72]. $\mathcal{I}_2$ is said to be $\mathcal{I}_1$-strongly order local if the following three conditions are satisfied:*

Figure 5.1 Saturation algorithm modulo $\mathcal{H}$

Input:

The intruder deduction rules $\mathcal{L}_0$.

Initialisation:

Let $\mathcal{I} = \langle \mathcal{F}, \mathcal{L}, \emptyset \rangle$ be the variant of the intruder deduction system $\mathcal{I}_0$ (Definition 19, Chapter 2). We recall that

$$\mathcal{L} = \bigcup_{\substack{x_1,\ldots,x_n \to f(x_1,\ldots,x_n) \in \mathcal{L}_0 \\ \theta \text{ variant substitution of } f(x_1,\ldots,x_n)}} x_1\theta,\ldots,x_n\theta \to (f(x_1,\ldots,x_n)\theta)\downarrow$$

Step 1.

Start with $\mathcal{L}' = \mathcal{L}$.

Repeat the *closure rule* given below until no more new rule can be added on $\mathcal{L}'$.

$$\frac{\widetilde{l_1} \to r_1 \in \mathcal{L}'_{inc}\,; \quad \widetilde{l_2}, s \to r_2 \in \mathcal{L}'}{\mathcal{L}' \leftarrow \mathcal{L}' \cup \left\{(\widetilde{l_1}, \widetilde{l_2} \to r_2)\sigma\right\}} \quad \begin{array}{c} s \notin \mathcal{X} \\ \sigma = mgu(r_1, s) \end{array}$$

Output:

Output $\mathcal{L}'$.

$Variant$

Let $\mathcal{I}'_1 = \langle \mathcal{F}, \mathcal{L}_{\mathcal{I}'_1}, \emptyset \rangle$ be the variant intruder deduction system of $\mathcal{I}_1$, $\mathcal{L}_{\mathcal{I}'_1} \subseteq \mathcal{L}_{\mathcal{I}_2}$.

SOL_1

For any set of ground terms E in normal form and any ground term t in normal form we have: $t \in \overline{E}^{\mathcal{I}_1}$ if and only if $t \in \overline{E}^{\mathcal{I}_2}$.

SOL_2

For any set of terms E in normal form and any terms t in normal form such that $t \in \overline{E}^{\mathcal{I}_2}$. For all $\mathcal{I}_2$-derivations D starting from E of goal t we have either D is well-formed or there is another $\mathcal{I}_2$-derivation D' starting from E of goal t such that $Cons(D) \subseteq Cons(D')$ and D' is well-formed.

5.3.1 Saturation algorithm

We define in Figure 5.1 the *saturation algorithm modulo $\mathcal{H}$* which takes as input the set of deduction rules $\mathcal{L}_0$ and outputs another set of deduction rules called *a saturated set of deduction rules*.

We define two new deduction systems $\mathcal{I} = \langle \mathcal{F}, \mathcal{L}, \emptyset \rangle$ and $\mathcal{I}' = \langle \mathcal{F}, \mathcal{L}', \emptyset \rangle$. We

remark that $\mathcal{I}_0$ satisfies the definition of intruder deduction system as given in Definition 16 (Chapter 2) and in [73], and the intruder deduction systems $\mathcal{I}, \mathcal{I}'$ satisfy the definition of intruder deduction system as given in [72].

5.3.2 Properties of the saturation

We prove in Lemmas 41 and 42 that $\mathcal{I}'$ is $\mathcal{I}_0$-strongly order local: in fact, the saturation algorithm given in Figure 5.1 implies easily that $\mathcal{I}'$ satisfies $Variant$, Lemma 41 shows that SOL_1 is satisfied and Lemma 42 shows that SOL_2 is satisfied.

Lemma 41 *For any set of ground terms E in normal form and any ground term t in normal form we have: $t \in \overline{E}^{\mathcal{I}_0}$ if and only if $t \in \overline{E}^{\mathcal{I}'}$.*

PROOF.

The direct implication is trivial since $\mathcal{L} \subseteq \mathcal{L}'$. Let us prove the converse implication. In Section 2.1.11 (Chapter 2), we proved that: $E \rightarrow_{\mathcal{I}} F$ implies $E \rightarrow_{\mathcal{I}_0} F$ for any two sets of ground terms E and F in normal form, an hence $t \in \overline{E}^{\mathcal{I}}$ implies $t \in \overline{E}^{\mathcal{I}_0}$ for any set of ground terms in normal form E and any ground term in normal form t. We prove now that $t \in \overline{E}^{\mathcal{I}'}$ implies $t \in \overline{E}^{\mathcal{I}}$ for any set of ground terms in normal form E and any ground term in normal form t.

Assume that there exists a $\mathcal{I}'$-derivation starting from E of goal t. Let us define an arbitrary total order on the rules of $\mathcal{L}$, and we extend this order to rules of $\mathcal{L}' \setminus \mathcal{L}$ as follows: rules of $\mathcal{L}$ are smaller than the rules of $\mathcal{L}' \setminus \mathcal{L}$ and rules of $\mathcal{L}' \setminus \mathcal{L}$ are ordered according to the order of their construction during the saturation. Let $\mathrm{M}(D)$ be the multiset of rules applied in D. Let $\Omega(E,t) = \{D \mid D : E \rightarrow^*_{\mathcal{I}'} F \ni t\}$. By construction, the ordering on rules is total and well-founded, and thus the pre-ordering on derivations in $\Omega(E,t)$ is also total and well-founded. Since $t \in \overline{E}^{\mathcal{I}'}$, we have $\Omega(E,t) \neq \emptyset$, and thus $\mathrm{M}(\Omega(E,t))$ has a minimum element which is reached. Let D be a derivation in $\Omega(E,t)$ having the minimum $\mathrm{M}(D)$, and let us prove that D employs only rules in $\mathcal{L}$. By contradiction, assume that D uses a rule $\widetilde{l} \rightarrow r \in \mathcal{L}' \setminus \mathcal{L}$ applied with a ground substitution σ on a set F. Since $l \rightarrow r \notin \mathcal{L}$, it has been constructed by closure rule. Thus, there exists two rules $\widetilde{l_1} \rightarrow r_1 \in \mathcal{L}'_{inc}$ and $\widetilde{l_2} \rightarrow r_2 \in \mathcal{L}'$, a term $s \in \widetilde{l_2} \setminus \mathcal{X}$ such that s and r_1 are unifiable, $\alpha = mgu(s, r_1)$, $\widetilde{l} = (\widetilde{l_1}, \widetilde{l_2} \setminus s)\alpha$ and $r = r_2\alpha$. Replacing the application of the rule $\widetilde{l} \rightarrow r$ by two steps applying first the rule $\widetilde{l_1} \rightarrow r_1$ and then $\widetilde{l_2} \rightarrow r_2$ yields another derivation D'. Since $\widetilde{l} \rightarrow r$ must have an order bigger than the order of $\widetilde{l_1} \rightarrow r_1$ and $\widetilde{l_2} \rightarrow r_2$ and the last two rules are in $\mathcal{L}'$, we deduce that $D' \in \Omega(E,t)$ and $\mathrm{M}(D') < \mathrm{M}(D)$ which contradicts the minimality of $\mathrm{M}(D)$. ■

The following Corollary is an obvious consequence of Lemma 41.

Corollary 3 *The saturation algorithm given in Figure 5.1 is correct and complete.*

The following lemma is a consequence of the computation of the closure (Figure 5.1). Notice that we do not assume here, nor afterward unless stated, that the saturation terminates.

Lemma 42 *Let E (respectively t) be a set of terms (respectively a term) in normal form such that $t \in \overline{E}^{\mathcal{I}'}$. For all $\mathcal{I}'$-derivations D starting from E of goal t we have either D is well-formed or there is another $\mathcal{I}'$-derivation D' starting from E of goal t such that $Cons(D) = Cons(D')$ and D' is well-formed.*

PROOF.

We have $t \in \overline{E}^{\mathcal{I}'}$ implies that the set $\Omega(E,t)$ of $\mathcal{I}'$-derivations starting from E of goal t is not empty. Let $D \in \Omega(E,t)$, $D : E = E_0 \rightarrow E_1 \rightarrow \ldots \rightarrow E_{n-1}, t$, we denote $\widetilde{l_i} \rightarrow r_i$ the rule applied at step i with the substitution σ_i and suppose that D is not well-formed. Let us (pre-)order derivations in $\Omega(E,t)$ with a measure M such that M(D') for a derivation D' is a multiset of integers constructed as follows: starting with $M(D') = \emptyset$, for all steps k, $1 \leq k \leq n$, for all terms $u \in l_k\sigma_k$ obtained by former increasing rule, add k to $M(D')$. Since this pre-order is well-founded, there exists a derivation $d \in \Omega(E,t)$ such that $M(d)$ is minimum and $Cons(d) = Cons(D)$. Let us prove that d is well-formed. By contradiction, assume that d is not well-formed and let j be the first step in d such that $\widetilde{l_j} \rightarrow r_j$ is the rule applied with substitution σ_j and there is a term $u \in \widetilde{l_j} \setminus \mathcal{X}$ obtained by a former increasing rule, let $\widetilde{l_h} \rightarrow r_h$ be this rule. Since $\widetilde{l_h} \rightarrow r_h \in \mathcal{L}'_{inc}$ and $u \notin \mathcal{X}$, *Closure* can be applied on $\widetilde{l_h} \rightarrow r_h$ and $\widetilde{l_j} \rightarrow r_j$ and the resulting rule can be applied at step j instead of $\widetilde{l_j} \rightarrow r_j$ yielding also E_j. Let d' be the derivation obtained after this replacement, $d' \in \Omega(E,t)$ and $Cons(d') = Cons(d)$. Since $h < j$ and by definition of M, we have $M(d') < M(d)$ which contradicts the minimality of $M(d)$. We deduce that d is well-formed and then we have the lemma. ■

From the definition of the saturation (Figure 5.1), and from Lemma 41 and Lemma 42, we deduce the following corollary.

Corollary 4 *The saturated intruder deduction system $\mathcal{I}'$ is $\mathcal{I}_0$-strongly order local.*

5.4 Algorithm for solving reachability problems

Presentation of the algorithm This section is devoted to the presentation of an algorithm for solving reachability problems and to a proof scheme of its completeness and correctness. In this section, we consider that $\mathcal{I}_1 = \langle \mathcal{F}, \mathcal{T}_{\mathcal{I}_1}, \mathcal{H} \rangle$ is

an intruder deduction system (Definition 16) and $\mathcal{I}_2 = \langle \mathcal{F}, \mathcal{L}_{\mathcal{I}_2}, \emptyset \rangle$ is another intruder deduction system with $\mathcal{L}_{\mathcal{I}_2}$ a set of intruder deduction rules of the form $l_1, \ldots, l_n \rightarrow r$ and l_i, r are terms in $\mathcal{T}(\mathcal{F}, \mathcal{X})$. We also consider that $\mathcal{I}_2$ is $\mathcal{I}_1$-strongly order local. We recall that $\mathcal{L}_{\mathcal{I}_2}$ is partitioned into two disjoint sets of deduction rules $\mathcal{L}_{\mathcal{I}_2 inc}$ and $\mathcal{L}_{\mathcal{I}_2 dec}$ (by definition of *increasing* and *decreasing* rules). The algorithm comprises two steps, and is described in Fig. 5.2

Figure 5.2 Algorithm for solving constraint systems.

Resolution($\mathcal{C}^0$)
We let $\mathcal{C}^0 = ((E_i^0 \vdash v_i^0)_{i \in \{1,\ldots,n\}}, \mathcal{U}^0)$ be an $\mathcal{I}_1$-constraint system.

Step 1: guess of a variant constraint system
Guess a variant $\mathcal{I}_1$-constraint system associated to $\mathcal{C}^0$. Let $\mathcal{C} = (E'_1 \rhd t'_1, \ldots, E'_n \rhd t'_n)$ be the variant $\mathcal{I}_1$-constraint system associated to $\mathcal{C}^0$. $\mathcal{C}$ is constructed from $\mathcal{C}^0$ as given in Definition 28 (Chapter 2).

Step 2.
Apply non-deterministically the transformation rules of Fig. 5.3

Step 3.
If a solved form is reached, return SAT, else return FAIL.

We prove below that there exists a solution to the original $\mathcal{I}_1$-constraint system $\mathcal{C}^0$ if and only if there exists a solution to one of the possible modified constraint systems computed in the first step for the $\mathcal{I}_2$ deduction system.

Lemma 43 *Let $\mathcal{C}^0$ be a $\mathcal{I}_1$-constraint system. If σ is a substitution in normal form such that $\sigma \models_{\mathcal{I}_1} \mathcal{C}^0$, then there exists a modified constraint system $\mathcal{C}$ in the output of Step 1 and a substitution σ' in normal form such that $\sigma' \models_{\mathcal{I}_2} \mathcal{C}$.*

PROOF.
Let $\mathcal{C}^0 = (E_1 \vdash v_1, \ldots, E_n \vdash v_n, \mathcal{U})$ be a $\mathcal{I}_1$-constraint system, and let σ be a substitution in normal form such that $\sigma \models_{\mathcal{I}_1} \mathcal{C}^0$. This implies that there exists a substitution θ in normal form and a substitution τ in normal form such that $\tau \models_{\mathcal{I}_1'} ((E_1\theta)\downarrow \rhd (v_1\theta)\downarrow, \ldots, (E_n\theta)\downarrow \rhd (v_n\theta)\downarrow)$, with $\mathcal{I}'_1$ is a variant intruder deduction system of $\mathcal{I}_1$, and $((E_1\theta)\downarrow \rhd (v_1\theta)\downarrow, \ldots, (E_n\theta)\downarrow \rhd (v_n\theta)\downarrow)$ is a variant $\mathcal{I}_1$-constraint system associated to $\mathcal{C}$ (Lemma 12 at Chapter 2). Since $\mathcal{L}'_{I_1} \subseteq \mathcal{L}_{\mathcal{I}_2}$, we have that $\tau \models_{\mathcal{I}_2} ((E_1\theta)\downarrow \rhd (v_1\theta)\downarrow, \ldots, (E_n\theta)\downarrow \rhd (v_n\theta)\downarrow)$, which concludes the proof. ■

Lemma 44 *Let $\mathcal{C}^0$ (respectively $\mathcal{C}$) be a $\mathcal{I}_1$- (respectively $\mathcal{I}_2$-) constraint system. Assume that $\mathcal{C}$ is obtained from $\mathcal{C}^0$ at the end of Step 1. If $\mathcal{C}$ is $\mathcal{I}_2$-satisfiable then $\mathcal{C}^0$ is $\mathcal{I}_1$-satisfiable.*

PROOF.

Let $\mathcal{C}^0$ (respectively $\mathcal{C}$) be a $\mathcal{I}_1$- (respectively $\mathcal{I}_2$-) constraint system and assume that $\mathcal{C}$ is obtained from $\mathcal{C}^0$ at then end of $Step\ 1$. This implies that $\mathcal{C}^0 = (E_1 \vdash v_1, \ldots, E_n \vdash v_n, \mathcal{U})$ and $\mathcal{C} = ((E_1\theta)\downarrow \rhd (v_1\theta)\downarrow, \ldots, (E_n\theta)\downarrow \rhd (v_n\theta)\downarrow)$ while θ is a substitution in normal form (the construction of θ is shown in Definition 28). Since $\mathcal{C}$ is $\mathcal{I}_2$-satisfiable there exists a normal substitution σ' such that $(v_i\theta)\downarrow\sigma' \in \overline{(E_i\theta)\downarrow\sigma'}^{\mathcal{I}_2}$ and thus $(v_i\theta\sigma')\downarrow \in \overline{(E_i\theta\sigma')\downarrow}^{\mathcal{I}_2}$, which implies that $(v_i\theta\sigma')\downarrow \in \overline{(E_i\theta\sigma')\downarrow}^{\mathcal{I}_1}$ (Lemma 41). Let $\sigma = (\theta\sigma')\downarrow$, we have that $(v_i\sigma)\downarrow \in \overline{(E_i\sigma)\downarrow}^{\mathcal{I}_0}$, and by construction of θ, we have that $\theta \models_{\mathcal{I}_1} \mathcal{U}$, and hence $\sigma \models_{\mathcal{I}_1} \mathcal{U}$ which concludes the proof. ■

Transformation in solved form

In the rest of this chapter, we denote by $l_x, l_1, \ldots, l_n \rightarrow r$ a $\mathcal{L}_{\mathcal{I}_2}$-rule such that l_x is a finite set of variables and $\{l_1, \ldots, l_n\}$ is a finite set of non-variable terms. Unless otherwise specified, $\mathcal{I}_2$ is the intruder deduction system implicit in all notations.

In the rest of this section, we prove a *progress* property: If a satisfiable modified constraint system is not in solved form, then a rule of Fig. 5.3 can be applied on it to yield another satisfiable modified constraint system.

Figure 5.3 System of transformation rules.

Unif :

$$\frac{\mathcal{C}_\alpha, E \rhd t, \mathcal{C}_\beta}{(\mathcal{C}_\alpha, \mathcal{C}_\beta)\sigma} \quad \begin{array}{c} u \in E \setminus \mathcal{X},\ t \notin \mathcal{X}, \\ \sigma = mgu(u, t) \end{array}$$

Reduce 1 :

$$\frac{\mathcal{C}_\alpha, E \rhd t, \mathcal{C}_\beta}{(\mathcal{C}_\alpha, (E \rhd y)_{y \in l_x}, \mathcal{C}_\beta)\sigma} \quad \begin{array}{c} l_x, l_1, \ldots, l_n \rightarrow r \in \mathcal{L}_{\mathcal{I}_2 inc} \text{ and } t \notin \mathcal{X} \\ e_1, \ldots, e_n \in E \setminus \mathcal{X} \text{ and } \sigma = mgu(\left\{e_i \stackrel{?}{=}_\emptyset l_i\right\}_{1 \le i \le n}, r \stackrel{?}{=}_\emptyset t) \end{array}$$

Reduce 2 :

$$\frac{\mathcal{C}_\alpha, E \rhd t, \mathcal{C}_\beta}{(\mathcal{C}_\alpha, (E \rhd y)_{y \in l_x}, E \cup r \rhd t, \mathcal{C}'_\beta)\sigma} \quad \begin{array}{c} l_x, l_1, \ldots, l_n \rightarrow r \in \mathcal{L}_{\mathcal{I}_2 dec} \text{ and } t \notin \mathcal{X} \\ e_1, \ldots, e_n \in E \setminus \mathcal{X} \text{ and } \sigma = mgu(\left\{e_i \stackrel{?}{=}_\emptyset l_i\right\}_{1 \le i \le n}) \\ \mathcal{C}'_\beta \textit{ is obtained from } \mathcal{C}_\beta \textit{ by} \\ \textit{adding } r \textit{ to left hand side of constraints} \end{array}$$

Simplification step. Let $\mathcal{C} = (\mathcal{C}_\alpha, E \rhd t, \mathcal{C}_\beta)$ be a modified constraint system such that $\mathcal{C}_\alpha$ in solved form and $t \notin \mathcal{X}$. If we apply *Reduce 1* (respectively *Reduce 2*) on $\mathcal{C}$ using a rule $l_x, l_1, \ldots, l_n \rightarrow r$ such that there is a variable $x \in l_x \setminus Var(\{l_1, \ldots, l_n, r\})$ then the constraint $E \rhd x$ will be in the obtained modified

constraint system $\mathcal{C}'$ and x does not appear twice in $\mathcal{C}'$. By lemma 10, this constraint can be deleted from $\mathcal{C}'$. As a consequence, we apply a simplification step on the deduction system $\mathcal{L}_{\mathcal{I}_2}$ that eliminates variables $x \in l_x \setminus Var(\{l_1, \ldots, l_n, r\})$ for all rules $l_x, l_1, \ldots, l_n \rightarrow r \in \mathcal{L}_{\mathcal{I}_2}$.

The next Lemma shows that the application of a rule from Figure 5.3 on a modified constraint system outputs a modified constraint system.

Lemma 45 *Let $\mathcal{C}'$ be a modified constraint system. The application of $Unif$, $Reduce$ 1 and $Reduce$ 2 rules on $\mathcal{C}'$ outputs a modified constraint system.*

PROOF.

Let $\mathcal{C}' = (E_1 \rhd t_1, \ldots, E_n \rhd t_n)$ be a modified constraint system. Definition 22 (Chapter 2) implies that $E_i \subseteq E_{i+1}$ and $Var(E_i) \subseteq Var(\{t_1, \ldots, t_{i-1}\})$ for $i \in \{1, \ldots, n\}$. Assume $\mathcal{C}' = (E_1 \rhd x_1, \ldots, E_{i-1} \rhd x_{i-1}, E_i \rhd t_i, E_{i+1} \rhd t_{i+1}, \ldots, E_n \rhd t_n)$ with $t_i \notin \mathcal{X}$, and let us prove that the application of $Unif$, $Reduce$ 1 and $Reduce$ 2 rules on $\mathcal{C}'$ outputs a modified constraint system.

Unif. The application of *Unif* on $\mathcal{C}'$ outputs $\mathcal{C}" = (E_1\sigma \rhd x_1\sigma, \ldots, E_{i-1}\sigma \rhd x_{i-1}\sigma, E_{i+1}\sigma \rhd t_{i+1}\sigma, \ldots, E_n\sigma \rhd t_n\sigma)$ with $\sigma = mgu(u \stackrel{?}{=}_\emptyset t_i)$, $u \in E_i \setminus \mathcal{X}$. $E_j \subseteq E_{j+1}$ implies $E_j\sigma \subseteq E_{j+1}\sigma$. We prove next that $Var(E_j\sigma) \subseteq Var(x_1\sigma, \ldots, x_{i-1}\sigma, t_{i+1}\sigma, \ldots, t_{j-1}\sigma)$. Actually, we have $Var(E_j) \subseteq Var(x_1, \ldots, x_{i-1}, t_i, \ldots, t_{j-1})$. This implies that $Var(E_j\sigma) \subseteq Var(x_1\sigma, \ldots, x_{i-1}\sigma, t_i\sigma, t_{i+1}\sigma, \ldots, t_{j-1}\sigma)$. We have that $t_i\sigma = u\sigma$ and $u \in E_i$, thus $Var(t_i\sigma) = Var(u\sigma) \subseteq Var(E_i\sigma) \subseteq Var(\{x_1\sigma, \ldots, x_{i-1}\sigma\})$. This implies that $Var(E_j\sigma) \subseteq Var(x_1\sigma, \ldots, x_{i-1}\sigma, t_{i+1}\sigma, \ldots, t_{j-1}\sigma)$, and hence, the application of $Unif$ rule on $\mathcal{C}'$ outputs a modified constraint system.

Reduce 1. The application of *Reduce 1* on $\mathcal{C}'$ outputs $\mathcal{C}" = (E_1\sigma \rhd x_1\sigma, \ldots, E_{i-1}\sigma \rhd x_{i-1}\sigma, (E_i\sigma \rhd y\sigma)_{y \in l_x}, E_{i+1}\sigma \rhd t_{i+1}\sigma, \ldots, E_n\sigma \rhd t_n\sigma)$ with $\sigma = mgu(r \stackrel{?}{=}_\emptyset t_i, (u_k \stackrel{?}{=}_\emptyset l_k)_{1 \leq k \leq m})$, $u_k \in E_i \setminus \mathcal{X}$, and $l_x, l_1, \ldots, l_m \rightarrow r \in \mathcal{L}_{\mathcal{I}_2 inc}$. $E_j \subseteq E_{j+1}$ implies $E_j\sigma \subseteq E_{j+1}\sigma$. Let $j \leq i$, we have that $Var(E_j) \subseteq Var(\{x_1, \ldots, x_{j-1}\})$ and hence, $Var(E_j\sigma) \subseteq Var(\{x_1\sigma, \ldots, x_{j-1}\sigma\})$. Let $j > i$, and let us prove that $Var(E_j\sigma) \subseteq Var(\{x_1\sigma, \ldots, x_{i-1}\sigma\} \cup \{y\sigma\}_{y \in l_x} \cup \{t_{i+1}\sigma, \ldots, t_{j-1}\sigma\})$. We have that $Var(E_j\sigma) \subseteq Var(\{x_1\sigma, \ldots, x_{i-1}\sigma, t_i\sigma, t_{i+1}\sigma, \ldots, t_{j-1}\sigma\})$, and $Var(t_i\sigma) = Var(r\sigma) \subseteq Var(l_x\sigma) \cup Var(\{l_1\sigma, \ldots, l_m\sigma\}) \subseteq Var(l_x\sigma) \cup Var(E_i\sigma) \subseteq Var(l_x\sigma) \cup Var(\{x_1\sigma, \ldots, x_{i-1}\sigma\})$. We conclude that $Var(E_j\sigma) \subseteq Var(\{x_1\sigma, \ldots, x_{i-1}\sigma\} \cup \{y\sigma\}_{y \in l_x} \cup \{t_{i+1}\sigma, \ldots, t_{j-1}\sigma\})$, and hence, the application of $Reduce$ 1 rule on $\mathcal{C}'$ outputs a modified constraint system.

Reduce 2. The application of *Reduce 2* on $\mathcal{C}'$ outputs $\mathcal{C}" = (E_1\sigma \rhd x_1\sigma, \ldots, E_{i-1}\sigma \rhd x_{i-1}\sigma, (E_i\sigma \rhd y\sigma)_{y \in l_x}, E_i\sigma \cup r\sigma \rhd t_i\sigma, E_{i+1}\sigma \cup r\sigma \rhd t_{i+1}\sigma, \ldots, E_n\sigma \cup r\sigma \rhd t_n\sigma)$ with $\sigma = mgu((u_k \stackrel{?}{=}_\emptyset l_k)_{1 \leq k \leq m})$, $u_k \in E_i \setminus \mathcal{X}$,

and $l_x, l_1, \ldots, l_m \rightarrow r \in \mathcal{L}_{\mathcal{I}_2 dec}$. $E_j \subseteq E_{j+1}$ implies $E_j\sigma \subseteq E_{j+1}\sigma$. Let $j \leq i$, we have that $Var(E_j) \subseteq Var(\{x_1, \ldots, x_{j-1}\})$ and hence, $Var(E_j\sigma) \subseteq Var(\{x_1\sigma, \ldots, x_{j-1}\sigma\})$. By definition of decreasing rule (Definition 49), we have that $l_i \succeq r$ for $1 \leq i \leq m$ or $x \succeq r$ with $x \in l_x$.

- If $l_i \succeq r$, then $l_i\sigma \succeq r\sigma$, and thus $Var(r\sigma) \subseteq Var(l_i\sigma) = Var(u_i\sigma)$. We conclude that $Var(r\sigma) \subseteq Var(E_i\sigma)$.
- If $x \succeq r$ for $x \in l_x$ then $Var(r\sigma) \subseteq Var(x\sigma)$.

If both cases, we deduce that $Var(E_j\sigma \cup r\sigma) \subseteq Var(\{x_1\sigma, \ldots, x_{i-1}\sigma\}) \cup \bigcup_{y \in l_x} Var(y\sigma) \cup Var(\{t_i\sigma, \ldots, t_{j-1}\sigma\})$ for $j \geq i$. We conclude that the application of $Reduce\ 2$ rule on $\mathcal{C}'$ outputs a modified constraint system.

■

We prove below that the correctness and completeness of the rules in Fig. 5.3 with respect to the satisfiability of modified constraint systems.

Lemma 46 *A satisfiable modified constraint system not in solved form can be reduced into another satisfiable modified constraint system by applying a rule of figure 5.3.*

PROOF.

Let $\mathcal{C} = (E_j \rhd t_j)_{1 \leq j \leq n}$ be a satisfiable modified constraint system not in solved form and let i be the smallest integer such that $t_i \notin \mathcal{X}$. Let $\mathcal{C} = (\mathcal{C}_\alpha, E_i \rhd t_i, \mathcal{C}_\beta)$ where $\mathcal{C}_\alpha$ is in solved form. Since $\mathcal{C}$ is satisfiable there exists a substitution σ such that $\sigma \models_{\mathcal{I}_2} \mathcal{C}$. Let us prove that $\mathcal{C}$ can be reduced into another satisfiable modified constraint system $\mathcal{C}'$ by applying transformation rules given in figure 5.3. By lemma 9 (Chapter 2), $\sigma \models_{\mathcal{I}_2} \mathcal{C}$ implies $\sigma \models_{\mathcal{I}_2} (\mathcal{C}_\alpha, E_i \setminus \mathcal{X} \rhd t_i, \mathcal{C}_\beta)$ and that, since $\mathcal{I}_2$ is $\mathcal{I}_1$-strongly order local, there is a well-formed derivation D starting from $(E_i \setminus \mathcal{X})\sigma$ of goal $t_i\sigma$. We have two cases:

- If $t_i\sigma \in (E_i \setminus \mathcal{X})\sigma$ then there exists a term $u \in E_i \setminus \mathcal{X}$ such that $u\sigma = t_i\sigma$. Let $\mu = mgu(t_i, u)$, we have $\sigma = \mu\theta$ for some substitution θ. $\mathcal{C}$ can then be reduced to $\mathcal{C}'$ by applying *Unif* rule, $\mathcal{C}' = (\mathcal{C}_\alpha\mu, \mathcal{C}_\beta\mu)$ and $\theta \models_{\mathcal{I}_2} \mathcal{C}'$.
- If $t_i\sigma \notin (E_i \setminus \mathcal{X})\sigma$, let $D : (E_i \setminus \mathcal{X})\sigma \rightarrow \ldots \rightarrow F\sigma, t_i\sigma$ and for every step in D where $\widetilde{l} \rightarrow r$ is the rule applied with the substitution γ, for every $s \in \widetilde{l} \setminus \mathcal{X}$, we have either $s\gamma \in (E_i \setminus \mathcal{X})\sigma$ or $s\gamma$ was constructed by a former decreasing rule.
  - Suppose that all applied rules in D are increasing and let $\widetilde{l} \rightarrow r$ be the last applied rule with the substitution γ, this implies that $r\gamma = t_i\sigma$ and for every $s \in \widetilde{l} \setminus \mathcal{X}$, $s\gamma \in (E_i \setminus \mathcal{X})\sigma$ and then for every $s \in \widetilde{l} \setminus \mathcal{X}$ there exists a term $u \in E_i \setminus \mathcal{X}$ such that $s\gamma = u\sigma$. Let μ be the most general unifier of $\left\{ r \stackrel{?}{=}_\emptyset t_i, (s \stackrel{?}{=}_\emptyset u)_{\forall s \in \widetilde{l} \setminus \mathcal{X}, u \in E_i \setminus \mathcal{X} \text{ and } s\gamma = u\sigma} \right\}$, we have $\sigma = \mu\theta$ and

$\gamma = \mu\theta$ for some θ. This implies that $\mathcal{C}$ can be reduced to $\mathcal{C}' = (\mathcal{C}_\alpha, (E_i \rhd x)_{x\in\widetilde{l}}, \mathcal{C}_\beta)\mu$ by applying *Reduce 1* and $\theta \models_{\mathcal{I}_2} \mathcal{C}'$.

– Suppose that D contains decreasing rules and let j be the first step where the applied rule is decreasing. Let $\widetilde{l} \to r$ be this rule applied with substitution γ. $D : (E_i \setminus \mathcal{X})\sigma = F_0\sigma \to F_0\sigma, t_1\sigma \to \ldots \to F_{j-1}\sigma \to F_{j-1}\sigma, t_j\sigma \to \ldots \to F_{n-1}\sigma, t_i\sigma$. Since D is well-formed, we deduce that for every $s \in \widetilde{l} \setminus \mathcal{X}$, $s\gamma \in (E_i \setminus \mathcal{X})\sigma$ and then, for every $s \in l \setminus \mathcal{X}$ there exists a term $u \in E_i \setminus \mathcal{X}$ such that $s\gamma = u\sigma$. Let μ be the most general unifier, we have $\gamma = \mu\theta$ and $\gamma = \mu\theta$ for some substitution θ. This implies that $\mathcal{C}$ can be reduced to $\mathcal{C}' = (\mathcal{C}_\alpha, (E_i \rhd x)_{x\in\widetilde{l}}, (E_i \cup r) \rhd t_i, \mathcal{C}'_\beta)\mu$ by applying *Reduce 2* and $\theta \models_{\mathcal{I}_2} \mathcal{C}'$.

■

Lemma 47 *Let $\mathcal{C}$ and $\mathcal{C}'$ be two modified constraint systems such that $\mathcal{C}'$ is obtained from $\mathcal{C}$ by applying a transformation rule. If $\mathcal{C}'$ is satisfiable then so is $\mathcal{C}$.*

PROOF.

Let $\mathcal{C}$ and $\mathcal{C}'$ be two modified constraint systems such that $\mathcal{C}'$ is obtained from $\mathcal{C}$ by applying a transformation rule and suppose that $\mathcal{C}'$ is satisfiable. Let σ' be a solution of $\mathcal{C}'$ and let us prove that $\mathcal{C}$ is satisfiable. Since a transformation rule can be applied on $\mathcal{C}$, $\mathcal{C}$ can't be in solved form. Suppose that $\mathcal{C} = (\mathcal{C}_\alpha, E \rhd t, \mathcal{C}_\beta)$ where $\mathcal{C}_\alpha$ is in solved form and $t \notin \mathcal{X}$.

- If $\mathcal{C}'$ is obtained from $\mathcal{C}$ by applying *Unif* rule then, there exists a term $u \in E \setminus \mathcal{X}$ such that u and t are unifiable. Let μ be the most general unifier then $\mathcal{C}' = (\mathcal{C}_\alpha\mu, \mathcal{C}_\beta\mu)$. Since $\sigma' \models_{\mathcal{I}_2} \mathcal{C}'$, we have $\sigma'\mu \models_{\mathcal{I}_2} (\mathcal{C}_\alpha, \mathcal{C}_\beta)$ and by the fact that μ is the most general unifier of t and a term in E we have $\sigma'\mu \models_{\mathcal{I}_2} E \rhd t$. We deduce that $\sigma'\mu \models_{\mathcal{I}_2} \mathcal{C}$.

- If $\mathcal{C}'$ is obtained from $\mathcal{C}$ by applying *Reduce 1* then there exists an increasing rule $l_x, l_1, \ldots, l_n \to r$, a set of terms $e_1, \ldots, e_n$ in $E \setminus \mathcal{X}$ such that $\left\{r \stackrel{?}{=}_\emptyset t, (l_i \stackrel{?}{=}_\emptyset e_i)_{1\le i\le n}\right\}$ has solution. Let μ be the most general unifier. $\mathcal{C}' = (\mathcal{C}_\alpha, (E \rhd x)_{x\in l_x}, \mathcal{C}_\beta)\mu$. Since $\sigma' \models_{\mathcal{I}_2} \mathcal{C}'$ and by definition of μ, we have $\sigma'\mu \models_{\mathcal{I}_2} \mathcal{C}$.

- If $\mathcal{C}'$ is obtained from $\mathcal{C}$ by applying *Reduce 2* then there exists a decreasing rule $l_x, l_1, \ldots, l_n \to r$ and a set of terms $e_1, \ldots, e_n$ in $E \setminus \mathcal{X}$ such that $\left\{(l_i \stackrel{?}{=}_\emptyset e_i)_{1\le i\le n}\right\}$ has solution. Let μ be the most general unifier. $\mathcal{C}' = (\mathcal{C}_\alpha, (E \rhd x)_{x\in l_x}, (E \cup r) \rhd t, \mathcal{C}'_\beta)\mu$. Since $\sigma' \models_{\mathcal{I}_2} \mathcal{C}'$ and by definition of μ and modified constraint systems, we have $\sigma'\mu \models_{\mathcal{I}_2} \mathcal{C}$.

■

5.5 Decidability results

5.5.1 Decidability of the ground reachability problems

We recall that $\mathcal{I}_1 = \langle \mathcal{F}, \mathcal{T}_{\mathcal{I}_1}, \mathcal{H} \rangle$ is the initial deduction system, $\mathcal{L}_1 = \mathcal{L}_{\mathcal{I}_1}$ is the associated intruder deduction rules, and $\mathcal{I}_2 = \langle \mathcal{F}, \mathcal{L}_2, \emptyset \rangle$ is a deduction system which is $\mathcal{I}_1$-strongly order local.

A core result of this chapter is the following lemma.

Lemma 48 *Let $\mathcal{I}_2$ be a deduction system such that $\mathcal{I}_2$ is $\mathcal{I}_1$-strongly order local, and $\mathcal{L}_2$ is finite. Applying the transformation algorithm of Fig. 5.3 on a modified constraint system $\mathcal{C}$ without instantiating the variables of $\mathcal{C}$ yields only a finite number of different modified constraint systems.*

PROOF.

Assume the application of rules of Fig. 5.3 yields an infinite sequence of modified constraint systems $\mathcal{C}_1, \ldots, \mathcal{C}_n, \ldots$. Let us prove there is only a finite number of different $\mathcal{C}_i$ when identical constraints within a modified constraint system are identified.

Let us first prove that there is only a finite number of different left-hand side of deduction constraints. The number of different left-hand sides in a modified constraint system does not change (or decrease) when a UNIF or REDUCE1 rule is applied. Assume now that a decreasing rule $l_x, l_1, \ldots, l_n \rightarrow r \in \mathcal{L}_2$ is applied with a substitution σ on a constraint with left-hand side E. If $r\sigma \in E$, the number of different left-hand side does not change. Thus let us assume $r\sigma \notin E$, and thus $r\sigma \notin \{l_1\sigma, \ldots, l_n\sigma\}$. Since r is smaller or equal to a term of the left-hand side of there rule, we have two case:

- Either there exists i with $l_i\sigma \succ r\sigma$, and thus there exists $e \in E$ such that $e \succ r\sigma$.
- Or $r \in l_x \setminus Var(l_1, \ldots, l_n)$. Then the obtained modified constraint system contains the deduction constraints $E \rhd r$ and $E \cup \{r\} \rhd t$ and not other constraint contains r. By Lemma 9 the obtained modified constraint system is equivalent to the one in which $E \cup \{r\} \rhd t$ is replaced by $E \rhd t$.

Let us now consider the set T which is the union of all left-hand side of deduction constraints reachable from E by employing a decreasing rule.

- the root is labelled by $\emptyset$;
- the sons of the root are labelled by the terms in a left-hand side E;
- The sons of the non-root node are defined as follows: assume there exists two left-hand sides E' and E'' where E' is reachable from E, and there is a decreasing rule whose application leads to the addition of a deduction

constraint with left-hand side $E'' = E'', t_1$. Let $t_2 \in E'$ be the term strictly greater than t_1. We then set t_1 as a son of t_2.

Since $t_2 \succ t_1$ there is no cycle, and since we consider sets reachable from E, the "is son of" relation is connected. It thus defines a tree. We note that t_2 is the instance of a non-variable term l in the left-hand side of a decreasing rule. There is only a finite number of such terms. Since we consider deductions in the empty theory, for each l there is a unique substitution σ such that $l\sigma = t_2$. Given the properties of ordering we have $Var(r) \subseteq Var(l)$ and thus $t_1 = r\sigma$ is uniquely determined by the rule applied. Thus, each term t_2 has a finite number of sons t_1. Along each branch of the tree a node t is strictly smaller than its parent. Since $\succ$ is a well-founded ordering, this implies that each branch is finite. Thus, by König's Lemma, this tree is finite. We conclude that T itself is finite. Each left-hand side of a deduction constraint is a subset of T, thus there is only a finite number of different left-hand sides.

When applying REDUCE 1 or REDUCE 2 on a constraint $E \rhd t$, the newly introduced constraints $E \rhd t'$ are such that t' is a strict subterm of a term in E or t. Let $E' \rhd t'$ be a deduction constraint reached from $E \rhd t$. Either t' is a subterm of t or there exists E'' reachable from E such that t' is a strict subterm of E''. Since there is only a finite number of different E'', there is thus only a finite number of possible right-hand side of constraints.

In conclusion only a finite number of deduction constraints $E' \rhd t'$ can be reached from a deduction constraint $E \rhd t$. Thus only a finite number of modified constraint systems can be reached from a given one by applying rules that do not instantiate the variables in the modified constraint system. ■

Let E and t be respectively a set of ground terms in normal form and a ground term in normal form. We recall that $t \in \overline{E}^{\mathcal{I}_1}$ if and only if $t \in \overline{E}^{\mathcal{I}_2}$ ($\mathcal{I}_2$ is $\mathcal{I}_1$-strongly order local). This implies that solving $\mathcal{I}_1$-ground reachability problem is reduced to solving $\mathcal{I}_2$-ground reachability problem. It is then routine to see that a ground constraint system is satisfiable if, and only if, it reduces to an empty sequence of deduction constraints. Thus by Lemma 48 we have:

Theorem 10 *If $\mathcal{L}_2$ is finite, the $\mathcal{I}_1$-ground reachability problem is decidable.*

We recall that $\mathcal{I}_0 = \langle \mathcal{F}, \mathcal{T}_{\mathcal{I}_0}, \mathcal{H} \rangle$ and that $\mathcal{I}' = \langle \mathcal{F}, \mathcal{L}_{\mathcal{I}'}, \emptyset \rangle$ is the output of the saturation algorithm given in Figure 5.1. Since $\mathcal{I}'$ is $\mathcal{I}_0$-strongly order local (Corollary 4), and by Lemmas 46, 47, and 48, we conclude that the following theorem.

Theorem 11 *If the saturation algorithm terminates on $\mathcal{L}_0$, the $\mathcal{I}_0$-ground reachability problem is decidable.*

While the termination of the saturation on $\mathcal{L}_0$ implies the decidability of its ground reachability problem, we prove next that this condition is not sufficient to decide $\mathcal{I}_0$ general reachability problem.

5.5.2 Termination of Saturation does not imply decidability of general reachability problems

It is well-known how to encode 2-stack automata into deduction systems. However the saturation will typically not terminate on standard encodings as it will amount in this case to the pre-computation of all possible executions of the automaton. We can however adapt the construction so that saturation terminates. We consider a signature $\mathcal{F}$ such that, for all symbol $f \in \mathcal{F}_0$ of arity n, there is a deduction rule $x_1, \ldots, x_n \to f(x_1, \ldots, x_n)$, and the signature $\mathcal{F} = \mathcal{F} \cup \{g\}$ with g a symbol of arity 1. Let $(Q, Q_I, Q_F, \Sigma, \Pi, \Delta)$ be a finite 2-stack automaton, where Q is the finite set of states of the automaton, Q_I and Q_F its initial and final states, Σ denotes the alphabet of the words read by the automaton, and Π denotes the elements in the stacks of the automaton. We shall encode the emptiness of the language recognised by this automaton into a general reachability problem. Let us assume there exists:

- $\bot \in \mathcal{F}_0$ be a constant denoting the empty stack or the empty word;
- one unary symbol u_α for each letter $\alpha \in \Sigma \cup \Pi$;
- one constant $q \in \mathcal{F}$ for each state in Q;
- one symbol $s \in \mathcal{F}$ of arity 4 where we intend that:
  - the first argument represents the word that remains to be read by the automaton;
  - the second argument represents the current state of the automaton;
  - the third and fourth arguments represent the two stacks of the automaton.
- one symbol f of arity 2.

We represent a transition from a state σ_1 to a state σ_2 with a symbol τ of arity 1 unique to the transition and a rewriting rule $\tau(g(f(\sigma_1, f(\sigma_2, x)))) \to g(f(\sigma_2, x))$. Given the unicity of τ, the rewriting system has no critical pairs, and thus is convergent. Since every narrowing step decreases strictly the number of "τ" symbols in a term, narrowing terminates, and thus the equational theory has the finite variant property. At the end of the first step of the saturation the system will contain the rules enabling the attacker to build sequences of states, and additional rules $g(f(\sigma_1, f(\sigma_2, x))) \to g(f(\sigma_2, x))$ that are decreasing for any recursive path ordering. Since there is no increasing rule with the symbol g in the right-hand side, the saturation terminates, and hence that ground reachability problems are decidable.

However, the instance of x in the following reachability problem encodes a word recognised by the automaton after a run encoded by the instance of y:

$$\emptyset \rhd f(s(x, q_0, \bot, \bot), y), g(f(s(x, q_0, \bot, \bot), y)) \rhd g(s(\bot, q_f, \bot, \bot))$$

This example proves (with $q_0 \in Q_I$ and $q_f \in Q_F$) that the saturation can terminate and yield a deduction system for which general reachability problems are not decidable.

The undecidability comes from the fact that one can apply an unbounded number of decreasing rules on a non-ground terms, and from the "lack of regularity" on the terms obtained.

5.5.3 Decidability of the general reachability problems

We recall that the initial intruder system is given by $\mathcal{I}_0 = \langle \mathcal{F}, \mathcal{T}_0, \mathcal{H} \rangle$ while $\mathcal{H}$ is generated by a convergent equational theory and has the finite variant property, $\mathcal{L}_0 = \mathcal{L}_{\mathcal{I}_0}$ is the intruder deduction rules associated to $\mathcal{I}_0$. We recall also that $\mathcal{I}' = \langle \mathcal{F}, \mathcal{L}', \emptyset \rangle$ is the saturated intruder system.

We give here a simple criterion that permits to ensure the termination of the resolution of a constraint problem with a saturated deduction system. Let T be a set of terms, $T = \{t_1, \ldots, t_m\}$, we let $\Delta(T)$ to be the set of strict maximal subterms of T and we define:

$$\delta(T) = \begin{cases} +\infty & \text{if } T \subseteq \mathcal{X} \\ |T \setminus \mathcal{X}| - |Var(T \setminus \mathcal{X}) \setminus (T \cap \mathcal{X})| & \text{otherwise.} \end{cases}$$

Now let us define $\mu(T)$. We consider the image of the set of terms T by the rewriting system $\mathcal{U}$ containing rules $f(x_1, \ldots, x_n) \to x_1, \ldots, x_n$ for every symbol f in the signature of the deduction system. We define:

$$\mu(T) = \min_{\substack{T\sigma \to_{\mathcal{U}}^* T' \\ \sigma \text{ mgu of subterms of } T}} \delta(T')$$

We extend μ to rules as follows. Let $\mathcal{L}'$ be the set of deduction rules. We recall that $\mathcal{L}'$ is partitioned into two disjoint sets of deduction rules, the set of increasing rules $\mathcal{L}'_{inc}$ and the set of decreasing rules $\mathcal{L}'_{dec}$. For every rule $\widetilde{l} \to r \in \mathcal{L}'$,

$$\mu(\widetilde{l} \to r) = \begin{cases} \mu(\Delta(\widetilde{l} \setminus \mathcal{X} \cup \{r\})) \text{ if } \widetilde{l} \to r \text{ is increasing,} \\ \mu(\Delta(\widetilde{l} \setminus \mathcal{X})) \text{ otherwise.} \end{cases}$$

Definition 53 *(Contracting deduction systems) A saturated deduction system $\mathcal{I}' = \langle \mathcal{F}, \mathcal{L}', \emptyset \rangle$ is* contracting *if for all rules $\widetilde{l} \to r$ in $\mathcal{L}'$ we have $\mu(\widetilde{l} \to r) > 0$.*

Figure 5.4 Martelli-Montanari $\emptyset$-unification algorithm

Given an $\emptyset$-unification system $\mathcal{U}$, repeatedly perform any of the following transformations. If no transformation applies, stop with success.

- Select any equation of the form $t \stackrel{?}{=}_\emptyset x$ where t is not a variable and x is a variable, and rewrite it as $x \stackrel{?}{=}_\emptyset t$.
- Select any equation of the form $x \stackrel{?}{=}_\emptyset x$ where x is variable, and erase it.
- Select any equation of the form $t' \stackrel{?}{=}_\emptyset t"$ where t' and $t"$ are not variables. If the two root function symbols are different, stop with failure; otherwise, assume $t' = f(t_1, \ldots, t_n)$ and $t" = f(s_1, \ldots, s_n)$ with f a function symbol with arity n and apply the following:

  If $n = 0$, then f is a constant symbol, and the equation is simply erased;
  otherwise, replace $f(t_1, \ldots, t_n) \stackrel{?}{=}_\emptyset f(s_1, \ldots, s_n)$ with the following equations:
  $$t_1 \stackrel{?}{=}_\emptyset s_1, \ldots, t_n \stackrel{?}{=}_\emptyset s_n.$$

- Select any equation of the form $x \stackrel{?}{=}_\emptyset t$ where x is a variable which occurs somewhere else in the unification system and where $t \neq x$. If x occurs in t, then stop with failure; otherwise, apply the substitution $\sigma = \{x \mapsto t\}$ to both terms of all other equations in the unification system (without erasing $x \stackrel{?}{=}_\emptyset t$).

Martelli-Montanari unification algorithm. In this paragraph, we recall the unification algorithm due to Martelli-Montanari [147], which is used in the proof of the next lemma. In [147], A. Martelli and U. Montanari gave a $\emptyset$-unification algorithm based on the transformation of a given $\emptyset$-unification system $\mathcal{U}$ into an equivalent and simpler unification system. An unification system $\mathcal{U}$ is said to be in *solved form* if and only if it satisfies the following conditions:

- every equation in $\mathcal{U}$ is of the form $x \stackrel{?}{=}_\emptyset t$;
- every variable which is the left member of some equation occurs only there.

An $\emptyset$-unification system $\mathcal{U} = \left\{x_1 \stackrel{?}{=}_\emptyset t_1, \ldots, x_n \stackrel{?}{=}_\emptyset t_n\right\}$ in solved form has the obvious unifier $\sigma = \{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$, which is its most general $\emptyset$-unifier.

We give in Figure 5.4 Martelli-Montanari $\emptyset$-unification algorithm. In [147], A. Martelli and U. Montanari proved that for any $\emptyset$-unification system $\mathcal{U}$

- their algorithm always terminates, no matter which choices are made,

- if the algorithm terminates with failure, $\mathcal{U}$ has no unifier, and if the algorithm terminates with success, $\mathcal{U}$ has been transformed into an equivalent $\emptyset$-unification system in solved form.

We remark that Martelli-Montanari $\emptyset$-unification algorithm provides a widely general version from which most unification algorithms [35, 55, 148, 167, 177, 176, 197, 198] can be derived.

Lemma 49 *Let $S = \{s_1, \ldots, s_n\}$ and $T = \{t_1, \ldots, t_n\}$ be two sets of terms and let σ be the most general unifier of $V = \left\{s_1 \stackrel{?}{=}_\emptyset t_1, \ldots, s_n \stackrel{?}{=}_\emptyset t_n\right\}$. If $\mu(T) > 0$ then either* $\text{nbv}(s_1, \ldots, s_n) > \text{nbv}((s_1, \ldots, s_n, t_1, \ldots, t_n)\sigma)$ *or* $\text{nbv}(s_1, \ldots, s_n) = \text{nbv}((s_1, \ldots, s_n, t_1, \ldots, t_n)\sigma)$, *$S = S\sigma$ and for all $x \in Var(T)$ there is $i \in \{1, \ldots, n\}$ such that $\sigma(x) \preceq s_i$.*

PROOF.
Let $V = \left\{s_1 \stackrel{?}{=}_\emptyset t_1, \ldots, s_n \stackrel{?}{=}_\emptyset t_n\right\}$. In order to solve V, we apply the first step of the unification algorithm of Martelli-Montanari [147] (Figure 5.4). We reduce V to $V' = \left\{x_1 \stackrel{?}{=}_\emptyset u_1, \ldots, x_k \stackrel{?}{=}_\emptyset u_k, x_{k+1} \stackrel{?}{=}_\emptyset u_{k+1}, \ldots, x_m \stackrel{?}{=}_\emptyset u_m\right\}$ such that for every equation $x \stackrel{?}{=}_\emptyset u \in V'$, we have either $x \in Var(S)$ and $u \in Sub(T)$ or $x \in Var(T)$ and $u \in Sub(S)$. We suppose that $x_j \in Var(T)$ for $j \in \{1, \ldots, k\}$.

- If $k = m$ then we have $x_j \in Var(T)$ for $j \in \{1, \ldots, m\}$. We suppose that $x_i \neq x_j$ for all $i, j \in \{1, \ldots, m\}$ and $i \neq j$. This implies that $S\sigma = S$ and $Var(T)$ are instantiated by subterms of S, that is $Var(T)\sigma$ are smaller or equal than terms in S. We conclude also that $\text{nbv}(s_1, \ldots, s_n) = \text{nbv}((s_1, \ldots, s_n, t_1, \ldots, t_n)\sigma)$.

- If $k \neq m$ assume $\{u_{k+1}, \ldots, u_m\} \notin Var(T)$, we have different cases:

  - If for all different $i, j \in \{1, \ldots, m\}$ we have $x_i \neq x_j$ then $m-k$ variables of S, $x_{k+1}, \ldots, x_m$, are instantiated by subterms of T, $u_{k+1}, \ldots, u_m$. This implies that when we apply σ to $\mathcal{U}$, new variables, $Var(u_{k+1}, \ldots, u_m) \setminus \{x_1, \ldots, x_k\}$ will appear in $S\sigma$. There exists a set $T' \not\subseteq \mathcal{X}$ such that $T \rightarrow_{\mathcal{U}}^* T'$ and $T' = \{x_1, \ldots, x_k, u_{k+1}, \ldots, u_m\}$. Since $\mu(T) > 0$, we have $|T' \setminus \mathcal{X}| > |Var(T' \setminus \mathcal{X}) \setminus (x_1, \ldots, x_k)|$. This implies that $nbv(s_1, \ldots, s_n) > nbv((s_1, \ldots, s_n, t_1, \ldots, t_n)\sigma)$.
  - If there is different $i, j \in \{1, \ldots, m\}$ such that $x_i = x_j$:
    - If $i, j \leq k$ then we have to unify two subterms of S. Let u_i and u_j be these two subterms and α be their most general unifier.
      Let us apply α on V and to solve V we have to solve $V\alpha = \left\{s_1\alpha \stackrel{?}{=}_\emptyset t_1, \ldots, s_n\alpha \stackrel{?}{=}_\emptyset t_n\right\}$. To solve $V\alpha$ we reduce it to another

system V" where equations have the same form as in V'. We note that nbv(T) in $V\alpha$ is the same as in V and nbv(S) is reduced.
By the same reasoning as above, we deduce that nbv$(S) >$ nbv$(S\sigma, T\sigma)$.

* If $i, j > k$ then we have to unify two subterms of T. Let u_i and u_j be these two subterms and α be their most general unifier. Let us apply α on V and to solve V we have to solve $V\alpha = \left\{s_1 \stackrel{?}{=}_\emptyset t_1\alpha, \ldots, s_n \stackrel{?}{=}_\emptyset t_n\alpha\right\}$ and to solve $V\alpha$, we have to reduce it to another system V" where equations have the same form as in V'. $V" = \left\{x_1 \stackrel{?}{=}_\emptyset u_1, \ldots, x_m \stackrel{?}{=}_\emptyset u_m\right\}$ where $x_1 \ldots, x_k \in Var(T\alpha)$ and $x_{k+1}, \ldots, x_m \in Var(S)$. By definition of μ and by following the same reasoning as above, we deduce that:
  - If $k = m$ and for all different $i, j \in \{1, \ldots, m\}$ we have $x_i \neq x_j$, we deduce that $S = S\sigma$, $Var(T)\sigma$ are smaller or equals than terms in S and then nbv$(S) =$ nbv$(S\sigma, T\sigma)$.
  - If $k = m$ and there is different $i,\ j$ such that $x_i = x_j$ then we have to unify two subterms of S and then we conclude that nbv$(S) >$ nbv$(S\sigma, T\sigma)$.
  - If $k \neq m$ we deduce that nbv$(S) >$ nbv$(S\sigma, T\sigma)$.

■

It is easy to see that the following corollary immediately follows from Martelli-Montanari $\emptyset$-unification algorithm [147] (Figure 5.4) and from Lemma 49.

Corollary 5 *For any arbitrary terms s and t, if $\sigma = mgu_\emptyset(s, t)$ then*

- nbv$(s\sigma) \leq$ nbv(s, t),
- *and if* nbv$(s\sigma) =$ nbv(s, t) *then* $s = t$.

The definition of μ is tailored to the proof of the following Lemma.

Remark. Let T be a set of terms and let $\Sigma(T)$ be the set of substitutions σ such that σ is the most general unifier of some subterms of T. We remark that $\mu(T)$ is defined with respect to $T\sigma$ for every $\sigma \in \Sigma$. It will be more natural if $\mu(T)$ is defined with respect to T instead of some instances of T. The simpler definition would be as follow:

$$\mu(T) = \min_{T \rightarrow_{\mathcal{U}}^* T'} \delta(T')$$

Using this simpler definition of μ, we remark that $\mu(T) > 0$ does not imply $\mu(T\sigma) > 0$ for a set of terms T and a substitution $\sigma \in \Sigma(T)$. Let $T =$

$\{f(x,x), f(x,y), f(y,x)\}$ and let σ be such that $\sigma(x) = y$. Using the general definition of μ, we remark that $\mu(T) > 0$ and $\mu(T\sigma) = 0$.

Lemma 50 *Let $\mathcal{I}'$ be a saturated contracting deduction system, $\mathcal{C}$ be a modified $\mathcal{I}'$-constraint system not in solved form. If a transformation is applied on $\mathcal{C}$ to yield a modified constraint system $\mathcal{C}'$, then either the substitution applied does not instantiate the variables of $\mathcal{C}$ and $Var(\mathcal{C}') \subseteq Var(\mathcal{C})$ or* nbv$(\mathcal{C}') <$ nbv$(\mathcal{C})$.

PROOF.
Let $\mathcal{C}$ be a modified constraint system such that a transformation rule can be applied on it. This implies that $\mathcal{C}$ is not in solved form. Let $\mathcal{C} = (\mathcal{C}_\alpha, E \rhd t, \mathcal{C}_\beta)$ such that $\mathcal{C}_\alpha$ is in solved form and $t \notin \mathcal{X}$. We have three cases:

- If we apply *Unif* rule on $\mathcal{C}$ then there exists a term $e \in E \setminus \mathcal{X}$ such that e and t are unifiable and σ is the most general unifier. $\mathcal{C}$ is then reduced to $\mathcal{C}' = (\mathcal{C}_\alpha, \mathcal{C}_\beta)\sigma$. Since we unify two subterms of $\mathcal{C}$ in the empty theory, either σ does not instantiate the variables of $\mathcal{C}$ and then $\mathcal{C}' = (\mathcal{C}_\alpha, \mathcal{C}_\beta)$ (which implies that $Var(\mathcal{C}') \subseteq Var(\mathcal{C})$) or σ instantiates the variables of $\mathcal{C}$ (and then nbv$(\mathcal{C}') <$ nbv$(\mathcal{C})$) (Corollary 5).

- Assume we apply REDUCE 1 on $\mathcal{C}$. By definition of REDUCE 1 there exists an increasing rule $l_x, l_1, \ldots, l_n \to r \in \mathcal{L}'$, a set of terms $e_1, \ldots, e_n \in E \setminus \mathcal{X}$ such that $\mathcal{U} = \left\{ r \stackrel{?}{=}_\emptyset t, (e_i \stackrel{?}{=}_\emptyset l_i)_{1 \le i \le n} \right\}$ has a solution. Let σ be its most general unifier. Either $\sigma_{|Var(\mathcal{C})} = Id$ or not. Let us examine the two cases.

  $\sigma_{|Var(\mathcal{C})} = Id$**.** In this case, $\mathcal{C}$ is reduced to $\mathcal{C}' = (\mathcal{C}_\alpha, (E \rhd x\sigma)_{x \in l_x}, \mathcal{C}_\beta)$. For each $l_i \in \{l_1, \ldots, l_n\}$ we have, by definition of σ, $l_i\sigma = e_i$. Also, we have $r\sigma = t$. Thus for each $x \in Var(l_1, \ldots, l_n, r)$ we have $Var(x\sigma) \subseteq Var(\mathcal{C})$. Since $l_x \subseteq Var(l_1, \ldots, l_n, r)$ we deduce that $Var(\mathcal{C}') \subseteq Var(\mathcal{C})$.

  $\sigma_{|Var(\mathcal{C})} \neq Id$**.** In this case $\mathcal{C}$ is reduced to $\mathcal{C}' = (\mathcal{C}_\alpha, (E \rhd x)_{x \in l_x}, \mathcal{C}_\beta)\sigma$. Since the e_i and r are not variables, we can decompose all equations in $\mathcal{U}$ to obtain a set of equations in which each equation has a member in $\Delta(l_1, \ldots, l_n, r)$. Since the deduction system is contracting Lemma 49 implies nbv$(e_1, \ldots, e_n, t) >$ nbv$(e_1\sigma, \ldots, e_n\sigma, t\sigma, l_1\sigma, \ldots, l_n\sigma, r\sigma)$. Since $l_x \subseteq Var(l_1, \ldots, l_n, r)$ we deduce that nbv$(\mathcal{C}) >$ nbv$(\mathcal{C}')$.

- Let us finally assume REDUCE 2 is applied. First let us prove we can assume $l_x \cup Var(r) \subseteq Var(\{l_1, \ldots, l_n\})$. Since the rule is decreasing there exists a term $l \in l_x \cup \{l_1, \ldots, l_n\}$ such that $Var(r) \subseteq Var(l)$. Thus it suffices to prove $l_x \subseteq Var(\{l_1, \ldots, l_n\})$. By definition of the REDUCE 2 rule, the

modified constraint system $\mathcal{C}$ is transformed into

$$\begin{array}{rl} & (\mathcal{C}_\alpha, (E \rhd y)_{y\in l_x\setminus\{x\}}, E \rhd x, E\cup\{x\} \rhd t, \mathcal{C}'_\beta)\sigma \\ = & \mathcal{C}_\alpha\sigma, (E\sigma \rhd y\sigma)_{y\in l_x\setminus\{x\}}, E\sigma \rhd x, E\sigma\cup\{x\} \rhd t\sigma, \mathcal{C}'_\beta\sigma \\ \equiv & \mathcal{C}_\alpha\sigma, (E\sigma \rhd y\sigma)_{y\in l_x\setminus\{x\}}, E\sigma \rhd x, E\sigma \rhd t\sigma, \mathcal{C}_\beta\sigma \\ \equiv & \mathcal{C}_\alpha\sigma, (E\sigma \rhd y\sigma)_{y\in l_x\setminus\{x\}}, E\sigma \rhd t\sigma, \mathcal{C}_\beta\sigma \end{array}$$

where the first $\equiv$ is by Lemma 9, and the second one by Lemma 10. Thus the resulting system is equivalent for solutions to one in which $l_x \subseteq Var(l_1, \ldots, l_n)$. We can then apply the same reasoning as above.

■

We may now conclude by applying the previous results and again König's Lemma.

Theorem 12 *Let $\mathcal{I}_0 = \langle \mathcal{F}, \mathcal{L}_0, \mathcal{H}\rangle$ be a deduction system such that the saturation of $\mathcal{L}_0$ terminates , and the resulting deduction system is contracting. Then the $\mathcal{I}_0$-reachability problem is decidable.*

PROOF.

It suffices to prove that the application of rules of Fig. 5.3 terminates. Assume there exists a modified $\mathcal{I}'$-constraint system $\mathcal{C}$ and an infinite sequence of transformations starting from $\mathcal{C}$. Let $\mathcal{C}_1, \ldots, \mathcal{C}_n, \ldots$ be the resulting sequence of modified constraint systems. By Lemma 50, at each step $nbv(\mathcal{C}_i) \geq nbv(\mathcal{C}_{i+1})$ and if there is equality, then the substitution applied on $\mathcal{C}_i$ is the identity (does not instantiate the variables of $\mathcal{C}$). Since we must have a positive number of variables, there is only a finite number of steps where the substitution is not the identity. Let $\mathcal{C}_n$ be the last obtained modified constraint system with $nbv(\mathcal{C}_{n-1}) > nbv(\mathcal{C}_n)$. Since all subsequent transformation do not instantiate the variables of $\mathcal{C}_n$ and its successor, the sequence has only a finite number of different modified constraint systems.

Since $\mathcal{L}'$ is finite, each modified constraint system has only a finite number of successors. Thus by König Lemma there is only a finite number of different modified constraint systems. ■

5.6 Applications: some relevant equational theories

We give here some examples of well-known equational theories where the saturation applied on the corresponding initial set of deduction rules terminates.

5.6.1 Dolev-Yao theory with explicit destructors

The Dolev-Yao theory with explicit destructors is the classical Dolev-Yao model with explicit destructors such as decryption and projections. This theory is given by the following set of equations:

$$\mathcal{H}_{DV} = \begin{cases} dec^s(enc^s(x,y),y) = x, \\ enc^s(dec^s(x,y),y) = x, \\ dec^p(enc^p(x,Pk(y)),Sk(y)) = x, \\ enc^p(dec^p(x,Sk(y)),Pk(y)) = x, \\ \pi_1(\langle x,y\rangle) = x, \\ \pi_2(\langle x,y\rangle) = y. \end{cases}$$

By orienting equations of $\mathcal{H}_{DV}$ from left to right, we obtain a rewrite system $\mathcal{R}_{DV}$ generating $\mathcal{H}_{DV}$. We remark that $\mathcal{R}_{DV}$ is convergent and $\mathcal{H}_{DV}$ has finite variant property.

The initial set of deduction rules is given by the following set of rules:

$$\mathcal{L}_0 = \begin{cases} x, y \to \langle x,y\rangle, \\ x \to \pi_1(x), \\ x \to \pi_2(x), \\ x, y \to enc^p(x,y), \\ x, y \to dec^p(x,y), \\ x, y \to enc^s(x,y), \\ x, y \to dec^s(x,y). \end{cases}$$

The saturation (modulo the simplification introduced after the lemma 10) outputs the following set of deduction rules:

$$\mathcal{L}' = \mathcal{L}_0 \cup \begin{cases} \langle x,y\rangle \to x, \\ \langle x,y\rangle \to y, \\ dec^p(x,Sk(y)),Pk(y) \to x, \\ enc^p(x,Pk(y)),Sk(y) \to x, \\ dec^s(x,y),y \to x, \\ enc^s(x,y),y \to x, \\ x,Pk(y),Sk(y) \to x. \end{cases}$$

5.6.2 Digital signature theory with duplicate signature key selection property

The theory of digital signature with duplicate signature key selection property is defined in [66] and is given by the following set of equations:

$$\mathcal{H}_{DSKS} = \begin{cases} ver(x,sig(x,Sk(y)),Pk(y)) = 1, \\ ver(x,sig(x,Sk'(y_1,y_2)),Pk'(y_1,y_2)) = 1, \\ sig(x,Sk'(Pk(y),sig(x,Sk(y)))) = sig(x,Sk(y)). \end{cases}$$

The equational theory $\mathcal{H}_{DSKS}$ is generated by:

$$\mathcal{R}_{DSKS} = \begin{cases} ver(x,sig(x,Sk(y)),Pk(y)) \to 1, \\ ver(x,sig(x,Sk'(y_1,y_2)),Pk'(y_1,y_2)) \to 1, \\ ver(x,sig(x,Sk(y)),Pk'(Pk(y),sig(x,Sk(y)))) \to 1, \\ sig(x,Sk'(Pk(y),sig(x,Sk(y)))) \to sig(x,Sk(y)). \end{cases}$$

We remark that $\mathcal{R}_{DSKS}$ is convergent and $\mathcal{H}_{DSKS}$ has the finite variant property.

The initial set of deduction rules is given by the following set of rules:

$$\mathcal{L}_0 = \left\{ \begin{array}{l} x, y \rightarrow sig(x, y), \\ x, y, z \rightarrow ver(x, y, z), \\ x, y \rightarrow Sk'(x, y), \\ x, y \rightarrow Pk'(x, y), \\ \emptyset \rightarrow 1. \end{array} \right.$$

The saturation (modulo the simplification introduced in Section 5.4) outputs the following set of deduction rules:

$$\mathcal{L}' = \mathcal{L}_0 \cup \left\{ \begin{array}{l} x, sig(x, Sk(y)), Pk(y) \rightarrow 1, \\ x, sig(x, Sk'(y_1, y_2)), Pk'(y_1, y_2) \rightarrow 1, \\ x, sig(x, Sk(y)), Pk'(Pk(y), sig(x, Sk(y))) \rightarrow 1, \\ x, Sk'(Pk(y), sig(x, Sk(y))) \rightarrow sig(x, Sk(y)), \\ Sk(y), Pk(y) \rightarrow 1, \\ Sk'(y_1, y_2), Pk'(y_1, y_2) \rightarrow 1, \\ x, Sk(y), Pk'(Pk(y), sig(x, Sk(y))) \rightarrow 1, \\ x, Pk(y), sig(x, Sk(y)) \rightarrow sig(x, Sk(y)), \\ x, Pk(y), Sk(y) \rightarrow sig(x, Sk(y)), \\ y_1, y_2, Pk'(y_1, y_2) \rightarrow 1, \\ x, y_1, y_2, sig(x, Sk'(y_1, y_2)) \rightarrow 1, \\ y_1, y_2, Sk'(y_1, y_2) \rightarrow 1, \\ x, Pk(y), sig(x, Sk(y)) \rightarrow 1, \\ x, Pk(y), Sk(y), sig(x, Sk(y)) \rightarrow 1, \\ x, Sk(y), Pk(y), Pk'(Pk(y), sig(x, Sk(y))) \rightarrow 1, \\ x, Sk(y), Pk(y) \rightarrow sig(x, Sk(y)). \end{array} \right.$$

5.7 Decidability of ground reachability problems for the blind signature theory

Blind signature was introduced in [136], it is defined by the signature $\mathcal{F}_{BS} = \{sig, ver, Bl, Ubl, Pk, Sk\}$ which satisfies the following set of equations:

$$\mathcal{H}_{BS} = \left\{ \begin{array}{l} ver(sig(x, Sk(y)), Pk(y)) = x, \\ Ubl(Bl(x, y), y) = x, \\ Ubl(sig(Bl(x, y), Sk(z)), y) = sig(x, Sk(z)). \end{array} \right.$$

Let $\mathcal{R}_{BS}$ be the set of rules obtained by orienting equations of $\mathcal{H}_{BS}$ from left to right, $\mathcal{R}_{BS}$ is convergent and it is obvious that any basic narrowing derivation (Definition 13) issuing from any of the right hand side term of the rules of $\mathcal{R}_{BS}$ terminates. This implies that any narrowing derivation (and in particular basic narrowing derivation) issuing from any term terminates (Theorem 1) and thus $\mathcal{H}_{BS}$ has finite variant property [86].

The initial deduction system is given by the tuple $\mathcal{I}_0 = \langle \mathcal{F}_{BS}, \mathcal{L}_0, \mathcal{H}_{BS} \rangle$ and we have:

$$\mathcal{L}_0 = \begin{cases} 1: \ x, y \rightarrow sig(x, y), \\ 2: \ x, y \rightarrow ver(x, y), \\ 3: \ x, y \rightarrow Bl(x, y), \\ 4: \ x, y \rightarrow Ubl(x, y). \end{cases}$$

The initialisation step of saturation (Figure 5.1) outputs the following set of deduction rules:

$$\mathcal{L} = \mathcal{L}_0 \cup \begin{cases} 5: \ sig(x, Sk(y)), Pk(y) \rightarrow x, \\ 6: \ Bl(x, y), y \rightarrow x, \\ 7: \ sig(Bl(x, y), Sk(z)), y \rightarrow sig(x, Sk(z)). \end{cases}$$

We define a new deduction system $\mathcal{I} = \langle \mathcal{F}_{BS}, \mathcal{L}, \emptyset \rangle$ and by lemma 31, we have: $t \in \overline{E}^{\mathcal{I}_0}$ if and only if $t \in \overline{E}^{\mathcal{I}}$ for every set of ground terms E (respectively a ground term t) in normal form. From now we remark that the equational theory employed is the empty one.

Now, let us apply the first step of saturation. The closure applied on rules 1 and 5 outputs the rule $8: \ x, Sk(y), Pk(y) \rightarrow x$, the closure applied on rules 3 and 6 outputs the rule $9: \ x, y \rightarrow x$ which will be deleted by the simplification step introduced in Section 5.4. The closure applied on rules 1 and 7 outputs the rule $10: \ y, Bl(x, y), Sk(z) \rightarrow sig(x, Sk(z))$.

We prove in the next lemma that the last rule is redundant when the employed equational theory is the empty one.

Lemma 51 *Let* $\mathcal{L}'_1 = \mathcal{L} \cup \{x, Sk(y), Pk(y) \rightarrow x\} \cup \{y, Bl(x, y), Sk(z) \rightarrow sig(x, Sk(z))\}$ *and let* $\mathcal{L}'_2 = \mathcal{L}'_1 \setminus \{y, Bl(x, y), Sk(z) \rightarrow sig(x, Sk(z))\}$. *Suppose that the employed equational theory is the empty one. For any two sets of ground terms in normal form E and F we have: $E \rightarrow^*_{\mathcal{L}'_2} F$ if and only if $E \rightarrow^*_{\mathcal{L}'_1} F$.*

PROOF.

Let E and F be two sets of normal ground terms. The direct implication is obvious, let us prove the second one. Suppose that $E \rightarrow^*_{\mathcal{L}'_1} F$ and let us prove that $E \rightarrow^*_{\mathcal{L}'_2} F$. Suppose that in the $\mathcal{L}'_1$-derivation D starting from E to F there is some steps where the applied rule is in $\mathcal{L}'_1 \setminus \mathcal{L}'_2$ that is, by definition of $\mathcal{L}'_1$ and $\mathcal{L}'_2$, the applied rule is $y, Bl(x, y), Sk(z) \rightarrow sig(x, Sk(z))$.

Let i be the first step in the derivation where the applied rule is $y, Bl(x, y), Sk(z) \rightarrow sig(x, Sk(z))$, we prove that this step can be replaced by other steps where the respectives applied rules are in $\mathcal{L}'_2$. $D: \ E = E_0 \rightarrow \ldots \rightarrow E_i \rightarrow_{y, Bl(x,y), Sk(z) \rightarrow Sig(x, Sk(z))} E_{i+1} \ldots \rightarrow F$. There is a ground substitution σ in normal form such that $\{y\sigma, Bl(x, y)\sigma, Sk(z)\sigma\} \subseteq E_i$ and $E_{i+1} = E_i \cup sig(x\sigma, Sk(z)\sigma)$. Thus, the rule $Bl(x, y), y \rightarrow x \in \mathcal{L}'_2$ with the substitution σ can be applied first on E_i and outputs $E_{i1} = E_i \cup x\sigma$, then the rule $x, y \rightarrow sig(x, y) \in \mathcal{L}'_2$ also with the substitution σ can be applied on E_{i1} and outputs $E_{i1} \cup sig(x\sigma, Sk(z\sigma)) = E_{i+1}$. We deduce that each application of the

rule $y, Bl(x,y), Sk(z) \rightarrow sig(x, Sk(z))$ in D can be replaced by the application of two rules in $\mathcal{L}'_2$. We conclude that $E \rightarrow^*_{\mathcal{L}'_1} F$ implies $E \rightarrow^*_{\mathcal{L}'_2} F$. ■

Remarks.

Enforcing the termination of the Saturation. The application of the *Saturation* algorithm as is described in section 5.3.1 does not terminate. In fact, the rule 10 is an increasing one and *closure* rule can be applied on rules 10 and 7. The application of the closure outputs the rule $11 : \ y, y', Bl(Bl(x,y), y'), Sk(z) \rightarrow sig(x, Sk(z))$ which is increasing. We remark that *closure* rule can be applied on the rules 11 and 7 and this application outputs a new increasing rule. In addition, *closure* rule can be applied again on the new obtained rule and the rule 7. We remark also that each such application of closure rule outputs a new increasing rule where the size of the terms in the left hand side is increased and closure rule can be applied again on this new obtained rule and the rule 7. This implies that we have an infinite sequence of application of *closure* rule. We remark that this infinite sequence is due to the presence of the rule 10.

As a consequence from the previous lemma (where we prove that the rule $y, Bl(x,y), Sk(z) \rightarrow sig(x, Sk(z))$ is redundant), we can delete this rule from the system immediately after its creation. This deletion enforces the termination of the *Saturation*.

Saturated deduction system. Let $\mathcal{I}' = \langle \mathcal{F}_{BS}, \mathcal{L}', \emptyset \rangle$ be the saturated deduction system (obtained after enforcing the termination of the saturation), we have:

$$\mathcal{L}' = \mathcal{L}_0 \cup \begin{cases} sig(x, Sk(y)), Pk(y) \rightarrow x, \\ Bl(x,y), y \rightarrow x, \\ sig(Bl(x,y), Sk(z)), y \rightarrow sig(x, Sk(z)), \\ x, Sk(y), Pk(y) \rightarrow x. \end{cases}$$

In $\mathcal{L}'$, we note that only $\mathcal{L}_0$-rules are increasing and the others are decreasing (by definition of *increasing* and *decreasing* rules).

Following Lemma 41, it is easy to see that for any set of ground terms E in normal form and any ground term t in normal form, we have: $t \in \overline{E}^{\mathcal{I}_0}$ if and only if $t \in \overline{E}^{\mathcal{I}'}$.

We recall that a derivation D starting from E of goal t is *well-formed* if for all rules $l \rightarrow r$ applied with substitution σ, for all $u \in l \setminus \mathcal{X}$ we have either $u\sigma \in E$ or $u\sigma$ was constructed by a former decreasing rule.

Lemma 52 *Let E (respectively t) be a set of terms (respectively a term) in normal form such that $t \in \overline{E}^{\mathcal{I}'}$. For all $\mathcal{I}'$-derivations D starting from E of goal t we have either D*

is well-formed or there is another $\mathcal{I}'$-derivation D' starting from E of goal t such that $Cons(D) \subseteq Cons(D')$ and D' is well-formed.

PROOF.

We have $t \in \overline{E}^{\mathcal{I}'}$ implies that the set $\Omega(E,t)$ of $\mathcal{I}'$-derivations starting from E of goal t is not empty. Let $D \in \Omega(E,t)$, $D : E = E_0 \to E_1 \to \ldots \to E_{n-1}, t$, we denote $\widetilde{l_i} \to r_i$ the rule applied at step i with the substitution σ_i: this rule is well-applied if for all $u \in \widetilde{l_i} \setminus \mathcal{X}$, we have either $u\sigma \in E$ or $u\sigma$ was obtained by a former decreasing rule, otherwise it is bad-applied.

Suppose that D is not well-formed then there is at least one step in the derivation D where the applied rule is bad-applied. At each such step, one the following rule is applied:

$$\begin{cases} sig(x, Sk(y)), Pk(y) \to x, \\ Bl(x,y), y \to x, \\ sig(Bl(x,y), Sk(z)), y \to sig(x, Sk(z)), \end{cases}$$

We note that the rule $x, Sk(y), Pk(y) \to x$ can not be applied at such step because the rules $y \to Sk(y)$ and $y \to Pk(y)$ are not in $\mathcal{L}'$.

Let us prove that each application of the first (respectively the second) rule in D such that there is a non variable term in left hand side of the rule where the instance is obtained by a former increasing rule can be deleted from D without altering $Cons(D)$. Let i be the first step where the first (respectively the second) rule is bad applied, that is there is a non variable term in left hand side where the instance is obtained by a former increasing rule. There is only one non variable term in the left hand side of the first (respectively the second) rule which can be obtained by a former increasing rule, this term is $sig(x, Sk(y))$ (respectively $Bl(x,y)$). Since the instance of this term, $sig(x, Sk(y))\sigma$ (respectively $Bl(x,y)\sigma$), is obtained by a former increasing rule this last rule will be $x, y \to sig(x,y)$ (respectively $x, y \to Bl(x,y)$) and let h ($h < i$) be the step where this rule is applied. We deduce that $\{x\sigma, Sk(y\sigma)\}$ $(respectively\ \{x\sigma, y\sigma\}) \subseteq E_h$ and then the rule applied at step i (which adds $x\sigma$) does not add a new term and the step i can be deleted without modifying in $Cons(D)$. Let D' be the obtained derivation, we have $Cons(D') = Cons(D)$. We deduce that every step in D' where the rule $sig(x, Sk(y)), Pk(y) \to x$ (respectively the rule $Bl(x,y), y \to x$) is bad applied can be deleted without altering in the trace of D' and let d be the obtained derivation. We note that every application of the rule $sig(x, Sk(y)), Pk(y) \to x$ (respectively the rule $Bl(x,y), y \to x$) in d is a well-application.

Suppose that d is not well-formed then there is at least one step where the rule applied is bad-applied. Let i be the first such step and let the applied rule be $sig(Bl(x,y), Sk(z)), y \to sig(x, Sk(z))$ and $Sig(Bl(x,y), SK(z))\sigma$ is obtained by a former increasing rule, $x, y \to sig(x,y)$. Let $h < i$ be the step at which this increasing rule is applied. We deduce that $\{Bl(x,y)\sigma, Sk(z)\sigma\} \subseteq E_h$. If $x\sigma \notin E_i$ then the rule applied at step i in d can be replaced first by the ap-

plication of $Bl(x,y), y \rightarrow x$ then the application of $x, y \rightarrow sig(x,y)$. Let d' be the obtained derivation, $d' : E \rightarrow \ldots \rightarrow E_i \rightarrow_{Bl(x,y),y\rightarrow x} E_i, x\sigma \rightarrow_{x,y\rightarrow sig(x,y)} E_i, x\sigma, sig(x, Sk(z))\sigma \rightarrow \ldots \rightarrow E_{n-1}, t$. By above and since $x\sigma \notin E_i$ we have either $Bl(x,y)\sigma \in E$ or $Bl(x,y)\sigma$ is obtained by a former decreasing rule.

If $x\sigma \in E_i$ then the rule applied at step i in d can be replaced by the application of $x, y \rightarrow sig(x,y)$. Let d'' be the obtained derivation, $d'' : E \rightarrow \ldots \rightarrow E_i \rightarrow_{x,y\rightarrow sig(x,y)} E_i, sig(x, Sk(z))\sigma \rightarrow \ldots \rightarrow E_{n-1}, t$.

This implies that each bad application of the rule $sig(Bl(x,y), Sk(z)), y \rightarrow sig(x, Sk(z))$ can be replaced by one (or two) well-applied rules. We deduce that if the derivation D is not well-formed there is another well-formed derivation D'' starting from E of goal t such that $Cons(D) \subseteq Cons(D'')$. ■

We recall that $\mathcal{I}_0 = \langle \mathcal{F}_{BS}, \mathcal{T}_{\mathcal{I}_0}, \mathcal{H}_{BS} \rangle$ and $\mathcal{I}' = \langle \mathcal{F}_{BS}, \mathcal{L}_{\mathcal{I}'}, \emptyset \rangle$. We recall also that $\mathcal{H}_{BS}$ has the finite variant property. The construction of $\mathcal{I}'$ and Lemma 52 implies that the conditions $Variant$, SOL_1 and SOL_2 from Definition 52 are satisfied, and hence we conclude that $\mathcal{I}'$ is $\mathcal{I}_0$-strongly order local.

In order to solve $\mathcal{I}_0$-ground reachability problems, we apply the algorithm defined in section 5.4. Since $\mathcal{L}'_{\mathcal{I}}$ is finite and $\mathcal{I}'$ is $\mathcal{I}_0$-strongly order local, by lemmas (43, 44, 46, 47 and 48) we deduce the following theorem:

Theorem 13 *The $\mathcal{I}_0$-ground reachability problem is decidable.*

5.8 Decidability of reachability problems for subterm convergent theories

In this section, we give a decidability result for the reachability problems for a class of subterm convergent equational theories. We recall that subterm convergent equational theories have finite variant property [86]. The result of this section is entailed by a more general result by Baudet [31], but the proof here in this specific case is much shorter.

We recall that $\mathcal{F}$ is a set of functions symbols and we denote by $\mathcal{H}$ a subterm convergent equational theory and by $\mathcal{I}_0 = \langle \mathcal{F}, \mathcal{L}_0, \mathcal{H} \rangle$ the initial deduction system such that $\mathcal{L}_0$ is the union of functions $x_1, \ldots, x_n \rightarrow f(x_1, \ldots, x_n)$ for some function symbols $f \in \mathcal{F}$.

Definition 54 *(Subterm convergent theories.) An equational theory $\mathcal{H}$ is subterm convergent if it is generated by a convergent rewriting system $\mathcal{R}$ and for each rule $l \rightarrow r \in \mathcal{R}$, r is a strict subterm of l.*

In the rest of this section, we give an algorithm to decide the following reachability problem, "$\mathcal{I}_0$-Reachability Problem":

Input: An $\mathcal{I}_0$-constraint system $\mathcal{C}$.

Output: SAT if and only if there exists a substitution σ such that $\sigma \models_{\mathcal{I}_0} \mathcal{C}$.

We let $\mathcal{I}' = \langle \mathcal{F}, \mathcal{L}', \emptyset \rangle$ to be the saturated deduction system, we recall that $\mathcal{I}'$ is $\mathcal{I}_0$-strongly order local. We suppose that $r \notin \widetilde{l}$ for all rules $\widetilde{l} \rightarrow r \in \mathcal{L}'$ that is rules not satisfying this property will be deleted.

In the following lemma we prove that, in the case of subterm convergent equational theories and under our assumption on the form of initial deduction rules $\mathcal{L}_0$, *Saturation* terminates and the obtained new rules are decreasing.

Lemma 53 *The saturation of $\mathcal{L}_0$ terminates and for every rule $\widetilde{l} \rightarrow r \in \mathcal{L}' \setminus \mathcal{L}_0$ there exists a term $s \in \widetilde{l}$ such that r is a strict subterm of s.*

PROOF.

Let $\widetilde{l} \rightarrow r \in \mathcal{L}' \setminus \mathcal{L}_0$ and let us prove that this rule satisfies the following property: there is a term $s \in \widetilde{l}$ such that $r \in SSub(s)$. By induction on the number of saturations needed to obtain a rule $\widetilde{l} \rightarrow r$.

Let us first prove this property is true for rules obtained by the step 1 of the saturation. By definition of $\mathcal{H}$, by the fact that variants of term are in normal form and given the assumption that all original rules are $x_1, \ldots, x_n \rightarrow f(x_1, \ldots, x_n)$, this implies:

$$(f(x_1, \ldots, x_n)\theta)\downarrow \in SSub(f(x_1, \ldots, x_n)\theta)$$

Thus, there exists i such that: $$(f(x_1, \ldots, x_n)\theta)\downarrow \in Sub(x_i\theta)$$

If there is equality, the rule is removed (since $r \notin \widetilde{l}$ for all rules $\widetilde{l} \rightarrow r$). This implies that all rules obtained from step 1 of saturation satisfies the property. Since $\mathcal{L}_0$ is finite and since subterm convergent equational theories have finite variant property [86], first step of saturation terminates. Since $u \in SSub(v)$ implies $u \prec v$, rules obtained by step 1 are decreasing. Let $\mathcal{L}$ be the set of rules obtained by step 1 and let us prove that rules obtained by closure satisfy the property. Let us prove it for the first rule obtained by closure. By definition of closure rule and since rules in $\mathcal{L} \setminus \mathcal{L}_0$ are decreasing, the first closure will be applied on rules $x_1 \ldots, x_n \rightarrow f(x_1, \ldots, x_n) \in \mathcal{L}_0$ and $f(s_1, \ldots, s_n), \widetilde{l} \rightarrow r \in \mathcal{L} \setminus \mathcal{L}_0$. Again by definition of closure, the obtained rule is $s_1, \ldots, s_n, \widetilde{l} \rightarrow r$. By definition of decreasing rule, there is a term $u \in \left\{ f(s_1, \ldots, s_n), \widetilde{l} \right\}$ such that $r \in SSub(u)$, if $u = \widetilde{l}$ then the new rule satisfies the property and if $u = f(s_1, \ldots, s_n)$ then there is an integer i such that $r \in Subs_i$. If $r \in SSubs_i$ the obtained rule satisfies the property else the rule can not be in $\mathcal{L}'$ (since rules $l \rightarrow r$ with $r \in l$ are deleted). We conclude that the first rule obtained by closure is decreasing and if we apply again closure, it will be applied on a rule in $\mathcal{L}_0$ and a rule not in $\mathcal{L}_0$. We conclude that rules obtained by step 2 satisfy the property and are decreasing . We conclude also that step 2 terminates. ■

We recall that increasing rules are of form $x_1, \ldots, x_n \rightarrow f(x_1, \ldots, x_n)$ for a function symbol $f \in \mathcal{F}$ (Lemma 53).

5.8.1 Decidability result

We recall that our goal is to solve $\mathcal{I}_0$-reachability problem.

Algorithm. Let $\mathcal{C}^0 = ((E_i^0 \vdash v_i^0)_{i\in\{1,\ldots,n\}}, \mathcal{U}^0)$.

Step 1: Guess a variant $\mathcal{I}_0$-constraint system associated to $\mathcal{C}^0$. Let $\mathcal{C} = (E'_1 \rhd t'_1, \ldots, E'_n \rhd t'_n)$ be the variant $\mathcal{I}_0$-constraint system associated to $\mathcal{C}^0$. $\mathcal{C}$ is constructed from $\mathcal{C}^0$ as given in Definition 28 (Chapter 2).

We remark that this step terminates and it is also correct (Lemma 44) and complete (Lemma 43). Unless otherwise specified, $\mathcal{I}'$ is the deduction system implicit in all notations in the rest of this section.

We now introduce the notation $\rhd_{inc}$ to denote a deduction constraint that has to be solved using only increasing rules. We say a constraint $E \rhd_{inc} t$ is in solved form if t is a variable. The constraint system is in solved form if all the deduction constraints are in solved form. The application of a decreasing rule $\widetilde{l} \rightarrow r$ on a constraint $E \rhd t$ is defined as follows, and in accordance with the fact that $\mathcal{I}'$ is $\mathcal{I}_0$-strongly order local.

- let σ be the mgu of the terms in $\widetilde{l} \setminus \mathcal{X}$ with a subset F of $E \setminus \mathcal{X}$
- if $\{x_1, \ldots, x_k\} = \widetilde{l} \cap \mathcal{X}$, replace $\mathcal{C}_\alpha, E \rhd t, \mathcal{C}_\beta$ with:

$$(\mathcal{C}_\alpha, E \rhd_{inc} x_1, \ldots, E \rhd_{inc} x_k, E \cup \{r\} \rhd t, \mathcal{C}'_\beta)\sigma$$

  Where $\mathcal{C}'_\beta$ is constructed from $\mathcal{C}_\beta$ by adding r to each left-hand side. This last construction aims at preserving the inclusion of knowledge sets.

Step 2. Iterate until the constraint system is in solved form or unsolvable:

1. Put all tagged deduction constraints $E \rhd_{inc} t$ in solved form;
2. If all constraints preceding an untagged $E \rhd t$ are in solved form, Apply non-deterministically $|Sub(E) \setminus Var(E)|$ decreasing rules on E. Replace $E \rhd t$ by the obtained deduction constraints, all tagged with inc.

Let us prove the completeness and termination of Step 2.

Completeness. The proof of the following lemma is trivial by the form of increasing rules.

Lemma 54 *If $\sigma \models E \rhd_{inc} f(t_1, \ldots, t_n)$ then either $f(t_1, \ldots, t_n)\sigma \in E\sigma$ or $x_1, \ldots, x_n \rightarrow f(x_1, \ldots, x_n)$ will be in $\mathcal{L}_0$ and for each $i \in \{1, \ldots, n\}$ we have $\sigma \models E \rhd_{inc} t_i$.*

The first part of the iteration consists either in transforming a deduction constraint $E \rhd_{inc} f(t_1, \ldots, t_n)$ into $E \rhd_{inc} t_1, \ldots, E \rhd_{inc} t_n$, or in unifying $f(t_1, \ldots, t_n)$ with $e \in E$. By Lemma 54, given a ground substitution σ such that $\sigma \models E \rhd_{inc} f(t_1, \ldots, t_n)$ there exists a sequence of choices reducing $E \rhd_{inc} f(t_1, \ldots, t_n)$ to a (possibly empty) set of deduction constraints $E\tau \rhd_{inc} u_1, \ldots E\tau \rhd_{inc} u_k$ where the $u_1, \ldots, u_k$ are variables or constants. If there is a constant which is not in $E\tau$ the constraint is not satisfiable (by definition of increasing rules), and the sequence of choices fails.

Let us now consider the second part of the iteration.

Lemma 55 *Assume $\sigma \models E \rhd_{inc} x$ with x the first variable in the sequence of deduction constraints such that $t \in Sub(x\sigma)$ for some ground term t. Then either there exists $u \in Sub(E)$ such that $u\sigma = t$ or $t \in \overline{E\sigma}^{\mathcal{L}'_{inc}}$.*

PROOF.

Let us assume there does not exist $u \in Sub(E)$ such that $u\sigma = t$. By minimality of x and the determinacy of constraint systems we have $t \notin Sub(Var(E)\sigma)$. Since $Sub(E\sigma) = Sub(E)\sigma \cup Sub(Var(E)\sigma)$ we have $t \notin Sub(E\sigma)$ and, by hypothesis on x and t, $t \in Sub(x\sigma)$. Since $\sigma \models E \rhd_{inc} x$ consider a derivation $E_1 = E\sigma \rightarrow \ldots \rightarrow E_{n-1} \cup x\sigma$, and let i be minimal such that $t \in Sub(E_i)$. The index i exists since $t \in Sub(x\sigma)$, and is different from 1 since $t \notin Sub(E\sigma)$. By definition of the increasing rules we then must have $E_i = E_{i-1}, t$. ■

Consider a $\mathcal{I}'$-constraint system $\mathcal{C} = (\mathcal{C}_\alpha, E \rhd t, \mathcal{C}_\beta)$ satisfied by a substitution σ and all deduction constraints in $\mathcal{C}_\alpha$ are in solved form. By Lemma 53 and by the facts that $\mathcal{I}'$ is $\mathcal{I}_0$-strongly order local and $r \notin \widetilde{l}$ for all rules $\widetilde{l} \rightarrow r \in \mathcal{L}'$, all decreasing rules applied on $E\sigma$ yield a term in $Sub(E\sigma)$. Thus there are at most $|Sub(E) \setminus Var(E)|$ different terms that can be obtained by decreasing rule starting from $E\sigma$ and which are not in $Sub(Var(E)\sigma)$. Assume a term t is in $Sub(Var(E)\sigma) \setminus Sub(E)\sigma$, and let x be the first variable (in the ordering of deduction constraints) such that $t \in Sub(x\sigma)$. By definition of constraint systems there exists a deduction constraint $E_x \rhd_{inc} x$ in $\mathcal{C}_\alpha$. Since $E_x \subseteq E$, by Lemma 55, we have $t \in \overline{E_x\sigma}^{\mathcal{L}'_{inc}}$. Again, since $E_x\sigma \subseteq E\sigma$, this implies $t \in \overline{E\sigma}^{\mathcal{L}'_{inc}}$: the decreasing rule was not useful, and can be replaced by a sequence of derivation ending with an increasing rule. Thus in $\overline{E\sigma}$ at most $|Sub(E) \setminus Var(E)|$ terms are deducible using decreasing rules that can not be deduced

with increasing rules. Thus, after a right choice of at most $|Sub(E) \setminus Var(E)|$ decreasing rules, all terms deducible from the obtained knowledge set can be deduced using only increasing rules, hence the tagging with inc of the final deduction constraint $E \cup \{r_1, \ldots, r_k\} \rhd_{inc} t$, $k = |Sub(E) \setminus Var(E)|$.

Termination of Step 2. First let us notice that if a unification is chosen, it unifies two subterms of the constraint system in the empty theory, and thus either the two terms were already equal or it reduces strictly the number of variables in the constraint system. Thus the number of unification choices is bounded by the number of variables in the constraint system. Once all unification have been performed, the termination of the first part of the iteration can easily be proved by considering the multiset of the right-hand side of the deduction constraints, ordered by the extension to multisets of the (well-founded) subterm ordering. The second part of the iteration obviously terminates. Thus each iteration terminates. Since each iteration decreases strictly the number of non-labelled deduction constraints, Step 2. terminates.

5.9 Related works

Several decidability results have been obtained for cryptographic protocols in a similar setting [15, 156, 16, 178]. These results have been extended to handle algebraic properties of cryptographic primitives [68, 70, 66, 9]. In [10], a decidability result was given to the ground reachability problems in the case of subterm convergent equational theories. This result was extended in [9] and a decidability result to the ground reachability problems in the case of locally stable AC-convergent equational theories was given. Moreover, again in [9], a decidability result was given to the ground reachability problems for the theory of blind signature [136] while this theory was not included in [10]. To obtain a decidability result for the theory of blind signature, M. Abadi and V. Cortier [9] use a new extended definition of subterm. The result obtained in [10] was extended in [31] in different way than in [9] and a decidability result was obtained to the general reachability problem for the class of subterm convergent equational theory. The first result of our chapter is a decidability result to the ground reachability problems for a class of equational theories which includes the class studied in [10]. We note that the class studied in [9] is incomparable to ours and we note also that the proof used in [9] to decide the ground reachability problems for the theory of blind signature is different from the ours. Another result of this chapter is a decidability result to the general reachability problem for a class of equational theories under some conditions on the deduction systems and the class studied in [31] is incomparable with ours. In [40], a decidability result was given to the general reachability problems under some

syntactic conditions on the intruder deduction rules, this result is incomparable with ours.

5.10 Conclusion

In [80], H. Comon-Lundh proposes a two-steps strategy for solving general reachability problems: first, decide ground reachability problems and, second, reduce general reachability problems to ground reachability ones, *e.g.* by providing a bound on the size of a minimal solution of a problem. Our results are in this line: for *contracting* deduction systems, general reachability can be reduced to ground reachability. We strongly conjecture that it permits one to provide a bound on the size of minimal solutions. Thus, this chapter adds a new criterion to the one already known for deciding reachability problems. In future works, we will investigate how the construction presented here can be extended to equational theories having the finite variant property w.r.t. a non-empty equational theory. We will also try to weaken the definition of $\mu(T)$ for a set of terms T. In the next chapter, we employ similar techniques to solve ground entailment problems for saturated first-order theories.

Chapter 6

Decidability of ground entailment problems for saturated sets of clauses

In this chapter, we study the ground entailment problems in the first order logic, and more precisely the ground entailment problems for a set of first order clauses. For clauses and Horn clauses sets, the satisfiability and ground entailment problems are undecidable. Many decidability results have been obtained for several fragments of first order logic [150, 28, 84, 180, 205]. In this chapter, we introduce a new fragment of first order logic and we prove the decidability of its ground entailment problem using the selected resolution, and a special ordering over atoms and terms. We then show how to use this result in order to decide the insecurity problem for cryptographic protocols in the case of bounded number of sessions.

Outline of the chapter We introduce in Section 6.1 the basic notions of this chapter including clauses, ground entailment problems, and resolution methods. In Section 6.2, we mention some obtained decidability results for the ground entailment problem, and in Section 6.3 we show that the ground entailment problem and satisfiability problem in first order logic can be used to analyse cryptographic protocols and we mention some obtained results in that field. In Section 6.4, we introduce our modelisation of cryptographic protocols using first order clauses and we show how to reduce insecurity problem for cryptographic protocols to ground entailment problem in the first order logic. In Section 6.5, we present selected resolution for which we show soundness and completeness, we introduce a new fragment of first order logic and we show the

decidability of its ground entailment problem. The conclusion is given in Section 6.6.

6.1 Preliminaries

6.1.1 Basic notions

We assume that we have an infinite set of variables $\mathcal{X}$, an infinite set of constant symbols $\mathcal{C}$, a set of predicate symbols $\mathcal{P}$ and a set of function symbols $\mathcal{F}$. We associate the function $arity$ to function symbols and predicate symbols, $arity : \mathcal{F} \cup \mathcal{P} \to \mathbb{N}$. The arity of a function symbol (respectively predicate symbol) indicates the number of arguments that the function symbol (respectively the predicate symbol) expects. We define the set of *terms* $\mathcal{T}(\mathcal{F}, \mathcal{X})$ as follows: $\mathcal{X}, \mathcal{C} \subseteq \mathcal{T}(\mathcal{F}, \mathcal{X})$, and for each function symbol $f \in \mathcal{F}$ with arity $n \geq 0$, for each terms $t_1, \ldots, t_n \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, we have $f(t_1, \ldots, t_n) \in \mathcal{T}(\mathcal{F}, \mathcal{X})$. We define *atoms* as follows: if I is a predicate symbol in $\mathcal{P}$ with arity $n \geq 0$, and $t_1, \ldots, t_n$ are terms in $\mathcal{T}(\mathcal{F}, \mathcal{X})$, then $I(t_1, \ldots, t_n)$ is an atom. A *literal* L is either positive literal A or negative literal $\neg A$ where A is an atom, and $\neg$ denotes the *negation*. A *(full) clause* is a formula of the form $\Gamma \to \Delta$, where Γ and Δ are sets of atoms; Γ represents *the antecedent of the clause* and Δ *its succedent*. A clause $\{A_1, \ldots, A_m\} \to \{B_1, \ldots, B_n\}$ may also be viewed as a set of literals $\{\neg A_1, \ldots, \neg A_m, B_1, \ldots, B_n\}$. The antecedent of the clause represents *its the negative literals*, and the succedent of the clause represents *its positive literals*. A clause $\Gamma \to \Delta$ is *Horn* when Δ is a singleton or empty. A clause $\Gamma \to \Delta$ is *unit* when it has only one literal, $i.e.$ either Γ is a singleton and $\Delta = \emptyset$, or $\Gamma = \emptyset$ and Δ is a singleton. A clause $\Gamma \to \Delta$ is *positive* when it has only a succedent, $i.e.$ $\Gamma = \emptyset$ and a clause is *negative* when it has only an antecedent, $i.e.$ $\Delta = \emptyset$. We will write Γ_1, Γ_2 to indicate the union of sets and usually omit braces, for example, writing Γ, A or A, Γ for the union of $\{A\}$ and Γ or writing $A_1, \ldots, A_m \to B_1, \ldots, B_n$ for $\{A_1, \ldots, A_m\} \to \{B_1, \ldots, B_n\}$. We also make more simplifications, by writing A for the positive unit clause $\emptyset \to A$, and $\neg A$ for the negative unit clause $A \to \emptyset$. If C is a clause, by $\neg C$ we denote the set of unit Horn clauses $\neg L$, with L a literal in C, for example, $\neg C = \{A_1, \ldots, A_m, \neg B_1, \ldots, \neg B_n\}$ for $C = A_1, \ldots, A_m \to B_1, \ldots, B_n$. We say that a term t occurs in an atom A if A is of the form $I(\ldots, t, \ldots)$ and t occurs in a clause if it occurs in an atom of the clause. If M is an expression (i.e. a term, an atom, clause, or a set of such objects), and σ is a substitution, then $M\sigma$ is obtained by applying σ to M as usually defined. We also extend as usual the notation of unifiers from terms to atoms, for example, the atoms $I(a),\ I'(b, x)$ are not unifiable while the atoms $I(x),\ I(a)$ are unifiable and their most general unifier is $\sigma = \{x \mapsto a\}$. An expression is ground if it does not contain any variables.

Herbrand interpretations. A *(Herbrand) interpretation* is a set of ground atoms. A ground atom A is *true* in the interpretation $\mathfrak{Int}$ if $A \in \mathfrak{Int}$, and it is *false* otherwise. A ground clause $\Gamma \rightarrow \Delta$ is *satisfied* by $\mathfrak{Int}$ if either $\Gamma \not\subseteq \mathfrak{Int}$ or else $\Delta \cap \mathfrak{Int} \neq \emptyset$. We also say that a ground clause C is true in $\mathfrak{Int}$, if C is satisfied by $\mathfrak{Int}$, and that C is false in $\mathfrak{Int}$ otherwise. An interpretation $\mathfrak{Int}$ satisfies a non-ground clause if it satisfies all ground instances of that clause. For any interpretation $\mathfrak{Int}$, we never have $\mathfrak{Int}$ satisfies C and $\neg C$ in the same time, where C is a clause. An interpretation $\mathfrak{Int}$ is an *(Herbrand) model* of a clause C if it satisfies the clause, and $\mathfrak{Int}$ is a model of a set of clauses S if it satisfies all clauses in S. A clause (or a set of clauses) is satisfiable if it has a model, and unsatisfiable otherwise. We say that the set of clauses S *entails* the clause C (or C is *logical consequence* of S), and write $S \models C$, if C is true in every model of S. It is easy to see that $S \models C$ if and only if $S \cup \neg C$ is unsatisfiable.

Ground entailment problem. Given a set of clauses S, the *(ground) entailment problem for* S is defined as follows:
$\underline{Input}$: a ground clause C
$\underline{Output}$: entailed if and only if $S \models C$

The goal of this chapter is to provide conditions on S which make its ground entailment problem decidable.

Inference rules. An inference rule is described as follows:

$$\frac{C_1 \quad C_2 \quad \dots \quad C_n}{C}$$

where $C_1 \dots, C_n$ (the *premises*) and C (the *conclusion*) are clauses.

We call an inference system any set of inference rules.

Refutational and direct proofs. Let $\mathfrak{J}$ be an inference system, and let S be a set of clauses and C be a ground clause. A *refutational proof of* $S \models C$ is a tree whose root is labelled by the empty clause, and whose internal nodes are labelled by results of the applications of inference rules in $\mathfrak{J}$ on their direct descendents and whose leaves are ground instances of clauses in S or unit clauses $\neg L$, with L a literal in C. We call the inference rule applied on the direct descendent of the root *the end inference rule* of the proof. We say that the $\emptyset$ *clause is derived from* $S \cup \neg C$ if there exists a refutational proof of $S \models C$.

When S is a set of Horn clauses, a *direct proof of a ground atom* A is an unordered tree, labelled by atoms, whose root is A and whose subtrees are direct proof of atoms that constitute the antecedent of a ground instance of a clause in

S with succedent A. Leaves are ground instances of positive unit clauses in S. A direct proof of a ground clause $A_1, \ldots, A_n \rightarrow B$ from S is a direct proof of B from $S \cup \{A_1, \ldots, A_n\}$.

A term t occurs in either kind of proof if t occurs in a clause or in an atom labelling the proof tree and an atom A occurs in either kind of proof if A occurs in a clause labelling the proof tree. If π is a proof, by $Terms(\pi)$ (respectively $\mu(\pi)$) we denote the set of terms (respectively set of atoms) occurring in π.

An inference system $\mathfrak{J}$ is said to be *complete* if the empty clause can be derived from any unsatisfiable set of clauses. An inference system $\mathfrak{J}$ is said to be *sound* if the empty clause can not derived from any satisfiable set of clauses. It is easy to see that $S \models C$ if and only if the $\emptyset$ clause is derived from $S \cup \neg C$ for any complete and sound inference system $\mathfrak{J}$.

> In the rest of this chapter, we will be interested only by the refutational proofs, and for simplicity, we will write proofs instead of refutational proofs. In addition, when the inference system is complete and sound, we will abuse the notation and use the notation $S \models C$ to mean that there is a proof of $S \models C$.

6.1.2 Resolution

The resolution is one of the most successful methods for automated proof search. It was developed in [176]. We introduce in this section some of the well-known resolution inference strategies. Some of these strategies use orderings, transitive and irreflexive binary relations, on atoms and terms. We denote by $\succ_a$ the ordering on atoms and by $\succ_t$ the ordering on terms.

(Binary) Resolution

The (binary) resolution is described by the following two inference rules:

$$Resolution \quad \frac{\Gamma \rightarrow \Delta, A \qquad A', \Gamma' \rightarrow \Delta'}{(\Gamma, \Gamma' \rightarrow \Delta, \Delta')\alpha}$$

where α is the most general unifier of A and A'.

The clause $(\Gamma, \Gamma' \rightarrow \Delta, \Delta')\alpha$ is called a *resolvent* of the premises or a *conclusion* of the inference, and the atom $A\alpha$ is called the *resolved* atom.

$$Factoring \quad \frac{\Gamma \rightarrow \Delta, A, A'}{(\Gamma \rightarrow \Delta, A)\alpha}$$

where α is the most general unifier of A and A'.

The clause $(\Gamma \rightarrow \Delta, A)\alpha$ is called a *factor* of the premise or a *conclusion* of the inference, and the atom $A\alpha$ is called the *factored* atom.
(Binary) resolution is a *refutationally complete theorem proving method:* the empty clause (i.e., a contradiction) can be derived from any unsatisfiable set of clauses. The search of a contradiction proceeds by saturating the given set of clauses, that is, systematically applying all inferences rules until no more new clauses can be added [176].

Ordered Resolution

Ordered resolution is described by the following two inference rules:

$$Ordered\ \ resolution \quad \frac{\Gamma \rightarrow \Delta, A \qquad A', \Gamma' \rightarrow \Delta'}{(\Gamma, \Gamma' \rightarrow \Delta, \Delta')\alpha}$$

where α is the most general unifier of A and A',
$A\alpha$ is strictly maximal with respect to $\Gamma\alpha,\ \Delta\alpha$,
and $A\alpha$ is maximal with respect to $\Gamma'\alpha,\ \Delta'\alpha$.

$$Ordered\ \ factoring \quad \frac{\Gamma \rightarrow \Delta, A, A'}{(\Gamma \rightarrow \Delta, A)\alpha}$$

where α is the most general unifier of A and A',
$A\alpha$ is strictly maximal with respect to $\Gamma\alpha$,
and maximal with respect to $\Delta\alpha$.

Ordered resolution (respectively ordered factoring) inference rule requires that there is no atom in the conclusion greater than the resolved (respectively factored) atom. Ordered resolution, *i.e.* ordered resolution inference rule together with ordered factoring rule, is refutationally complete and sound, and hence, for any set S of clauses, S is unsatisfiable if and only if empty clause can be derived from S [24].

We remark that not every ground instance of an inference by ordered resolution is an inference by ordered resolution. For example, let Inf be the following inference by ordered resolution

$$\frac{I(y) \rightarrow I'(x) \qquad I'(x'), I(y') \rightarrow I''(y')}{I(y), I(y') \rightarrow I''(y')}$$

The most general unifier of $I'(x), I'(x')$ is $\alpha = \{x' \mapsto x\}$. By definition of ordered resolution inference, we have that $I(y\alpha) \not\geq I'(x\alpha)$, $I(y'\alpha) \not> I'(x\alpha)$, and $I''(y'\alpha) \not> I'(x\alpha)$. Let the ground substitution σ be such that $\sigma = \alpha\alpha'$, $I(y\sigma) > I(y'\sigma) > I''(y'\sigma) > I'(x\sigma)$ and $I'(x\sigma) = I'(x'\sigma)$. $Inf\sigma$ is a ground instance of Inf, and it is easy to see that $Inf\sigma$ is not an inference by ordered resolution.

Remarks. All these resolution inference rules have two premises. We implicitly suppose here that these premises do not share variables, which can be obtained by renaming the variables of one of the premises. Note that, by the definition of clauses, the same literal can not appear twice in a clause, for example the clause $A_1, A_1, A_2 \rightarrow B_1, B_2$ is not permitted. Indeed, we suppose that the resolution inference rules contain an implicit factorisation which immediately replaces $A, A, \Gamma \rightarrow \Delta$ (respectively $\Gamma \rightarrow B, B, \Delta$) by $A, \Gamma \rightarrow \Delta$ (respectively $\Gamma \rightarrow B, \Delta$).

6.1.3 Orderings

We shall make use of various ordering relations on expressions. A (strict) ordering $\succ$ on a set of elements E is a transitive and irreflexive binary relation on E. The ordering $\succ$ is said to be:

- *well-founded* if there is no infinite descending chain $e \succ e1 \succ \ldots$ for any element e in E
- *monotone* if $e \succ e'$ then $e\sigma \succ e'\sigma$ for any elements e, e' in E and any substitution σ
- *stable* if $e \succ e'$ then $u[e] \succ u[e']$ for any elements u, e and e' in E
- *subterm* if $e[e'] \succ e'$ for any elements e, e' in E
- *complete* if it is total over ground elements of E

Any ordering $\succ$ on a set E can be extended to an ordering $\succ^{set}$ on finite sets over E as follows: if η_1 and η_2 are two finite sets over E, we have $\eta_1 \succ^{set} \eta_2$ if (i) $\eta_1 \neq \eta_2$ and (ii) whenever for every $e \in \eta_2 \setminus \eta_1$ then there is $e' \in \eta_1 \setminus \eta_2$ such that $e' \succ e$. Given a set, any smaller set is obtained by replacing an element by a (possibly empty) set of strictly smaller elements. We will call an element e *maximal* (respectively *strictly maximal*) with respect to a set η of elements, if for any element e' in η we have $e' \not\succ e$ (respectively $e' \not\succeq e$). Similarly, any ordering $\succ$ on a set E can be extended to an ordering $\succ^{mul}$ on finite multisets over E as follows: if ξ_1 and ξ_2 are two finite multisets over E, we have $\xi_1 \succ^{mul} \xi_2$ if (i) $\xi_1 \neq \xi_2$ and (ii) whenever $\xi_2(e) > \xi_1(e)$ then $\xi_1(e') > \xi_2(e')$, for some e' such that $e' \succ e$; $\xi(e)$ denotes the number of occurrences of e in the multiset ξ, and $>$ denotes the standard "greater-than" relation on the natural numbers. Given a multiset, any smaller multiset is obtained by replacing an element by occurrences of smaller elements. We will call an element e *maximal* (respectively *strictly maximal*) with respect to a multiset ξ of elements, if for any element e' in ξ we have $e' \not\succ e$ (respectively $e' \not\succeq e$).

If the ordering $\succ$ is total (respectively, well-founded), so is its multiset extension [1]. It is easy to see that if the ordering $\succ$ is total (respectively, well-founded), so is its set extension: actually, by definition of multisets, a set is a multiset; since the multiset extension $\succ^{mul}$ of a total (respectively well-founded) order $\succ$ is also total (respectively well-founded) [1], we deduce that the set extension $\succ^{set}$ of a total (respectively well-founded) order $\succ$ is also total (respectively well-founded).

Clause and proof orderings. By an *atom ordering* (respectively *term ordering*) we mean an arbitrary ordering on atoms (respectively on terms). To extend an atom ordering $\succ_a$ to an ordering on clauses, we identify a (positive or negative) literal A with a set $\{A\}$, and a clause with the union of its literals, or more precisely with the union of sets of atoms identifying its literals. For example, the clause $A_1, A_2 \rightarrow B$ is identified with the following union of literals $\neg A_1 \cup \neg A_2 \cup B$, that is with the following union of sets of atoms $\{A_1\} \cup \{A_2\} \cup \{B\}$ which is equal to the set of atoms $\{A_1, A_2, B\}$. From now on, we denote by $\mu(C)$ the set of atoms representing the clause C, that is the set of atoms equal to the union of the set of atoms identifying its literals. For example $\mu(A_1, A_2 \rightarrow B) = \{A_1, A_2, B\}$. Then, for clauses C and C', we define $C \succ_c C'$ if and only if the set of atoms representing C is strictly bigger than the set of atoms representing C' for the set extension of $\succ_a$, that is $\mu(C) \succ_a{}^{set} \mu(C')$. Clearly, if the ordering $\succ_a$ is well-founded and total on ground atoms, so is its extension to ground clauses. We extend the definition of μ from a clause to a set of clauses, let S be a set of clauses, $S = \{C_1, \ldots, C_n\}$, $\mu(S) = \mu(C_1) \cup \ldots \cup \mu(C_n) = \cup_{i=1}^{n} \mu(C_i)$. By definition of proofs, each atom that appears in a proof belongs to a clause labelling one of its leaves. We extend next the atom ordering to an ordering on proofs. If π is a proof, the set $leaves(\pi)$ denotes the set of clauses labelling its leaves, and $\mu(\pi) = \mu(leaves(\pi))$. More precisely $\mu(\pi)$ is the union of set of atoms identifying clauses labelling its leaves. For example, let the proof π given as below:

$$\dfrac{\dfrac{\dfrac{\emptyset \rightarrow C \qquad \dfrac{C \rightarrow A, B \qquad B \rightarrow \emptyset}{C \rightarrow A}}{\emptyset \rightarrow A} \qquad \dfrac{A \rightarrow B \qquad B \rightarrow \emptyset}{A \rightarrow \emptyset}}{\emptyset}}{}$$

$leaves(\pi)$ is equal to the following set of clauses $\{\emptyset \rightarrow C; C \rightarrow A, B; A \rightarrow B; B \rightarrow \emptyset\}$, and $\mu(\pi) = \{A, B, C\}$. Let π, π' be two proofs, we define $\pi \succ_p \pi'$ if and only if $\mu(\pi) \succ_a{}^{set} \mu(\pi')$.

6.2 Decidable fragments of first order logic

It is known that the ground entailment problem for Horn clauses and full clauses sets is undecidable. Here, we mention some obtained decidability results under some restrictions.

6.2.1 McAllester's work

McAllester's work [150] is based on Horn clauses. He defined a set of Horn clauses S to be subterm local if for every ground Horn clause C, we have $S \models C$ if and only if C is entailed from a set of ground instances of clauses in S in which all terms are subterms of ground terms in S and C. He proved that if a set S of Horn clauses is finite and subterm local then its ground entailment problem is decidable.

6.2.2 Basin and Ganzinger work

In their work, D. Basin and H. Ganzinger [28] generalised McAllester's work by allowing arbitrary term ordering, any strict well-founded order over terms, and full (not Horn) clauses. We recall some definitions used by D. Basin and H. Ganzinger. A set of clauses S is said to be order local with respect to a term ordering $\succ$ if for every ground clause C, we have $S \models C$ if and only if C is entailed from a set of ground instances of clauses in S in which each term is smaller than or equal (under the $\succeq$) to some term in C. A term ordering $\succ$ is said to be of complexity f, g, whenever for each clause of size n (the size of a term is the number of nodes in its tree representation, and the size of a clause is the sum of sizes of its terms) there exists $O(f(n))$ terms that are smaller or equal (under $\succeq$) to a term in the clause, and that may be enumerated in time $g(n)$. It is easy to see that if $\succ$ is of complexity f, g, each ground term has finitely many smaller terms that may be enumerated in finite time. D. Basin and H. Ganzinger obtained the following results:

1. If S is a set of Horn clauses that is order local with respect to a term ordering $\succ$ of complexity f, g then the ground entailment problem for S is decidable.

2. If S is a set of (full) clauses that is order local with respect to a term ordering $\succ$ of complexity f, g then the disentailment ground problem for S is decidable.

In [28], D. Basin and H. Ganzinger showed that the saturation of a set S of Horn clauses and hence (full) clauses is not sufficient to obtain the decidability of the ground entailment problem for S. To show this, they considered the following example. We recall that for arbitrary Horn clause sets, the satisfiability and ground entailment problems are undecidable.

Example 22 *Let S be an arbitrary set of Horn clauses, and let S' consist of the set of Horn clauses $q(a), \Gamma \rightarrow A$ such that $\Gamma \rightarrow A$ is in S. Choose an ordering $\prec$ such that a is the maximal term. For any compatible atom ordering, there is no ordered inferences from S', and hence S' is saturated under ordered resolution. It is undecidable if $S' \models \neg q(a)$ since this is equivalent to the inconsistency of S.*

In [28], D. Basin and H. Ganzinger showed that restrictions on occurrences of variables can lead to decidability for full and hence also Horn clauses. This result is given as follows:

> If S is a set of (full) clauses saturated up to redundancy under ordered resolution with respect to a complete well-founded ordering over atoms, and if, for each clause in S, each of its maximal atoms contains all the variables of the clause, then the ground entailment problem for S is decidable.

6.3 From cryptographic protocols to logic of clauses

Lots of researchers [84, 82, 180, 187, 205] have been interested by clauses and Horn clauses in order to analyse cryptographic protocols. In this line of research, many models have been given. In the most common models, intruder behaviour, protocol description and security properties are encoded as clauses and in particular as Horn clauses; and, the insecurity problem of the cryptographic protocol is reduced to the satisfiability problem for a class of Horn clauses. While the satisfiability problem for first order logic is undecidable in general, several classes have been proved to be decidable [60], and in Section 6.2, we mentioned some of them. In this section, we focus on the fragments of first order logic which are well-suited to modelling security protocols. In sections 6.3.1, 6.3.2 and 6.3.3, we present some of such fragments that are shown to be decidable, and in Section 6.4 we introduce our result. In what follows, a class of clauses means a set of sets of clauses. In Sections 6.3.1 and 6.3.2, we assume a unique unary predicate symbol I.

6.3.1 Comon-Lundh and Cortier work

H. Comon-Lundh and V. Cortier [84] were focused on the secrecy property. In [82], they have showed that, for the secrecy property, it is sufficient to verify the correctness of the protocol for only k honest agents and one dishonest agent, k being the number of roles in the protocol. In [84], they represented the capacities of the intruder (respectively the description of the protocol and the intruder knowledge) by a set of Horn clauses, and the secrecy property by a negative unit Horn clause. They reduced the insecurity problem to the satisfiability problem of the set of clauses constructed from the intruder clauses and the clauses modelising the protocol, intruder knowledge and security goal (secrecy property). Assuming $\mathfrak{C}_{\mathcal{I}}$ to be a class of Horn clauses containing at most one function symbol, and $\mathfrak{C}_{\mathfrak{P}}$ to be a class of Horn clauses containing at most one variable, $\mathfrak{C}_{\mathcal{I}}$ represents the class of intruder clauses, and $\mathfrak{C}_{\mathfrak{P}}$ represents the modelisation of the protocol. As noticed in [84], the protocols with *single blind*

copying, *i.e.* protocols for which, at each step of the protocol, at most one message is blindly copied, fall into the class $\mathfrak{C}_{\mathfrak{P}}$. While not all intruder clauses fall into the class $\mathfrak{C}_{\mathcal{I}}$, for example the intruder clause $I(\{x\}_y^p), I(y^{-1}) \rightarrow I(x)$ representing the asymmetric decryption does not fall in $\mathfrak{C}_{\mathcal{I}}$, H. Comon-Lundh and V. Cortier proposed a solution. As showed in [82], one needs to consider only a finite number of agents. H. Comon-Lundh and V. Cortier proposed to replace each problematic intruder clause, that is each intruder clause that does not fall in $\mathfrak{C}_{\mathcal{I}}$, by a finite set of Horn clauses where the terms dependent from the agents (for example, a secret key of an agent) are replaced by constants (agents parameters). The finiteness of this set is guaranteed by the fact that we need only a finite number of agents. For example, the Horn clause $I(\{x\}_y^p), I(y^{-1}) \rightarrow I(x)$ can be replaced by the Horn clause $I(\{x\}_{pk(a)}^p), I(sk(a)) \rightarrow I(x)$ if we consider a unique agent, *a*. In many cases, for example in the case of *Dolev-Yao* rules, the set of Horn clauses replacing a problematic Horn clause fall in $\mathfrak{C}_{\mathfrak{P}}$. Furthermore, $\mathfrak{C}_{\mathfrak{P}}$ contains all ground Horn clauses, hence clauses modelling the initial intruder knowledge and the secrecy property fall in the class. H. Comon-Lundh and V. Cortier showed that the satisfiability problem of a set of clauses of $\mathfrak{C}_{\mathcal{I}} \cup \mathfrak{C}_{\mathfrak{P}}$ is decidable in 3-EXPTIME, and H. Seidl and K. Verma [187] have shown that satisfiability is in fact DEXPTIME-complete.

6.3.2 Zalinescu's work

In [205], E. Zalinescu extended the result obtained in [84] to a larger class of clauses. Indeed, the prefix property, that is the ability of the intruder to get from any encrypted message the encryption of any of its prefixes, which is represented by the Horn clause $C_{pre} = I(\{< x, y >\}_z) \rightarrow I(\{x\}_z)$ does not fall in neither $\mathfrak{C}_{\mathcal{I}}$ nor $\mathfrak{C}_{\mathfrak{P}}$, and can not be replaced by a set of clauses falling in $\mathfrak{C}_{\mathfrak{P}}$. The same problem arises with the Horn clause representing blind signature $C_{bsig} = I(sig(blind(x, y), z), I(y) \rightarrow I(sig(x, z))$. In order to extend intruder power to clauses like C_{pre} and C_{bsig}, E. Zalinescu defined a class of special clauses, $\mathfrak{C}_S$, defined as follows. Given a strict and total ordering over function symbols, he considered a special function symbol f_0 which is the smaller over the function symbols, and $\mathfrak{C}_S$ is the class of Horn clauses of the form

$$I(f_0(u(g(y_1, \ldots, y_k)), v)), \bigwedge_{i=1}^{p} I(w_i(g(y_1, \ldots, y_k))), \bigwedge_{l=1}^{q} I(y_{i_l}) \rightarrow I(f_0(y_j, z))$$

where $\{j, i_1, \ldots, i_q\} \subseteq \{1, \ldots, k\}$, $p, q \leq 0$, u, w_i are ground contexts, v is a term with $var(v) = \{z\}$, $g \neq f_0$ and $I(f_0(u(g(y_1, \ldots, y_k)), v))$ is greater than any other atom in the clause. He introduced also the notion of *well-behaved term* and *well-behaved clause*. A term is *well-behaved* if for any of its subterms of the form $f_0(u, v)$, the following two implications hold:

if $var(u) \neq \emptyset$ then v is a constant;

if $var(v) \neq \emptyset$ then u is a ground term.

A clause in $\mathfrak{C}_S$ is *well-behaved* if the terms $u,\ v$ and w_i, for all i, are well-behaved. A clause not in $\mathfrak{C}_S$ is *well-behaved* if for any atom $I(w)$ in the clause, w is well-behaved. E. Zalinescu showed that if $\mathcal{I}$, $\mathfrak{P}$, $\mathcal{S}$ are finite set of clauses included respectively in the classes $\mathfrak{C}_\mathcal{I}$, $\mathfrak{C}_\mathfrak{P}$ and $\mathfrak{C}_S$, and if $\mathcal{I} \cup S$ is saturated by a *variant of ordered resolution* with respect to a monotone atom ordering and $\mathfrak{P} \cup S$ is well-behaved then the satisfiability of $\mathcal{I} \cup \mathfrak{P} \cup S$ is decidable.

6.3.3 Delaune, Lin and Lynch work

In [180], *S. Delaune, H. Lin and Ch. Lynch* introduced flexible and rigid clauses. A rigid clause is a clause where variables are only allowed to have one instantiation, and a flexible clause is a clause where variables are allowed as many instantiations as desired. *S. Delaune, H. Lin and Ch. Lynch* assume a finite set of unary predicate symbols $\mathcal{P}$, and a well-founded and total ordering $>$ over $\mathcal{P}$. Furthermore, they considered an *embedding ordering* $\succeq$ over term which is then extended to an atom ordering. They defined intruder clauses as defined in [84], that is with a unique function symbol, and then they extended this definition. To this end, they introduced a special flexible Horn clause of the form

$$I_0(f_0(g(x_1, \ldots, x_p), y_2, \ldots, y_q), I_1(x_{i_1}), \ldots, I_m(x_{i_m}) \rightarrow I_{m+1}(f_0(x_{i_0}, y_2, \ldots, y_q))$$

where $f_0 \neq g$, $x_{i_j} \in \{x_1, \ldots, x_p\}$ for every $j \in \{0, \ldots, m\}$, and $I_j \in \mathcal{P}$ for all j.They defined a protocol clause to be a rigid Horn clause of the form

$$I_1(u_1), \ldots, I_n(u_n) \rightarrow I_0(u_0)$$

where $var(u_0) \subseteq var(\{u_1, \ldots, u_n\})$, and $I_0 \geq I_i$ for all i. Note also that intruder knowledge and the secrecy goal, represented as before, are considered as protocol clauses. They defined intruder theory as follows. Assuming $C_\mathcal{I}$ to be a finite set of intruder clauses, C_S a special clause such that $C_\mathcal{I} \cup C_S$ is saturated by ordered resolution, and the unique predicate symbol in $C_\mathcal{I} \cup C_S$ is I. The intruder theory $\mathfrak{I}_{C_\mathcal{I} \cup C_S}$ is the set of clauses which contains $I_1(u_1), \ldots, I_n(u_n) \rightarrow I_0(u_0)$ if and only if

$I_0, \ldots, I_n \in \mathcal{P}$ and $I_0 \geq I_i$ for all i, and

$I(u_1), \ldots, I(u_n) \rightarrow I(u_0) \in C_\mathcal{I} \cup \{C_S, I(x) \rightarrow I(x)\}$

S. Delaune, H. Lin and Ch. Lynch reduced the insecurity problem to a satisfiability problem defined as follows. The insecurity problem takes as input a finite set $C_\mathfrak{P}$ of protocol clauses, a finite set $C_\mathcal{I}$ of intruder clauses and a special

clause C_S where I is the unique predicate symbols used in $C_{\mathcal{I}} \cup C_S$ and such that $C_{\mathcal{I}} \cup C_S$ is saturated by ordered resolution. This insecurity problem outputs *unsat* if and only if $C_{\mathfrak{P}}\theta \cup \mathfrak{I}_{C_{\mathcal{I}} \cup C_S}$ is unsatisfiable, for a substitution θ grounding for $C_{\mathfrak{P}}$. S. Delaune, H. Lin and Ch. Lynch showed the decidability of this problem.

While in the above sections (Section 6.3.1, 6.3.2 and 6.3.3), the secrecy property is the unique studied security property, other security properties can also be encoded as Horn Clauses. B. Blanchet has encoded authentication [47], and strong secrecy [51].

6.4 Our contribution

In Section 6.3, we have presented some decidable fragments of first order logic, and showed their use in the analysis of security protocols. To this end, intruder rules, protocol description and security property are represented by Horn clauses and the insecurity problem is reduced to the satisfiability problem of a fragment of first order logic.

In this section, we present our model to analyse security protocols using Horn clauses. As opposite to the models represented in Section 6.3, we do not represent protocol description by Horn clauses. As showed in Chapter 2, one line of research reduces the insecurity problem of cryptographic protocols to an intruder reachability problem, also called $\mathcal{I}$-reachability problem, and in this chapter we reduce the $\mathcal{I}$-reachability problem to the entailment problem (or satisfiability problem) in first order logic. We remark that Horn clauses and a unique unary predicate symbol I are sufficient to analyse security protocols. The predicate symbol I represents the knowledge of the intruder: $I(m)$ means that the intruder knows the term (or message) m. Thus a clause $I(u_1), \ldots, I(u_n) \rightarrow I(v)$ should be read as "if the intruder knows some messages of the form $u_1, \ldots, u_n$, then he knows a message of the form v". For example, let us consider the clause $I(x), I(y) \rightarrow I(< x, y >)$, and assume that the intruder knows the messages a, b, then, due to the clause given above, the intruder knows also the message $< a, b >$. There is thus a natural correspondence between constraint systems (respectively solving of constraint systems) and Horn clauses (respectively entailment problems).

6.4.1 Model for cryptographic protocols

Intruder clauses

Let $\mathcal{I}$ be an intruder and let $\mathcal{L}_{\mathcal{I}}$ be its deduction system ($\mathcal{L}_{\mathcal{I}}$ represents the intruder capacities). $\mathcal{L}_{\mathcal{I}}$ is a set of rules of the form $u_1, \ldots, u_n \rightarrow v$ where $u_1, \ldots, v_n, v$ are terms in the given algebra. The set of clauses associated with

$\mathcal{I}$ is

$$C_{\mathcal{I}} \stackrel{def}{=} \{I(u_1), \ldots, I(u_n) \rightarrow I(v) \; such \; that \; u_1, \ldots, u_n \rightarrow v \in \mathcal{L}_{\mathcal{I}}\}$$

Example 23 *The Dolev-Yao rules presented in Chapter 2, in Example 8, more exactly the rules without explicit destructors are represented by the following set of Horn clauses:*

$$C_{DY} = \left\{ \begin{array}{l} I(x), I(y) \rightarrow I(< x, y >) \\ I(x), I(y) \rightarrow I(\{x\}_y^s) \\ I(x), I(y) \rightarrow I(\{x\}_y^p) \\ I(< x, y >) \rightarrow x \\ I(< x, y >) \rightarrow I(y) \\ I(\{x\}_y^s), I(y) \rightarrow I(x) \\ I(\{x\}_y^p), I(y^{-1}) \rightarrow I(x) \\ I(\{x\}_{y^{-1}}^p), I(y) \rightarrow I(x) \end{array} \right.$$

Given a set of terms E, we define $C_E \stackrel{def}{=} \{I(u) \; such \; that \; u \in E\}$.

Lemma 56 *Let $C_{\mathcal{I}}$ be the set of clauses associated with the intruder $\mathcal{I}$, E a set of ground terms and m a ground term. If $m \in \bar{E}^{\mathcal{I}}$ then $C_E \cup C_{\mathcal{I}} \models I(m)$.*

PROOF.

Let E and m be respectively a set of ground terms and a ground term, and assume that $m \in \bar{E}^{\mathcal{I}}$. This implies that there is a derivation $\mathfrak{d}$ starting from E of goal m, $\mathfrak{d} = E \rightarrow_{\mathcal{I}} E_1 \rightarrow_{\mathcal{I}} \ldots \rightarrow_{\mathcal{I}} E_{n-1}, m$, where $E_i = E_{i-1} \cup m_i$, $E_0 = E$ and $m_n = m$. We reason by induction on the $lenghth$ of the derivation $\mathfrak{d}$.

- $lenghth(\mathfrak{d}) = 0$: then $m \in E$ and thus $I(m) \in C_E$ which implies that $C_E \models I(m)$ and then $C_E \cup C_{\mathcal{I}} \models I(m)$.
- $lenghth(\mathfrak{d}) = 1$: let $u_1, \ldots, u_n \rightarrow v$ be the intruder deduction rule applied on E in the derivation $\mathfrak{d}$, we have $\{u_i\sigma\}_{1 \leq i \leq n} \subseteq E$ and $m = v\sigma$ for a ground substitution σ. Hence, for all $i \in \{1, \ldots, n\}$, $I(u_i)\sigma \in C_E$ which implies that $C_E \models I(u_i)\sigma$. Consider an arbitrary model $\mathfrak{Int}$ of $C_E \cup C_{\mathcal{I}}$, $\mathfrak{Int}$ satisfies all $I(u_i)\sigma$. Let us prove that $I(m)$ which is equal to $I(v)\sigma$ is true in $\mathfrak{Int}$ by contradiction. If $I(v)\sigma$ is false in $\mathfrak{Int}$ then the clause $I(u_1)\sigma, \ldots, I(u_n)\sigma \rightarrow I(v)\sigma$ is not satisfied by $\mathfrak{Int}$, and hence neither the clause $I(u_1), \ldots, I(u_n) \rightarrow I(v)$ which belong to $C_{\mathcal{I}}$. But this contradicts $\mathfrak{Int}$ being a model of $C_E \cup C_{\mathcal{I}}$. Thus, $I(m) = I(v)\sigma$ is true in $\mathfrak{Int}$. We conclude that $C_E \cup C_{\mathcal{I}} \models I(m)$.
- Assume that $C_E \cup C_{\mathcal{I}} \models I(m)$ for any length k of the derivation $\mathfrak{d}$, $k \leq n$ and $n \geq 0$, and let us prove it for $lenghth(\mathfrak{d}) = n + 1$. Assume that $u_1, \ldots, u_n \rightarrow v$ be the last applied rule in the derivation $\mathfrak{d}$. Then all $u_i\sigma$ are in E_n and $v\sigma = m$ for a ground substitution σ. By induction we have $C_E \cup C_{\mathcal{I}} \models I(u_i)\sigma$ for all i. By the same reasoning as above, we deduce that $C_E \cup C_{\mathcal{I}} \models I(v)\sigma$, and thus $C_E \cup C_{\mathcal{I}} \models I(m)$.

Again, by induction, we conclude that $C_E \cup C_{\mathcal{I}} \models I(m)$ for any length of the derivation $\mathfrak{d}$, and then, we conclude the proof. ■

Insecurity problem of cryptographic protocols

We are concerned here with secrecy property of cryptographic protocols. We show in this Section how secrecy property of cryptographic protocols can be encoded as Horn clauses. As we see in the Section 6.4.1, the behaviour of the intruder represented in Chapter 2 by a set of deduction rules can be modeled by a set of Horn clauses. The initial knowledge of the intruder IK usually given by a set of ground terms can be modeled by a set of clauses C_{IK}. We show in Chapter 2 that the insecurity problem of a protocol is reduced to a reachability problem, and thus by constructing a constraint system for each execution of the protocol and reducing the insecurity problem of the execution to the satisfiability problem of its correspondant constraint system, and hence to a reachability problem. Now, we will show how to reduce the satisfiability problem of a ground constraint system to a ground entailment problem in the first order logic. Let the ground constraint system $\mathfrak{C} = (E_1 \rhd t_1, \ldots, E_n \rhd t_n)$, for each constraint $E_i \rhd t_i \in \mathfrak{C}$, we associate the following Horn clause $C_{E_i} \rightarrow I(t_i)$. For example, the clause $I(a), I(b) \rightarrow I(< a, b >)$ corresponds to the constraint $\{a, b\} \rhd < a, b >$. The constraint system $\mathfrak{C}$ will then be associated to the set of ground Horn clauses $C_{\mathfrak{C}} = \{C_{E_1} \rightarrow I(t_1), \ldots, C_{E_n} \rightarrow I(t_n)\}$, and we have that $\mathfrak{C}$ is satisfiable if and only if $C_{\mathfrak{C}}$ is entailed from $C_{\mathcal{I}}$. Thus, an execution $\mathfrak{exec}$ does not preserve the secrecy of a message s if and only if the set of Horn clauses $C_{\mathfrak{C}'_{\mathfrak{exec}}}$ is entailed from the set of Horn clauses $C_{\mathcal{I}}$, where $\mathfrak{C}'_{\mathfrak{exec}}$ is the extended ground constraint system representing the execution $\mathfrak{exec}$. The construction of the extended constraint system $\mathfrak{C}'_{\mathfrak{exec}}$ from an execution $\mathfrak{exec}$ is given in Chapter 2.
A set S_1 of Horn clauses is entailed from a set S_2 of Horn clauses if each clause C is S_1 is entailed from S_2. For example, $S \models \{C_1, C_2\}$ if $S \models C_1$ and $S \models C_2$, S is a set of clauses and C_1, C_2 are two clauses. Thus, assuming that the protocol runs correctly (*i.e.* runs in presence of a passive intruder), each protocol execution is associated to a ground constraint system, and then, the insecurity problem of a protocol execution and hence of a cryptographic protocol under a bounded number of sessions is reduced to the ground entailment problem for $C_{\mathcal{I}}$.

Example 24 ***Needham-Schroeder public key protocol.***

Presentation of the protocol *We consider the Needham-Schroeder symmetric key protocol [163] as an example. This protocol intends to permit Alice to establish a shared key (session key) with Bob and to obtain mutual conviction of the possession of the key by each other. The session key is created by a trusted server which shares a secret key*

with each partie. The protocol can be described as follows:

$$\mathfrak{P}_{NS}: \begin{cases} 1\ A \Rightarrow S: & A, B, N_A \\ 2\ S \Rightarrow A: & \{N_A, B, K_{AB}, \{K_{AB}, A\}^s_{K_{BS}}\}^s_{K_{AS}} \\ 3\ A \Rightarrow B: & \{K_{AB}, A\}^s_{K_{BS}} \\ 4\ B \Rightarrow A: & \{N_B\}^s_{K_{AB}} \\ 5\ A \Rightarrow B: & \{N_B - 1\}^s_{K_{AB}} \end{cases}$$

N_A (respectively N_B) represents the nonce freshly created by A (respectively B), K_{AS} (respectively K_{BS}) represents the secret key shared between A (respectively B) and the trusted server, and K_{AB} the session key shared between A and B.

The functions symbols $\{-\}^s_-, \{-\}^{-1s}_-$ used in the specification of protocol given above denotes respectively the symmetric encryption and symmetric decryption. While symmetric encryption has been denoted before by the symbol enc^s and symmetric decryption by dec^s, we use here the notation $\{-\}^s_-, \{-\}^{-1s}_-$ with the aim of being more readable.

In this protocol, we have three roles, the trusted server, Alice and Bob. The role server is given by the following rule:
$S1:$

$$v_1 \Rightarrow \{\pi_2(\pi_2(v_1)), \pi_1(\pi_2(v_1)), K_{\pi_1(v_1)\pi_1(\pi_2(v_1))}, \{K_{\pi_1(v_1)\pi_1(\pi_2(v_1))}, \pi_1(v_1)\}^s_{K_{\pi_1(\pi_2(v_1))S}}\}^s_{K_{\pi_1(v_1)S}};$$
$$v_1 \stackrel{?}{=} X_1, Y_1, Z_1$$

The role Alice is given by the following set of rules:

$$\begin{aligned} A1:&\ \emptyset \Rightarrow A, X_2, N_A; \emptyset \\ A2:&\ v_2 \Rightarrow \pi_2(\pi_2(\pi_2(\{v_2\}^{-1s}_{K_{AS}})));\ v_2 \stackrel{?}{=} \{N_A, X_2, Y_2, Z_2\}^s_{K_{AS}} \\ A3:&\ v_3 \Rightarrow \{\{v_3\}^{-1s}_{\pi_1(\pi_2(\pi_2(\{v_2\}^{-1s}_{K_{AS}})))} - 1\}^s_{\pi_1(\pi_2(\pi_2(\{v_2\}^{-1s}_{K_{AS}})))}; \\ &\ v_3 \stackrel{?}{=} \{X_2'\}^s_{\pi_1(\pi_2(\pi_2(\{v_2\}^{-1s}_{K_{AS}})))} \end{aligned}$$

And the role Bob is given by the following set of rules:

$$\begin{aligned} B1:&\ v_4 \Rightarrow \{N_B\}^s_{\pi_1(\{v_4\}^{-1s}_{K_{BS}})}; v_4 \stackrel{?}{=} \{X_3, Y_3\}^s_{K_{BS}} \\ B2:&\ v_5 \Rightarrow \emptyset; v_5 \stackrel{?}{=} \{N_B - 1\}^s_{\pi_1(\{v_4\}^{-1s}_{K_{BS}})} \end{aligned}$$

Attack on the protocol *Many attacks have been discovered on this protocol, D. Denning and G. Sacco [99] considered that communication keys may be compromised, and showed that the protocol is vulnerable to reply attack. Here we concentrate on the key exchange goal rather than on the authentication of the two parties. The key exchange goal can be expressed by the secrecy of the nonce N_B. Intuitively, if N_B remains secret than the key K_{AB} has also been kept secret. We will present the attack discovered*

Figure 6.1 Attack on the Needham-Schroeder protocol

$(1).1\ \ A \Rightarrow S:\ \ A, B, N_A$
$(1).2\ \ S \Rightarrow A:\ \ \{N_A, B, K_{AB}, \{K_{AB}, A\}^s_{K_{BS}}\}_{K_{AS}}$
$(2).3\ \ I(B) \Rightarrow A:\ \ \{N_A, B\}_{K_{AS}}$
$(2).4\ \ A \Rightarrow I(B):\ \ \{N'_A\}_{N_A}$

by O. Pereira and J.-J. Quisquater [169], this attack described in Figure 6.1 allows, as we will see below, the intruder to break the secrecy of N_B. This attack is possible if the encryption scheme used in the implementation of the protocol is used in the cipher-block-chaining (CBC) mode. In the case of CBC encryption, the intruder may be able to get from any encrypted message the encryption of any of its prefixes, and that without knowing the encryption key: from the message $\{< x, y >\}^s_z$, the intruder can deduce the message $\{x\}^s_z$. This property, called prefix property, is encoded by the deduction rule

$$\{< x, y >\}^s_z \rightarrow \{x\}^s_z$$

and hence by the clause

$$C_{pre} = I(\{< x, y >\}^s_z) \rightarrow I(\{x\}^s_z)$$

In a first session (1), the intruder can listen to the message $\{N_A, B, K_{AB}, \{K_{AB}, A\}^s_{K_{BS}}\}_{K_{AS}}$ and then, using the prefix property, computes the message $\{N_A, B\}_{K_{AS}}$. This message is of the form Alice might expect to receive as third message of a latter session of the protocol where Bob is considered to play initiator's role. Then once the message $\{N_A, B\}^s_{K_{AS}}$ computed, the intruder starts a another session of the protocol by sending to Alice the message $\{N_A, B\}_{K_{AS}}$. Alice thinks that Bob has started a session (2) with her and thus, Alice can be fooled into accepting the publicly known N_A as a secret key shared with Bob. Let us consider an instance of the protocol $\mathfrak{P}_{NS}$ where Alice instantiates once the role A and once the role B, Bob instantiates only once the role B and the agent s instantiates only once the role S, K_{bs} denotes the secret key shared between Bob and s, and K_{as} denotes the secret key shared between Alice and s.

The attack described above is a possible execution of this instance of $\mathfrak{P}_{NS}$. This execution is given by the set of rules $\{A_1(Alice), S_1(s), B_1(Alice)\}$. At the end of this execution, the intruder breaks the secrecy of the nonce n'_a. This execution is then given by the following set of rules:

$$\emptyset \Rightarrow Alice, Bob, n_a$$
$$v_1 \Rightarrow \{n_a, Bob, K_{ab}, \{K_{ab}, Alice\}^s_{K_{bs}}\}_{K_{as}}; v_1 \stackrel{?}{=} Alice, Bob, n_a$$
$$v_2 \Rightarrow \{n'_a\}_{n_a}; v_2 \stackrel{?}{=} \{n_a, Bob\}_{K_{as}}$$

We associate to this execution the ground constraint system $\mathfrak{C}$, $\mathfrak{C} = (E_1 \rhd Alice, E_1 \rhd Bob, E_1 \rhd n_a, E_1 \cup \{n_a, Bob, K_{ab}, \{K_{ab}, Alice\}^s_{K_{bs}}\}^s_{K_{as}} \rhd \{n_a, Bob\}^s_{K_{as}}, E_1 \cup$

$\{n_a, Bob, K_{ab}, \{K_{ab}, Alice\}^s_{K_{bs}}\}^s_{K_{as}} \cup \{n'_a\}^s_{n_a} \rhd n'_a)$ *and* E_1 *is the initial knowledge of the intruder,* $E_1 = \{Alice, Bob, s, K_{is}, n_a\}$.
We associate to this ground constraint system the following set of clauses $C_{\mathfrak{C}}$*:*

$$\begin{array}{l}
\mathcal{C}_{E_1} \rightarrow I(Alice) \\
\mathcal{C}_{E_1} \rightarrow I(Bob) \\
\mathcal{C}_{E_1} \rightarrow I(n_a) \\
\mathcal{C}_{E_1} \cup I(\{n_a, Bob, K_{ab}, \{K_{ab}, Alice\}_{K_{bs}}\}^s_{K_{as}}) \rightarrow I(\{n_a, Bob\}^s_{K_{as}}) \\
\mathcal{C}_{E_1} \cup I(\{n_a, Bob, K_{ab}, \{K_{ab}, Alice\}^s_{K_{bs}}\}^s_{K_{as}}) \cup I(\{n'_a\}^s_{n_a}) \rightarrow I(n'_a)
\end{array}$$

where $C_{E_1} = \{I(Alice), I(Bob), I(s), I(K_{is}), I(n_a)\}$. *It is easy to see that* $C_{\mathfrak{C}}$ *is entailed from* $C_{\mathcal{I}}$ *where* $C_{\mathcal{I}} = C_{DV} \cup C_{pre}$.

6.5 A decidability result

In this section, we show the decidability of the ground entailment problem for a new fragment of first order logic. To get this result, we consider a refinement of resolution, the *selected resolution*, that we present in Section 6.5.1, and for which we show the completeness and soundness. We prove our decidability result in Section 6.5.2.

6.5.1 Selected resolution

In Section 6.1.2, we have presented the resolution and the standard ordering refinement of resolution (the ordered resolution). In this section, we present another refinement of resolution, the *selected resolution* (or *resolution with selection*) and then show its completeness and soundness.The selected resolution is the resolution considered in the remainder of this chapter.

A *selection function* is a function that will be applied to each clause and selects (or marks) a possibly empty set of its atoms. A selection function is said to be *valid* if for each clause, either all maximal atoms are selected or, at least one atom appearing in the antecedent of the clause is selected [146].

The selection resolution has been widely studied in the literature [134, 133, 137, 146, 164]. In [134], R. Kowalski and D. Kuehner introduced the linear resolution with selection function, also called SL-resolution. This resolution is closely related to D. W. Loveland's model elimination system [143]. In [133], R. Kowalski introduced another variation of linear resolution with selection, which considers only Horn clauses and not general clauses as in [134]. In [146], Ch. Lynch introduced the resolution with selection as we employ it. He assumed a selection function which, for each clause, selects all maximal atoms or, at least one atom appearing in the antecedent of the clause. He assumed also that the resolution includes the following deletion rules: *tautology deletion* and

subsumption; *tautology deletion* deletes any clause of the form $\Gamma, A \to \Delta, A$, and *subsumption* deletes any clause C' such that there exists another clause C with $C\sigma \subseteq C'$ for a substitution σ. In [164], H. De Nivelle defined another variation of the resolution with selection, called *L-ordered resolution*. He assumed a selection function that, for each clause, selects the maximal atoms with respect to an atom ordering $\prec$ that verifies the following property: for every atoms A, B, if $A \preceq B$ then $A\theta \preceq B\theta$ for every substitution θ.

In this chapter, we make use of the resolution with selection introduced in [146], and from now on, we consider only a subset of the valid selection functions, namely the functions that select maximal atoms in clauses with respect to an atom ordering. We remark that the selection functions we consider are less general than in [146], and more general than in [164] since we consider an arbitrary atom ordering.

Selected resolution is described by the following two inference rules:

$$Selected\ \ resolution \quad \frac{\Gamma \to \Delta, A \qquad A', \Gamma' \to \Delta'}{(\Gamma, \Gamma' \to \Delta, \Delta')\alpha}$$

where α is the most general unifier of A and A',
and A, A' are selected in their respective premises.

$$Selected\ \ factoring \quad \frac{\Gamma \to \Delta, A, A'}{(\Gamma \to \Delta, A)\alpha}$$

where α is the most general unifier of A and A',
and A or A' is selected in the premise.

Selected resolution (respectively selected factoring) inference rule requires that all atoms in the resolvent of two ground premises (respectively in the factor of a ground premise) are strictly smaller than the resolved (respectively factored) atom. We define a notion of redundancy that identifies clauses and inferences that are not needed for establishing refutational completeness of selected resolution. To this end, we define next the relations $\to_{\mathcal{R}_S}$ and $\to_{\mathcal{R}_S^g}$ over atoms, and the notation $\mathcal{A}\downarrow_S$ where $\mathcal{A}$ is a set of atoms and S a set of clauses.

Definition 55 *Let S be a set of clauses. We define the relation $\to_{\mathcal{R}_S}$ as follows: we have $A \to_{\mathcal{R}_S} B$ if there exists a clause C in S and A, B two atoms in C such that $A \succ_a B$; and we define the relation $\to_{\mathcal{R}_S^g}$ as follows: we have $A \to_{\mathcal{R}_S^g} B$ if there exists two atoms A^s, B^s such that $A^s \to_{\mathcal{R}_S} B^s$, $A^s\sigma = A$ and $B^s\sigma = B$ for a substitution σ grounding $A^s,\ B^s$.*

Let the rewriting system $\mathcal{R}_S$ (respectively $\mathcal{R}_S^g$) be the set of rewriting rules $\to_{\mathcal{R}_S}$ (respectively $\to_{\mathcal{R}_S^g}$). By definition, it is easy to see that rules in $\mathcal{R}_S^g$ are ground, and that $\mathcal{R}_S^g$ is set of ground instances of rewriting rules in $\mathcal{R}_S$. We also

remark that if $S \subseteq S'$, then $\mathcal{R}_S \subseteq \mathcal{R}_S$ and $\mathcal{R}^g_S \subseteq \mathcal{R}^g_S$. We denote by $\rightarrow^*_{\mathcal{R}_S}$ (respectively $\rightarrow^*_{\mathcal{R}^g_S}$) the transitive cloture of $\rightarrow_{\mathcal{R}_S}$ (respectively $\rightarrow_{\mathcal{R}^g_S}$), more formally, if A and B are two ground atoms, we have $A \rightarrow^*_{\mathcal{R}_S} B$ (respectively $A \rightarrow^*_{\mathcal{R}^g_S} B$) if and only if $A \rightarrow_{\mathcal{R}_S} A_1 \rightarrow_{\mathcal{R}_S} \ldots \rightarrow_{\mathcal{R}_S} B$ (respectively $A \rightarrow_{\mathcal{R}^g_S} A_1 \rightarrow_{\mathcal{R}^g_S} \ldots \rightarrow_{\mathcal{R}^g_S} B$).

Definition 56 *Let S be a set of clauses, and $\mathcal{A}$ be a set of atoms. We define the set $\mathcal{A}\downarrow_S$ as follows: $\mathcal{A}\downarrow_S = \{A \mid \exists B \in \mathcal{A}$ with $B \rightarrow^*_{\mathcal{R}_S} A\}$.*

Definition 57 *(redundancy)*

- *A ground clause C is redundant in a set of clauses S with respect to $\succ_a$ if it can be derived, by binary resolution, from a finite set S' of ground clauses instances of clauses in S such that $\mu(S') \subseteq \mu(C)\downarrow_S$.*
- *We call an inference by selected resolution from ground clauses C', C'' redundant in the set of clauses S if the conclusion, C, can be derived, by binary resolution, from a finite set S' of ground clauses instances of clauses in S such that $\mu(S') \subseteq \mu(\{C_1, C_2\})\downarrow_S \setminus\{A\}$ where A is the resolved atom.*
- *We call an inference by selected factoring from a ground clause C' redundant in the set of clauses S if the conclusion, C, can be deduced, by binary resolution, from a finite set S' of ground clauses instances of clauses in S such that $\mu(S') \subseteq \mu(C')\downarrow_S \setminus\{A\}$ with A the factored atom.*
- *A non-ground inference is called redundant in S if all its ground instances are redundant.*
- *Finally, we call a set of clauses S saturated up to redundancy under selected resolution with respect to $\succ_a$, if any inference by selected resolution or factoring from premises in S is redundant in S.*

We remark that another notion of redundancy has been introduced in [28], that notion is based on the multiset ordering over clauses.

Remark Let us consider the set $S = \{q(x) \rightarrow q(f(x))\}$ of clauses, and the clause $C = q(x) \rightarrow q(f(f(x)))$. Following our definition of redundancy, C is redundant in S. But, C is not redundant in S following the definition of redundancy given in [28].

We also remark that our definition of *saturation* is different that the definition of *saturation* given in [146]: while we define the saturation with respect to a notion of redundancy, Ch. Lynch [146] defined the saturation with respect to the tautology deletion and subsumption rules.

We conjecture that under a slight modifications, our definition of saturation may include the *forward subsumption rule*: in fact, if (1) we order the clauses in a set S of clauses as follows:

we associate an index to each clause in S, then during the saturation of S under selected resolution, to each obtained new clause we associate an index which is bigger (with respect to the classical "greater than" order in the natural numbers) than the previous indexes

and if (2) we associate to each proof the multiset of indexes of the clauses labelling its leaves, then for any set of clauses S and any ground clause C, if there is a proof of $S \models C$ which uses a clause C_2 in S that can be subsumed by another clause C_1 in S with the index of C_1 is smaller than the index of C_2, then there is another proof of $S \models C$ which does not use the clause C_2 and which is smaller than the first proof with respect to the multiset of indexes associated to the proofs. Thus, we conjecture that under the modifications given above, we can introduce the *forward subsumption* in the saturation procedure.

We remark that if C is the conclusion of an inference by selected factoring from a ground clause C', this inference should be given as follows:

$$\frac{\Gamma \rightarrow \Delta, A, A}{\Gamma \rightarrow \Delta, A}$$

where $C' = \Gamma \rightarrow \Delta, A, A$ and $C = \Gamma \rightarrow \Delta, A$. As the definition of clauses prohibits the existence of the clause $\Gamma \rightarrow \Delta, A, A$, this clause is replaced directly by the clause $\Gamma \rightarrow \Delta, A$ and hence, the inference by selected factoring from a ground clause C' will be given as follows:

$$\frac{C'}{C'}$$

We conclude that such inferences can be ignored in the construction of (ground) proofs.

As for ordered resolution, we do not have that every ground instance of an inference by selected resolution is an inference by selected resolution. Let us consider the same example considered for ordered resolution, let Inf be the following inference by selected resolution

$$\frac{I(y) \rightarrow I'(x) \qquad I'(x'), I(y') \rightarrow I''(y')}{I(y), I(y') \rightarrow I''(y')}$$

The most general unifier of $I'(x), I'(x')$ is $\alpha = \{x' \mapsto x\}$. By definition of selected resolution inference, we have that $I(y) \not\geq I'(x)$, $I(y') \not> I'(x')$, and $I''(y') \not> I'(x)$. Let the ground substitution σ be such that $\sigma = \alpha\alpha'$, $I(y\sigma) > I(y'\sigma) > I''(y'\sigma) > I'(x\sigma)$ and $I'(x\sigma) = I'(x'\sigma)$. $Inf\sigma$ is a ground instance of Inf, and it is easy to see that $Inf\sigma$ is not an inference by selected resolution.

Soundness and Completeness. We show that selected resolution is sound and complete. We recall the definitions of soundness and completeness. The soundness means that if a set S of clauses is satisfiable then the empty clause can not

be derived from S, and the completeness means that if a set S of clauses is unsatisfiable then the empty clause can be derived from S. Furthermore, if a set S of clauses is saturated, we have: $\emptyset \in S$ if and only if S is unsatisfiable. We remark that the soundness and completeness of the selected resolution are also given in [146]. We prove these results in the following lemmas.

Lemma 57 *Each ordered resolution inference is a selected resolution inference which is, in its turn, a resolution inference.*

PROOF.

Let C and C' be two clauses, $C = \Gamma \rightarrow A, \Delta$ and $C' = A', \Gamma' \rightarrow \Delta'$. Assume that ordered resolution can be applied on C and C', this implies that A, A' are unifiable and let σ be their most general unifier. In addition, $A\sigma$ is strictly maximal with respect to $\Gamma\sigma, \Delta\sigma$ and $A'\sigma$ are maximal with respect to $\Gamma'\sigma, \Delta'\sigma$ for the ordering $\succ_a$.This implies that A, A' are maximals for the ordering $\succ_a$ in their respectives clauses and then, selected resolution can be applied on C and C'. In fact, if A is not maximal in C, there is an atom $B \in C$ such that $B \succ_a A$ and then, by monotonicity of $\succ_a$, $B\sigma \succ_a A\sigma$ which contradicts the maximality of $A\sigma$ with respect to $\Gamma\sigma, \Delta\sigma$. Now, let us prove the other part of the lemma. Assume that selected resolution can be applied on the clauses C and C', this implies, among other, that A and A' are unifiable, and hence, the resolution can be applied on C and C'. ■

Lemma 58 *Each ordered factoring inference is a selected factoring inference which is, in its turn, a factoring inference.*

PROOF.

Let $C = \Gamma \rightarrow \Delta, A, A'$ be a clause, and assume that ordered factoring can be applied on it. This implies that A, A' are unifiable and let σ be their most general unifier. In addition, $A\sigma$ is strictly maximal with respect to $\Gamma\sigma$ and maximal with respect to $\Delta\sigma$ for the ordering $\succ_a$.This implies that $A,\ A'$ are maximals for the ordering $\succ_a$ in the C and then, selected factoring can be applied on C.In fact, if A is not maximal in C, there is an atom $B \in C$ such that $B \succ_a A$ and then $B\sigma \succ_a A\sigma$ which contradicts the maximality of $A\sigma$ with respect to $\Gamma\sigma, \Delta\sigma$. Now, let us prove the other part of the lemma. Assume that selected factoring can be applied on the clause C, this implies, among other, that A and A' are unifiable, and hence, the factoring can be applied on C and C'. ■

Lemma 59 *If a set S of clauses is unsatisfiable then the empty clause can be deduced from S by applying selected resolution and selected factoring inferences.*

PROOF.

Let S be a set of unsatisfiable clauses. By the completeness of the ordered

resolution, we deduce that the empty clause can be derived from S by applying ordered resolution and ordered factoring inferences, and as each ordered resolution (respectively ordered factoring) inference is a selected resolution (respectively selected factoring) inference, by lemma 57, we conclude the proof. ■

Lemma 60 *If a set S of clauses is satisfiable then the empty clause can not be derived from S by applying selected resolution and selected factoring inferences.*

PROOF.

Let S be a set of satisfiable clauses and let us prove the lemma by contradiction. Assume that the empty clause can be derived from S by applying selected resolution and selected factoring inferences. This implies that the empty clause can be derived from S by applying resolution and factoring inferences, by lemma 57 which is impossible due to the soundness of resolution and the unsatisfiability of S. ■

6.5.2 A decidability result

In the rest of this chapter, we consider a complete simplification ordering $\succ_t$ over terms and a complete ordering $\succ_a$ over atoms compatible with $\succ_t$. We recall the definition of a complete ordering. An ordering over a set of elements is said to be complete if it is total over ground elements of this set. A *complete simplification ordering* over terms is any ordering over terms which is well-founded, monotone, stable, subterm, and total over ground terms. We call an atom ordering $\succ_a$ *compatible* with the term ordering $\succ_t$ if it satisfies the following condition: $p(t_1, \ldots, t_k) \succ_a q(s_1, \ldots, s_n)$ if and only if for any j with $1 \leq j \leq n$ there exists an i with $1 \leq i \leq k$ such that $t_i \succ_t s_j$.

In the following lemma, we show some properties of the atom ordering $\succ_a$.

Lemma 61 *The ordering $\succ_a$ is:*
(1) *well-founded,*
(2) *monotone,*
(3) *and $B \prec_a A$ implies $Var(B) \subseteq Var(A)$.*

PROOF.

We recall that the ordering $\succ_a$ is compatible with the complete simplification ordering $\succ_t$ and $\succ_a$ is total on ground atoms.

1. Let us prove that $\succ_a$ is well-founded by contradiction. Assume that $\succ_a$ is not well-founded. By definition, there is an infinite descending chain of atoms $A_0 \succ_a A_1 \succ_a \ldots$. By the compatibility of $\succ_a$ with $\succ_t$, we deduce that there is an infinite descending chain of terms $t_0 \succ_t t_1 \succ_t \ldots$ where t_i is a term

of the atom A_i. That implies $\succ_t$ is not well-founded which contradicts the fact that $\succ_t$ is a complete simplification ordering.

2. Let $A,\ B$ be two atoms such that $B \prec_a A$. Suppose that $A = I(t_1, \ldots, t_n)$ and $B = I'(s_1, \ldots, s_m)$. By the compatibility of $\succ_a$ with $\succ_t$, for all $i \in \{1, \ldots, m\}$, there is $j \in \{1, \ldots, n\}$ such that $s_i \prec_t t_j$, and then, by monotonicity of $\succ_t$, $s_i\sigma \prec_t t_j\sigma$ for any substitution σ. Again by the compatibility of $\succ_a$ with $\succ_t$, we deduce that $B\sigma \prec_a A\sigma$ for any σ and then the monotonicity of $\succ_a$.

3. Let $A,\ B$ be two atoms such that $B \prec_a A$. By the compatibility of $\succ_a$ with $\succ_t$, each term in B is smaller than a term in A for the ordering $\succ_t$ and as $Var(t) \subseteq Var(t')$ for all terms t, t' and $t \prec_t t'$ (Lemma 6), we deduce that $Var(B) \subseteq Var(A)$.

■

Lemma 62 *If C is a ground atom there is no infinite chain of atoms $C \rightarrow_{\mathcal{R}_S^g} C_1 \rightarrow_{\mathcal{R}_S^g} \ldots$.*

PROOF.

Let us prove the lemma by contradiction. Let C be a ground atom and suppose that there is an infinite chain of atoms $C = C_0 \rightarrow_{\mathcal{R}_S^g} C_1 \rightarrow_{\mathcal{R}_S^g} \ldots$. For every $i \geq 0$, we have $C_i \rightarrow_{\mathcal{R}_S^g} C_{i+1} \in \mathcal{R}_S^g$ and then, by definition of $\rightarrow_{\mathcal{R}_S^g}$ and by lemma 61, $C_i \succ_a C_{i+1}$. From the infinite chain starting from C using the relation $\rightarrow_{\mathcal{R}_S^g}$, we deduce that there is an infinite chain $C \succ_a C_1 \succ_a \ldots$ starting from C and using the ordering $\succ_a$ which contradicts the well-foundness of $\succ_a$ and thus, we conclude the proof. ■

Given a ground atom A, the set $Succ(A)$ of ground atoms is defined as follows: $B \in Succ(A)$ if and only if $A \rightarrow_{\mathcal{R}_S^g} B$.

Lemma 63 *If S is a finite and saturated set of clauses then for every ground atom C, $Succ(C)$ is finite.*

PROOF.

Let S be a finite and saturated set of clauses and let $\mathcal{R}_S$ and $\mathcal{R}_S^g$ be the sets of rewriting rules constructed as defined before (Section 6.5.1). Let $A^s \rightarrow B^s$ is a rewriting rule in $\mathcal{R}_S$ and let σ be a substitution grounding A^s and B^s. We recall that $\mathcal{R}_S^g$ is the set of ground instances of rewriting rules in $\mathcal{R}_S$, and hence, $A^s\sigma \rightarrow B^s\sigma \in \mathcal{R}_S^g$. Assume that $A^s\sigma \rightarrow B^s\sigma$ be a rewriting rule in $\mathcal{R}_S^g$ that can be applied on C. As C and $A^s\sigma$ are ground, we have $C = A^s\sigma$. This implies that C and A^s are unifiable. As C is ground and by the unification, σ is the unique unifier of C and A^s. We deduce that, for each rule in $\mathcal{R}_S$, at most one ground instance will be applied on C and by the finiteness of $\mathcal{R}_S$, obtained due to the finiteness of S, we conclude the proof. ■

Lemma 64 *If S is a finite and saturated set of clauses then for every ground atom C, $C\downarrow_S$ is finite.*

PROOF.

Let C be a ground atom and let $\mathcal{T}$ be the tree constructed as follows: the root of $\mathcal{T}$ is labelled by the atom C, if n is a node in $\mathcal{T}$ labelled by the atom D, each atom D' verifying $D \rightarrow_{\mathcal{R}_S^g} D'$ will be added as direct successor of the node labelled by D. We remark that the set of atoms labelling the nodes of $\mathcal{T}$ is equal to the set $C\downarrow_S$. Let us prove the lemma by contradiction. Suppose that $C\downarrow_S$ is infinite, this implies that the tree $\mathcal{T}$ is infinite. As, by lemma 63, each node in $\mathcal{T}$ has a finite number of direct successors, we deduce, by $König's$ lemma, that there is an infinite chain of atoms $C \rightarrow_{\mathcal{R}_S^g} C_1 \rightarrow_{\mathcal{R}_S^g} \ldots$ in the tree $\mathcal{T}$ which contradicts the non existence of infinite chain starting from a ground atom using the relation $\rightarrow_{\mathcal{R}_S^g}$ (lemma 62). We conclude that $\mathcal{T}$ is finite and hence the set $C\downarrow_S$ is finite also. ■

Let S be a set of saturated clauses, C be a ground clause, and assume that C is entailed from S. Let π be a proof of $S \models C$. We define a partial order $\succ_\pi$ over atoms in the proof π as follows: for each clause C' in S, for each ground instance $C'\sigma$ of C' labelling a leave of the proof π, we have $B^s\sigma \gg A^s\sigma$ if A^s, B^s are atoms in C' such that $B^s \succ_a A^s$. We define $\succ_\pi$ to be the transitive closure of $\gg$.

Lemma 65 *Let S be a set of clauses, C be a ground clause and π be a ground proof of $S \models C$. If A, B are two atoms in the proof π and $A \succ_\pi B$ then $A \rightarrow^*_{\mathcal{R}_S^g} B$.*

PROOF.

Let $M = \{B \in \mu(\pi)\ such\ that\ \exists A \in \mu(\pi), A \succ_\pi B\ and\ B \notin A\downarrow_S\}$ and let us prove that $M = \emptyset$ by contradiction. Suppose that $M \neq \emptyset$ and let B be a maximal atom in M for the ordering $\succ_\pi$. B can not be maximal in $\mu(\pi)$ for $\succ_\pi$ otherwise there would exist no atom $A \in \mu(\pi)$ such that $A \succ_\pi B$ which would imply $B \notin M$. Thus there exists an atom $A \in \mu(\pi)$ such that $A \succ_\pi B$. Let $M_B = \{A\ such\ that\ A \in \mu(\pi)\ and\ A \succ_\pi B\}$. The argument above proves that $M_B \neq \emptyset$.

Case 1. Let A be a minimal atom in M_B for the ordering $\succ_\pi$. By minimality of A in M_B and maximality of B in M there is no atom $D \in \mu(\pi)$ such that $A \succ_\pi D \succ_\pi B$. There is a ground clause $C\sigma$ instance of a clause C in S such that $C\sigma$ is labelling a leave in the proof π, and there are two atoms A, B in $C\sigma$, and two atoms A^s, B^s in C such that $A^s\sigma = A$, $B^s\sigma = B$ and $A^s \succ_a B^s$. By definition of the relation $\mathcal{R}_S^g$, we thus have $A \rightarrow_{\mathcal{R}_S^g} B$.

Case 2. If A is not minimal in M_B for $\succ_\pi$, there exists a minimal atom $A' \in M_B$ such that $A \succ_\pi A' \succ_\pi B$. By the maximality of B in M, $A' \notin M$ and thus

$A \rightarrow^*_{\mathcal{R}^g_S} A'$ and by minimality of A' in M_B the first case implies $A' \rightarrow_{\mathcal{R}^g_S} B$. By transitivity we thus have $A \rightarrow^*_{\mathcal{R}^g_S} B$.

Thus for every atom $A \in M_B$ we have $A \rightarrow^*_{\mathcal{R}^g_S} B$, which contradicts the assumption B is maximal in M, and therefore $M = \emptyset$. ■

Lemma 66 *Let S be a set of clauses and C be a ground clause, $C = A_1, \ldots, A_n \rightarrow B$. Let π be a ground proof of $S \models C$ and let A be an atom in π maximal with respect to atoms of π for the ordering $\succ_\pi$. There exists a clause $C' \in S \cup \bigcup_{i=1}^n A_i \cup \neg B$ and an atom $A' \in \mu(C')$ such that A' in maximal with respect to atoms of C' for the ordering $\succ_a$ and $A'\sigma = A$.*

PROOF.

Let π be a ground proof of $S \models C$. By definition, leaves of π are labelled by ground instances of clauses in S, positive unit clauses $\emptyset \rightarrow A_i$ for $1 \leq i \leq n$ and negative unit clause $B \rightarrow \emptyset$. Let $A \in \mu(\pi)$ be such that A is maximal with respect to atoms of π for the ordering $\succ_\pi$. As atoms in π are ground, we have either $A \in \{A_1, \ldots, A_n, B\}$ or there is a ground instance C_1 of a clause in S, such that $A \in \mu(C_1)$. If $A \in \{A_1, \ldots, A_n, B\}$ and as each atom in the set $\{A_1, \ldots, A_n, B\}$ is a unit clause then we conclude the lemma directly. Now, suppose that $A \notin \{A_1, \ldots, A_n, B\}$. Then there exists a ground instance of a clause in S, C_1 such that $A \in \mu(C_1)$. As C_1 is a ground instance of a clause in S, let C_1^s be that clause in S and let A^s be an atom in C_1^s such that A is a ground instance of A^s. We have σ is the applied ground substition. Now, let us prove that A^s is maximal with respect to atoms in C_1^s for the ordering $\succ_a$. Let the set of atoms $M = \{D \; such \; that \; D \; is \; an \; atom \; in \; C_1^s \; and \; D \; is \; maximal \; in \; C_1^s \; for \succ_a\}$ and let us prove that $A^s \in M$. By contradiction, suppose that $A^s \notin M$, then there is an atom $E^s \in M$ such that $E^s \succ_a A^s$ and then $E^s\sigma \succ_\pi A^s\sigma$, by definition of the ordering $\succ_\pi$. We have that $E^s\sigma \in C_1$ and then $E^s\sigma \succ_\pi A^s\sigma$ is an atom of the proof π and that contradicts the maximality of $A = A^s\sigma$ with respect to atoms of π for the ordering $\succ_\pi$. We conclude that A^s is maximal with respect to atoms of C_1^s for the ordering $\succ_a$ which concludes the proof. ■

Let π be a proof of $S \models C$ where S is a set of clauses and C is a ground clause. By definition of (refutational) proof, we remark that every atom appearing in π appears in a clause labelling a leave of π.

Lemma 67 *Let S be a finite saturated set of clauses, C be a ground clause and Π be the non-empty set of refutational ground proofs of $S \models C$. Given $\pi \in \Pi$ let:*

$$\delta_S(\pi, C) = (\mu(\pi) \downarrow_S) \setminus (\mu(C) \downarrow_S)$$

If $\pi \in \Pi$ is such that $\delta_S(\pi, C)$ is minimal then $\delta_S(\pi, C) = \emptyset$.

PROOF.

Let S be a finite saturated set of clauses and C be a ground clause such that Π is not empty. Let finally $\pi \in \Pi$ be such that $\delta_S(\pi, C)$ is minimal. Such π exists as atoms in π are ground and the ordering $\prec_a^{set}$ is well-founded. Since $\delta_S(\pi, C)$ depends only on the atoms of π we can assume w.l.o.g. that π is a proof by selected resolution. By contradiction assume $\delta_S(\pi, C) \neq \emptyset$, and let A be the maximal atom in $\delta_S(\pi, C)$ for the atom ordering $<_a$.

Claim. The atom A must be maximal for $\prec_\pi$.

> **Proof of the claim.** Otherwise there would exist an atom $B \in \mu(\pi)$ such that $B \rightarrow_S A$ by definition of $\prec_\pi$. If $B \in \mu(C)\downarrow_S$ then we could not have $A \in \delta_S(\pi, C)$. If $B \notin \mu(C)\downarrow_S$ then $B \in \delta_S(\pi, C)$, and hence A would not be maximal.

Let $\mathcal{T}_A$ be the set of subtrees of π where the end resolution inference eliminates the atom A and let π_A be a minimal subtree in $\mathcal{T}_A$. By minimality of π_A, its end inference is its unique inference that eliminates the atom A. Let $\frac{\Gamma_1 \rightarrow A, \Delta_1 \qquad A, \Gamma_2 \rightarrow \Delta_2}{\Gamma_1, \Gamma_2 \rightarrow \Delta_1, \Delta_2}$ be that inference. We remark that π is ground and that each atom appearing in the proof π is an atom appearing in a clause labelling a leave of π. Furthermore, by definition of refutational proof and selected resolution, there is a leave in π_A labelled by a clause containing the atom A in its succedent, and there is another leave in π_A labelled by a clause containing the atom A in its antecedent. By the maximality of A with respect to atoms of π for the ordering $\succ_\pi$, and by the previous lemma (Lemma 66), we have either $A \in \mu(C)$ or there exists a clause C' in S and an atom D in C' such that D is maximal with respect to atoms of C' for the ordering $\succ_a$ and $D\sigma = A$ for a ground substitution σ. We prove that $A \in \mu(C)$ by contradiction. Suppose that $A \notin \mu(C)$, and let C_1 and C_2 be two ground instances of clauses in S, C_1^s and C_2^s, such that $C_1,\ C_2$ label leaves in $\pi 8A$, A appears as positive literal in C_1 and negative literal in C_2, $C_1 = C_1^s\sigma$ and $C_2 = C_2^s\sigma$ for a ground substitution σ. Let A_1^s and A_2^s be two atoms respectively in C_1^s and C_2^s such that A is a ground instance of A_1^s and A_2^s. We have $C_1 = \Gamma \rightarrow A, \Delta$, $C_2 = A, \Gamma' \rightarrow \Delta'$, $C_1^s = \Gamma^s \rightarrow A_1^s, \Delta^s$ and $C_2^s = A_2^s, \Gamma'^s \rightarrow \Delta'^s$. Furthermore, A_1^s, A_2^s are maximals in C_1^s and C_2^s respectively for the ordering $\succ_a$ (by maximality of A for $\succ_\pi$), and $A_1^s\sigma = A_2^s\sigma = A$. This implies that A_1^s and A_2^s are unifiable and let θ be their most general unifier, $\sigma = \theta\theta'$ for a ground substitution θ'. By definition, selected resolution can be applied on C_1^s and C_2^s and outputs the clause C_3^s, $C_3^s = \Gamma^s\theta, \Gamma'^s\theta \rightarrow \Delta^s\theta, \Delta'^s\theta$. Let $C_3 = C_3^s\theta' = \Gamma, \Gamma' \rightarrow \Delta, \Delta'$.

For each possible C_1, C_2 and for each C_3 obtained, since the resolution is an instance of a resolution by selection between two clauses of S, there are two possibilities:

- **If C_3 is a ground instance of a clause in S (*i.e.* $C_3^s \in S$).** Then we have $\mu(C_3) = \mu(C_1, C_2) \setminus \{A\}$ and thus:

$$\begin{array}{rcl} \mu(C_3) & \subseteq & \mu(\pi) \setminus \{A\} \\ & \subseteq & \mu(\pi) \downarrow_S \setminus \{A\} \end{array}$$

  The second inclusion is correct since A is maximal in $\delta_S(\pi, C)$, and thus in $\mu(\pi)$.

- **Otherwise,** by definition of the saturation algorithm the selected resolution inference applied on C_1^s and C_2^s is redundant and, by definition of redundancy, all its ground instances are redundant and hence the inference $\frac{\Gamma \to A, \Delta \qquad A, \Gamma' \to \Delta'}{\Gamma, \Gamma' \to \Delta, \Delta'}$ is redundant. This implies that the clause $C_3 = \Gamma, \Gamma' \to \Delta, \Delta'$ can be deduced from a finite set S' of ground clauses instances of clauses in S such that

$$\begin{array}{rcl} \mu(S') & \subseteq & \mu(C_1, C_2) \downarrow_S \setminus \{A\} \\ & \subseteq & \mu(\pi) \downarrow_S \setminus \{A\} \end{array}$$

  Again, the second inclusion holds because A is maximal in $\delta_S(\pi, C)$, and thus in $\mu(\pi)$.

By iterating the construction on π we find another proof π' such that

$$\mu(\pi') \subseteq \mu(\pi) \downarrow_S \setminus \{A\}$$

Thus:

$$\mu(\pi') \downarrow_S \subseteq (\mu(\pi) \downarrow_S \setminus \{A\}) \downarrow_S$$

Since A is a maximal in π for $\prec_a$ this implies:

$$\mu(\pi') \downarrow_S \subseteq ((\mu(\pi) \setminus \{A\}) \downarrow_S \cup (\{A\} \downarrow_S \setminus \{A\})) \downarrow_S = (\mu(\pi) \downarrow_S \cup \{A\} \downarrow_S) \downarrow_S \setminus \{A\} =$$

Since $A \in \delta_S(\pi, C)$ we have $A \notin \mu(C) \downarrow_S$, and thus we have:

$$\begin{array}{rcl} \delta_S(\pi', C) & = & \mu(\pi') \downarrow_S \setminus \mu(C) \downarrow_S \\ & \subseteq & ((\mu(\pi) \downarrow_S \cup \{A\} \downarrow_S) \setminus \{A\}) \setminus \mu(C) \downarrow_S \\ & \subseteq & ((\mu(\pi) \downarrow_S \cup \{A\} \downarrow_S) \setminus \mu(C) \downarrow_S) \setminus \{A\} \\ & < & \{A\} \\ & < & \delta_S(\pi, C) \end{array}$$

Thus π is not such that $\delta_S(\pi, C)$ is minimal. ■

Lemma 68 *Let S be a saturated set of clauses, C be a ground clause and Π be a non-empty set of proofs of $S \models C$. Let $\pi \in \Pi$ be such that $\delta_S(\pi, C)$ is minimal. If A is an atom in the proof π then there is an atom $B \in \mu(C)$ such that $B \rightarrow^*_{\mathcal{R}^g_S} A$.*

PROOF.

By Lemma 67, the minimality of $\delta_S(\pi, C)$ implies that $\delta_S(\pi, C) = \emptyset$. This implies that if A is an atom in π ($A \in \mu(\pi)$) then $A \in \mu(C) \downarrow_S$, and hence, by definition, there is an atom $B \in \mu(C)$ such that $B \rightarrow^*_{\mathcal{R}^g_S} A$. ■

Theorem 14 *If the set of clauses S is saturated by selected resolution up to redundancy with respect to $\succ_a$ then ground entailment problem for S is decidable.*

PROOF.

Let S be a saturated set of clauses. Given a ground clause C, we prove that we can decide if $S \models C$.We make use of the fact that $S \models C$ if and only if $S \cup \neg C$ is unsatisfiable. By Lemma 59 and Lemma 60, $S \cup \neg C$ is unsatisfiable if and only if the set of $\mathcal{T}$ of ground refutational proofs of $S \cup \neg C \models \emptyset$ is not empty. Let π be a proof in $\mathcal{T}$ having the minimal $\delta_S(\pi, C)$. By lemma 68, the atoms appearing in the proof π belongs to the set $\mu(C) \downarrow_S$ and since the clause C is ground, Lemma 64 implies that the set $\mu(C) \downarrow_S$ is finite.Now, the strategy for deciding the problem $S \models C$ is to construct first the set $\mu(C) \downarrow_S$, then to construct the set S' of ground instances of clauses of S such that for every ground clause $C' \in S'$, and for every atom $A \in \mu(C')$, $A \in \mu(C) \downarrow_S$. By finiteness of the set $\mu(C) \downarrow_S$, the set S' is finite. We construct now the set of trees Π such that for each tree, the leaves are labelled by $\neg L$ with L is a literal in C and by clauses in S', and internal nodes are labelled by the result of selected resolution from their direct descendents. Since S' is finite, the set Π is finite. Moreover, by definition of selected resolution and since S' and S are ground, we have that each tree in Π is finite. Finally, we have $S \models Cl$ if and only if there is a tree in the set Π whose root is labelled by the empty clause, and as the existence of such tree is decidable then, the problem $S \models C$ for a ground clause C is decidable, and hence the ground entailment problem for S is decidable. ■

Example 25 *We consider the Needham-Schroeder symmetric key protocol described in Example 24. We analyse the security of this protocol in the presence of an intruder $\mathcal{I}$ to which we associate the following set of clauses, $C_{\mathcal{I}} = C_{DY} \cup C_{pre}$. The set of clauses C_{DY} is given in Example 23 and the clause C_{pre} is given in Example 24. As shown in Section 6.4.1, the insecurity problem for this protocol is reduced to the ground entailment problem for $C_{\mathcal{I}}$. We show that the decidability of insecurity problem for this protocol can be deduced from Theorem 14. In fact, $C_{\mathcal{I}}$ is saturated by selected resolution up to redundancy with respect to a complete atom ordering compatible with a complete simplification term ordering. Theorem 14 implies that the ground entailment problem for $C_{\mathcal{I}}$ is decidable, and hence the insecurity problem of Needham-Schroeder symmetric*

key protocol is decidable under a bounded number of sessions and under the hypothesis that the protocol runs correctly.

6.6 Discussions and conclusions

The main contribution of this chapter is a decidability result of the ground entailment problem for a set S of clauses under the hypothesis that S is saturated by selected resolution up to redundancy with respect to a complete atom ordering compatible with a complete simplification ordering over terms.

Many decidability results have been obtained for this problem, and we discuss some of them in Section 6.5. Among the works cited above, the closest to ours is [28]. In [28], D. Basin and H. Ganzinger proved the decidability of the ground entailment problem for a set S of clauses under the assumptions that (i) S is saturated up to redundancy under ordered resolution with respect to a complete well-founded ordering over atoms and that (ii) each maximal atom in each clause in S contains all variables appearing in the other atoms in the clause. In this chapter, we relax the condition (ii), and in order to get the decidability, we use an order more restrictive than the one used in [28]. Another contribution of this chapter is the reduction of insecurity problem of cryptographic protocols to a ground entailment problem in the first order logic. In this line of research, many results have been obtained, and some of them have been discussed in Section 6.3. We remark that our result is different than the results discussed in that section. In fact, we define a new model to analyse cryptographic protocols using Horn clauses, which is different that the models given in [84, 180, 205]. Moreover, in [84, 180, 205], the decidability results are obtained for Horn clauses and not full clauses as in this chapter. while in [84, 205], they can deal with an unbounded number of sessions, we consider only a bounded number of sessions but our result allows us to deal with a class of cryptographic protocols less restrictive than [84, 205].

In future works, we plan implement the decision procedure with respect to the result of D. Basin and H. Ganzinger [28], we remark that for ground entailment problems, it suffices to consider inferences by *selected resolution* in which one of the atoms is ground. Thus we do not need to construct the set of ground atoms before solving an entailment problem. Furthermore, we plan to extend the result to the case where the clauses are considered modulo an equational theory.

We thanks Michaël Rusinowitch for all the discussions we had concerning the results of this chapter

Chapter 7

Formalising voter verifiability properties for electronic voting protocols

The multiple benefits of *electronic-voting* push many governments to change their traditional methods and to move toward the electronic democracy. In fact, electronic voting allows, amongst others, to reduce the cost of the vote, to get the results faster and more accurately, and to reduce the risk of human errors. Over the last years, several electronic-voting systems have been proposed and studied.

In this chapter, we study and formally define the *voter verifiability property* for electronic voting protocols using the *applied pi calculus* [11], which is well suited for modelling security protocols and in particular electronic voting protocols [97, 25]. The concept of *voter verifiability* has emerged as an important property for electronic voting. It means that voters and observers can verify that the declared outcome corresponds to the votes cast. Voter verifiability property includes the *individual verifiability* and *universal verifiability* properties. Intuitively, a protocol satisfies voter verifiability if there is a test that a voter or observer can perform on the output of the election process; the test succeeds if and only if the output corresponds to the votes cast, and the declared outcome corresponds to the output. The test is in two parts, corresponding to the two aspects of voter verifiability mentioned above. The definition is a sufficient condition for voter verifiability, but it may not be necessary; there could be some protocols which offer voter verifiability but are not captured by our definition.

Our definition may be applied across a range of voting protocols. We give examples of protocols that are voter verifiable, and others

which are not: we show that the protocols due to Fujioka, Okamoto & Ohta [115] and Lee *et al.* [141] are voter verifiable.

Intuitively, voter verifiability may appear to contradict another important property of voting systems, namely *coercion-resistance*. If a voter is capable of verifying that her vote has been included in the tally, she may be able to use that capability to convince a coercer that she voted as he ordered. However, there are systems that satisfy both, such as the system by Lee *et al.* [141] which we consider in Section 7.5.4. This is achieved by ensuring that what constitutes a proof for the voter will not be a proof for the coercer. Therefore, it is important to ensure that voter verifiability is not defined so strongly that it is incompatible with coercion-resistance.

This work has been done in collaboration with Mark Ryan, Ben Smyth, and Steve Kremer during my internship at the School of Computer Science (University of Birmingham) from October, 2008 until January, 2009. A more recent version will be presented in the *4th Benelux Workshop on Information and System Security (WISSec 2009)* [191].

Outline. We introduce in Section 7.1 the electronic voting protocols. In Section 7.2, we introduce the applied pi calculus. The formalisation of electronic voting protocols and voter verifiability properties are given in Section 7.3 and Section 7.4. We give in Section some examples, and the related works are given in Section 7.6.

7.1 Electronic voting protocols

Electronic voting, also known as *e-voting*, is a term encompassing several different types of voting, embracing both electronic means of casting a vote and electronic means of counting votes. Electronic voting systems can include *punch cards*, *optical scan voting systems* and *specialised voting kiosks, including self-contained Direct-recording electronic (DRE) voting systems*. It can also involve transmission of ballots and votes via telephones, private computer networks, or the Internet. Electronic voting may offer advantages compared to other voting techniques. It promises the possibility of a convenient, efficient and secure facility for recording and tallying votes. It can be used for a variety of types of elections, from small committees or on-line communities through to full-scale national elections. Electronic voting for electorates have been in use since the 1960. However, the electronic voting machines used in recent US elections have been fraught with problems. In [132], the authors have analysed the source code of the machines sold by the second largest and fastest-growing vendor, which

are in use in 37 US states. This analysis has produced a catalogue of vulnerabilities and possible attacks.

A potentially much more secure system could be implemented, based on cryptographic protocols that specify the messages sent between the voters and administrators. Such systems are called *electronic voting protocols*.

Abstractly, electronic voting protocols are cryptographic protocols that specify the messages sent between the voters and administrators, they can be written as a sequence of messages sent between voters and administrators. Such protocols have been studied for several decades [97, 136, 30, 39, 62, 63, 65], and a various types of electronic voting protocols have been proposed in the literature [61, 64, 39, 92, 62, 115, 141, 181]. These protocols aim to provide security properties which go beyond those that can be achieved by paper-based voting systems. Some of these properties are given next.

7.1.1 Properties of electronic voting protocols

Some properties commonly sought for electronic voting protocols are the following:

- **Fairness:** no early results can be obtained which could influence the remaining voters.
- **Vote-privacy:** the fact that a particular voter voted in a particular way is not revealed to anyone.
- **Receipt-freeness:** a voter does not gain any information which can be used to prove to a coercer that she voted in a certain way.
- **Coercion-resistance:** a voter can not cooperate with a coercer to prove to him that she voted in a certain way.
- **Inalterability:** no one can change a voter's vote.
- **Declared tally:** the published outcome is a correct sum of the votes cast.
- **Eligibility:** only legitimate voters can vote, and only once.
- **Individual verifiability:** a voter can check that her ballot was included in the tally.
- **Universal verifiability:** anybody can check the correctness of the published outcome.
- **Eligibility verifiability:** anybody can check that each vote cast was created by a unique legitimate voter.

The properties *vote-privacy, receipt-freeness, coercion-resistance* are called *privacy-type properties* [97] since they guarantee that the link between the voter and her vote is not revealed by the protocol. They are considered by S. Delaune, S. Kremer and M. Ryan [97]. The properties *fairness, inalterability, eligibility* are called *correctness properties*, and some of them have been considered by M. Backes, C. Hritcu and M. Maffei [25]. The properties *individual verifiability, universal verifiability* are called *voter verifiability* and some of them have been considered by B. Chevallier-Mames et al. [76], Baskar et al. [30], Talbi et al. [193].

We remark that verifiability properties are related to the correctness properties, but verifiability properties are intuitively stronger than correctness properties. They allow voters and observers to check that the outcome is correct without having to trust the protocol, the administrators or the implementation. In contrast, correctness properties merely assert good behaviour in case the expected protocol was indeed executed by all the participants, *i.e.* one has to trust the software and the administrators.

7.2 Applied pi calculus

The applied pi calculus [11] is a language for describing concurrent systems and their interactions. It is based on the *pi calculus* [158, 159] which is a process calculus originally developed by R. Millen as a continuation of work on the process calculus *CCS* (*Calculus of Communicating Systems*). Th applied pi calculus is intended to be less pure than the pi calculus and therefore more convenient to use. The applied pi calculus is, in some sense, similar to the *spi calculus* [12]. The key difference between the two formalisms concerns the way the cryptographic primitives are handled. The spi calculus has a fixed set of primitives built-in (symmetric and public-key encryption), while the applied pi calculus allows one to define less usual primitives often used in electronic voting protocols, like zero knowledge proofs, by means of equational theory. The applied pi calculus has been used to study a variety of security protocols, such as a private authentication protocols [114], or a key establishment protocols [8], or electronic voting protocols [97].

In order to represent our properties, we slightly modify the applied pi calculus introduced in [11]. We assume an infinite set of names (which are used to name communication channels or other atomic data), an infinite set of variables and a finite signature Σ which consists of a finite set of function symbols each with an associated arity. A function symbol with arity 0 is a constant symbol. In the case of security protocols, typical function symbols will include enc for *encryption*, and dec for decryption. Terms are defined as names, variables, and function symbols applied to other terms. A terms is *ground* if it does not

contain variables. Although names, variables, and constant symbols haves similarities, they are kept separate. We let $a, b, c, m, n, r, s, t, v \ldots$ range over names and $x, y, z, \ldots$ over variables. We use metavariables u, w to range over both names and variables. F, L, M, N, R, T range over arbitrary terms. Terms and function symbols are sorted, and of course function symbol application must respect sorts and arities. We suppose that terms can be of sort Channel or Base type. Function symbols can only be applied to and return terms of base type. As a slight convenient extension, also introduced in [83], we suppose a set of predicates $\mathcal{P}$ over terms, each with an arity.

In the applied pi calculus, one has *plain processes* and *extended processes*. Plain processes are built up in a similar way to processes in the pi calculus, except that messages can contain terms (rather than only names). The grammar for plain processes is shown below where c is supposed to be of channel sort:

$\psi ::=$	tests
$\quad p(M_1, \ldots, M_n)$	predicate
$\quad \psi_1 \wedge \psi_2$	conjunction
$P, Q, V ::=$	processes
$\quad 0$	null process
$\quad P \mid Q$	parallel composition
$\quad \nu\, n.P$	name restriction
$\quad c(x).P$	message input
$\quad \overline{c}\langle N \rangle.P$	message output
$\quad$ if ψ then P else Q	conditional

The *null process* 0 does nothing; $P \mid Q$ is the parallel composition of P and Q. the process $\nu\, n.P$ makes a new, private name n then behaves as P. The process $c(x).P$ is ready to input from channel c, then to run P with the actual message replaced for the formal parameter x, while the process $\overline{c}\langle N \rangle.P$ is ready to output N on channel c, then to run P. In all cases, we omit P when it is 0. Finally, if ψ then P else Q behaves in the standard way.

Extended processes add *active substitutions* and *restriction on variables*, their grammar is shown below:

$A, B, C ::=$	extended processes
$\quad P$	plain process
$\quad A \mid B$	parallel composition
$\quad \nu\, n.A$	name restriction
$\quad \nu\, x.A$	variable restriction
$\quad \{^M/_x\}$	active substitution

We write $\{^M/_x\}$ for the substitution that replaces the variable x with the term M. Active substitutions generalise 'let' and the process $\nu\, x.(\{^M/_x\} \mid P)$

corresponds exactly to the process let $x = M$ in P. As usual, names and variables have scopes which are delimited by restrictions $\nu\, u$ and by inputs $c(x)$. We write $\mathsf{fv}(A)$, $\mathsf{bv}(A)$, $\mathsf{fn}(A)$, $\mathsf{bn}(A)$ for the sets of free and bound variables, respectively names, in A. An extended process A is *closed* if it has no free variables. We suppose that $\mathsf{bn}(A) \cap \mathsf{fn}(A) = \emptyset$ for any extended process A and names are at most bound once. A *frame* is an extended process built from 0, active substitutions and restriction. We write $\phi(A)$ for the frame obtained by replacing every plain process in A with 0. The domain $\mathsf{dom}(\varphi)$ of a frame φ is the set of variables that φ exports, *i.e.*, the variables x in active substitutions $\{^M/_x\}$ where x is not bound. It represents the static knowledge output by a process to its environment. A *context* $C[\_]$ is an extended process with a hole. We obtain $C[A]$ as the result of filling $C[\_]$'s hole with A. An *evaluation context* is a context whose hole is not under a replication, a conditional, an input, or an output.

The signature Σ is equipped with an equational theory $\mathcal{H}$, and we define the relation $=_{\mathcal{H}}$ as usual (Chapter 2). Predicates are interpreted as relations over closed terms modulo the equational theory $\mathcal{H}$. Conjunction of predicates are interpreted as expected. We always suppose that we have at least a binary predicate eq which holds on terms M_1, M_2 when $M_1 =_{\mathcal{H}} M_2$. For convenience we sometimes write $M = N$ for $\mathsf{eq}(M, N)$.

We define *structural equivalence* ($\equiv$) to be the smallest equivalence relation closed under α-conversion of bound variables, (the α-conversion of bound variables is obtained by renaming one or more bound variables), and application of evaluation contexts such that

$$
\begin{array}{lrcl}
\text{PAR-0} & A & \equiv & A \mid 0 \\
\text{PAR-A} & A \mid (B \mid C) & \equiv & (A \mid B) \mid C \\
\text{PAR-C} & A \mid B & \equiv & B \mid A
\end{array}
$$

$$
\begin{array}{lrcl}
\text{NEW-0} & \nu\, n.0 & \equiv & 0 \\
\text{NEW-C} & \nu\, u.\nu\, w.A & \equiv & \nu\, w.\nu\, u.A \\
\text{NEW-PAR} & A \mid \nu\, u.B & \equiv & \nu\, u.(A \mid B) \\
 & & & \text{where } u \notin \mathsf{fv}(A)
\end{array}
$$

$$
\begin{array}{lrcl}
\text{ALIAS} & \nu\, x.\{^M/_x\} & \equiv & 0 \\
\text{SUBST} & \{^M/_x\} \mid A & \equiv & \{^M/_x\} \mid A\{^M/_x\} \\
\text{REWRITE} & \{^M/_x\} & \equiv & \{^N/_x\} \\
 & & & \text{where } M =_{\mathcal{H}} N
\end{array}
$$

A slightly unusual detail is that we consider structural equivalence to be closed under α-conversion of variables but not names. This will be convenient in what follows to uniquely identify secrets, i.e., bound names, of a given voter.

Internal reduction ($\rightarrow$) is the smallest relation closed under structural equivalence and application of evaluation contexts and such that

$$\begin{array}{lll} \text{COMM} & \overline{c}\langle x\rangle.P \mid c(x).Q \to P \mid Q & \\ \text{THEN} & \text{if } \psi \text{ then } P \text{ else } Q \to P & \text{if } \psi \\ \text{ELSE} & \text{if } \psi \text{ then } P \text{ else } Q \to Q & \text{if } \neg\psi \end{array}$$

Example 26 *Consider the process $\nu\, a.(\nu\, k.\overline{a}\langle k\rangle \mid a(x).P)$ which models the distribution of key k using private channel a. We have $\nu\, a.(\nu\, k.\overline{a}\langle k\rangle \mid a(x).P) \equiv \nu\, x.(\{^k/_x\} \mid \nu\, a.\nu\, k.(\overline{a}\langle x\rangle \mid a(x).P))$. We can now model the communication using the COMM rule $\nu\, x.(\{^k/_x\} \mid \nu\, a.\nu\, k.(\overline{a}\langle x\rangle \mid a(x).P)) \to \nu\, x.(\{^k/_x\} \mid \nu\, a.\nu\, k.(0 \mid P)) \equiv \nu\, k.P\{^k/_x\}$ (where $a \notin fn(P)$).*

The labelled semantics ($\xrightarrow{\alpha}$) extends internal reduction by the following rules. We suppose that u is either a channel name or a variable of base type.

$$\text{IN} \qquad c(x).P \xrightarrow{c(M)} P\{^M/_x\}$$

$$\text{OUT-ATOM} \qquad \overline{c}\langle u\rangle.P \xrightarrow{\overline{c}\langle u\rangle} P$$

$$\text{OPEN-ATOM} \qquad \frac{A \xrightarrow{\overline{c}\langle u\rangle} A' \qquad u \neq c}{\nu\, u.A \xrightarrow{\nu\, u.\overline{c}\langle u\rangle} A'}$$

$$\text{SCOPE} \qquad \frac{A \xrightarrow{\alpha} A' \qquad u \text{ does not occur in } \alpha}{\nu\, u.A \xrightarrow{\alpha} \nu\, u.A'}$$

$$\text{PAR} \qquad \frac{A \xrightarrow{\alpha} A' \quad bv(\alpha) \cap fv(B) = bn(\alpha) \cap fn(B) = \emptyset}{A \mid B \xrightarrow{\alpha} A' \mid B}$$

$$\text{STRUCT} \qquad \frac{A \equiv B \qquad B \xrightarrow{\alpha} B' \qquad B' \equiv A'}{A \xrightarrow{\alpha} A'}$$

Example 27 *Continuing example 26, and let us consider the process $\nu\, a.(\nu\, k.\overline{a}\langle k\rangle \mid a(x).\overline{c}\langle \mathsf{sign}(m,k)\rangle)$ i.e. we have added the output of a signed message using the key distributed. Since internal reduction is closed by structural equivalence and by our previous reasoning we have $\nu\, a.(\nu\, k.\overline{a}\langle k\rangle \mid a(x).\overline{c}\langle \mathsf{sign}(m,k)\rangle) \to \nu\, k.\overline{c}\langle \mathsf{sign}(m,k)\rangle$. We model the output of $\mathsf{sign}(m,k)$ to the environment using labelled semantics: $\nu\, k.\overline{c}\langle \mathsf{sign}(m,k)\rangle \xrightarrow{\nu\, x.\overline{c}\langle x\rangle} \nu\, k.\{^{\mathsf{sign}(m,k)}/_x\}$.*

An extended process A is said to be *irreducible* if there does not exist B, α such that $A \to B$ or $A \xrightarrow{\alpha} B$.

Remark. We will abbreviate $(M_1, \ldots, M_n)$ as $\tilde{M}$ and occasionally we write $\tilde{x}_f$ for $(x_{f(1)}, \ldots, x_{f(n)})$ where $f : \{1, \ldots, n\} \to \{1, \ldots, m\}$. We abbreviate $\{^{M_1}/_{x_1}\}|\ldots|\{^{M_k}/_{x_k}\}$ as $\{^{\tilde{M}}/_{\tilde{x}}\}$ or σ. We also write $\sigma, \{^M/_x\}, \{^{\tilde{M}}/_{\tilde{x}}\}$ for substitions

and $N\sigma$ for the result of applying σ to N. We write $\tilde{m}^i$ for the tuple of names obtained from $\tilde{m}$ by annotating with subscript i. For example given $\tilde{m} = (n, n')$ we write $\tilde{m}^1$ for (n_1, n'_1). Finally, we write $a(x_1, ..., x_n).P$ as a convenient shorthand for $a(x).\mathsf{let}\ x_1 = \mathsf{nth}_1^n(x)\ \mathsf{in}\ \ldots\ \mathsf{let}\ x_n = \mathsf{nth}_n^n(x)\ \mathsf{in}\ P$.

7.3 Formalising electronic voting protocols

We refine our sort system with the new base type Candidate. A voting process is characterised with respect to a voter process V and administrators processes. We capture administrators who are assumed to be honest by the process P. Dishonest administrators do not need to be modelled as they are directly part of the adversarial environment in which the protocol is executed. Processes V, P will each be instantiated n times where n is the number of voters. We assume secrets $\tilde{s}$ only occur in process V; similarly $\tilde{t}$ only occur in process P; additionally, shared secrets $\tilde{m}$ appear in both V, P. Definition 58 formalises a voting process specification accordingly.

Definition 58 (Voting process specification) *A voting process specification is a tuple $\langle V, P, \tilde{s}, \tilde{t}, \tilde{m}\rangle$ where plain processes V, P do not contain any name restrictions, the tuples of names $\tilde{s}, \tilde{t}, \tilde{m}$ are disjoint and $(fn(V) \cup fn(P)) \cap \bigcup_{j\in\mathbb{N}}(\tilde{s}^j \cup \tilde{t}^j \cup \tilde{m}^j) = fn(V) \cap \tilde{t} = fn(P) \cap \tilde{s} = \emptyset$. We write $VP(\{^{\bar{v}_1}/_u\}, \ldots, \{^{\bar{v}_n}/_u\})$ for:*

$$\nu\, \tilde{m}^1, \ldots, \tilde{m}^n, \tilde{s}^1, \ldots, \tilde{s}^n, \tilde{t}^1, \ldots, \tilde{t}^n \big(V\{^{\tilde{s}^1}/_{\tilde{s}}, ^{\tilde{m}^1}/_{\tilde{m}}, ^{\bar{v}_1}/_u\} \mid P\{^{\tilde{t}^1}/_{\tilde{t}}, ^{\tilde{m}^1}/_{\tilde{m}}\} \mid \ldots \mid V\{^{\tilde{s}^n}/_{\tilde{s}}, ^{\tilde{m}^n}/_{\tilde{m}}, ^{\bar{v}_n}/_u\} \mid P\{^{\tilde{t}^n}/_{\tilde{t}}, ^{\tilde{m}^n}/_{\tilde{m}}\}\big)$$

where $\bar{v}_i$ of type Candidate *is the ith voter's vote.*

As an example consider the simple *"postal ballot"* voting protocol (Figure 7.1). A voter V receives her private signing key sk_V from a keying authority P, and then sends her signed vote to the bulletin board. The keying authority also sends the voter's public verification key $Pk(sk_V)$ to the bulletin board. By convention we use the channel bb (referring to the bulletin board). We consider the equational theory $\mathcal{H}$ defined by $\mathsf{getmsg}(\mathsf{sign}(x, y)) = x$ and the predicate checksign defined as $\mathsf{checksign}(M_1, M_2)$ if and only if $M_1 =_{\mathcal{H}} \mathsf{sign}(N_1, N_2) \wedge M_2 =_{\mathcal{H}} Pk(N_2)$.

To analyse verifiability, we consider a modified version of the voting process, in which all inputs that a voter receives are stored on the frame (perhaps in encoded form). To achieve this we extend the equational theory to include functions $\mathsf{senc}/2, \mathsf{sdec}/2$, the associated equation $\mathsf{sdec}(\mathsf{senc}(x, y), y) = x$ and define the process modification (Definition 59).

Definition 59 (Input-storing process) *Given channel c and process $\nu\,\tilde{n}.P$ where P has no restrictions, the* input-storing process *is defined as $\nu\,k, \tilde{n}.P^{c,\tilde{n},k}$ where:*

- $0^{c,\tilde{n},k} \mathrel{\widehat{=}} 0$
- $(P \mid Q)^{c,\tilde{n},k} \mathrel{\widehat{=}} P^{c,\tilde{n},k} \mid Q^{c,\tilde{n},k}$
- $(\nu\, m.P)^{c,\tilde{n},k} \mathrel{\widehat{=}} \nu\, m.P^{c,\tilde{n},k}$
- $(a(x).P)^{c,\tilde{n},k} \mathrel{\widehat{=}} a(x).\overline{c}\langle\mathsf{senc}(x,k)\rangle.P^{c,\tilde{n},k}$ *if* $a \in \tilde{n}$
- $(a(x).P)^{c,\tilde{n},k} \mathrel{\widehat{=}} a(x).\overline{c}\langle x\rangle.P^{c,\tilde{n},k}$ *otherwise*
- $(\overline{a}\langle M\rangle.P)^{c,\tilde{n},k} \mathrel{\widehat{=}} \overline{a}\langle M\rangle.P^{c,\tilde{n},k}$
- $(\textit{if } \psi \textit{ then } P \textit{ else } Q)^{c,\tilde{n},k} \mathrel{\widehat{=}} \textit{if } \psi \textit{ then } P^{c,\tilde{n},k} \textit{ else } Q^{c,\tilde{n},k}$

Given a voting process specification $\langle V, P, \tilde{s}, \tilde{t}, \tilde{m}\rangle$ and votes $\bar{v}_1, \ldots, \bar{v}_n$ we write $\widetilde{VP}(\{{}^{\bar{v}_1}/_{u}\}, \ldots, \{{}^{\bar{v}_n}/_{u}\})$ for the special input-storing process *defined as follows:*

$$\nu\,\tilde{n}.(V\{{}^{\tilde{s}^1}/_{\tilde{s}}, {}^{\tilde{m}^1}/_{\tilde{m}}, {}^{\bar{v}_1}/_{u}\}^{bb,\tilde{n},kpc_1} \mid P\{{}^{\tilde{t}^1}/_{\tilde{t}}, {}^{\tilde{m}^1}/_{\tilde{m}}\} \mid \ldots \mid V\{{}^{\tilde{s}^n}/_{\tilde{s}}, {}^{\tilde{m}^n}/_{\tilde{m}}, {}^{\bar{v}_n}/_{u}\}^{bb,\tilde{n},kpc_n} \mid P\{{}^{\tilde{t}^n}/_{\tilde{t}}, {}^{\tilde{m}^n}/_{\tilde{m}}\})$$

where $\tilde{n} = \bigcup_{j=1}^{n}(\tilde{m}^j \cup \tilde{s}^j \cup \tilde{t}^j \cup \{kpc_j\})$.

The definition is illustrated on the *postal ballot* example in Figure 7.1.

7.4 Formalising voter verifiability property

Now we introduce voter verifiability. As mentioned, there are two parts, corresponding to individual verifiability and universal verifiability. A voting process satisfies voter verifiability if there are two tests it can apply to check these two items. Each test is a predicate which, after substitutions from the bulletin board and elsewhere, evaluates to true or false.

Individual verifiability: The test R^{IV} is performed by a voter, and has parameters u (the vote cast by the voter), $x_1, \ldots, x_k$ (the items on the bulletin board corresponding to that vote), and $\tilde{z}$ (the secrets of the voter). The test is required to return true if and only if the correct items are given. In the definition below, the functions $f_1, \ldots, f_k$ pick out the k bulletin board items corresponding to the voter.

Universal verifiability: The test R^{UV} is performed by an observer, and has parameters $\tilde{u}$ (the declared outcome), $\tilde{x}_1, \ldots, \tilde{x}_k$ (the items on the bulletin board corresponding to all the voters), and $\tilde{y}$ (the items on the bulletin

board from the administrators). To model the possibility of dishonest administrators, we allow the attacker to supply $\tilde{y}$ by means of the substitution τ. The test is required to return true if and only if the declaration corresponds to the votes actually cast.

Definition 60 (Voter verifiability) *A voting process specification $\langle V, P, \tilde{s}, \tilde{t}, \tilde{m}\rangle$ satisfies* voter verifiability *if and only if for all n and votes $\bar{v}_1, \ldots, \bar{v}_n$ of type* Candidate*, there exist tests R^{IV}, R^{UV}, such that $(fv(R^{IV}) \cup fv(R^{UV})) \cap \{x'_1, x'_2, \ldots\} = fn(R^{IV}) \cup fn(R^{UV}) = \emptyset$ and for all irreducible extended processes B where $\phi(B) = \nu\, \tilde{n}.\sigma$,*

if

a) $\widetilde{VP}(\{\bar{v}_1/_u\}, \ldots, \{\bar{v}_n/_u\})(\rightarrow^* \xrightarrow{\alpha} \rightarrow^*)^* B$*; and*

b) $dom(\sigma) = \{x'_1, \ldots, x'_{k \cdot n}\}$*;*

where k is defined such that $k \cdot n$ is the number of occurrences of $\bar{c}\langle M\rangle$ in $\widetilde{VP}(\{\bar{v}_1/_u\}, \ldots, \{\bar{v}_n/_u\})$ for $c \notin \tilde{n}$,

then there exists injective maps $f_1, \ldots, f_k : \{1, \ldots, n\} \rightarrow \{1, \ldots, k \cdot n\}$ where the ranges of $f_1, \ldots, f_k$ are pairwise disjoint and the conditions below are satisfied. Moreover, we require the existence of B satisfying Conditions a) & b).

1. *Individual verifiability. For all $i_1, \ldots, i_k, j, v'$ we have:*

$$R^{IV}\{v'/_u, x'_{f_1(i_1)}/_{x_1}, \ldots, x'_{f_k(i_k)}/_{x_k}, \tilde{s}'_j/_{\tilde{z}}\}\sigma \Leftrightarrow i_1 = i_2 = \ldots = i_k = j \wedge v' = \bar{v}_j$$

   where $\tilde{s}'_j = (s_{1,j}, \ldots, s_{|\tilde{s}|,j}, kpc_j)$.

2. *Universal verifiability. For all $\tilde{v}'$ we have:*

$$\exists \tau.(dom(\sigma) \cap dom(\tau) = \emptyset \wedge R^{UV}\{\tilde{v}'/_{\tilde{u}}, \tilde{x}'_{f_1}/_{\tilde{x}_1}, \ldots, \tilde{x}'_{f_k}/_{\tilde{x}_k}\}\tau \circ \sigma) \Leftrightarrow \tilde{v}' = (\bar{v}_1, \ldots, \bar{v}_n)$$

We recall that $\widetilde{VP}$ is a modification of VP which stores all inputs the voter receives on the frame. Public channel inputs $a(x)$ are stored without modifications, whereas private channel inputs $a'(y)$ are stored in the form $\mathsf{senc}(y, k)$. The R^{IV} test for the original protocol VP can be extracted by replacing such inputs x and y into the test.

For convenience we use the shorthand $R^{IV}\Phi$ for $R^{IV}\{v'/_u, x'_{f_1(i_1)}/_{x_1}, \ldots, x'_{f_k(i_k)}/_{x_k}, \tilde{s}_j/_{\tilde{z}}\}\sigma$.
Similarly we write $R^{UV}\Phi$ for $R^{UV}\{\tilde{v}'/_{\tilde{u}}, \tilde{x}'_{f_1}/_{\tilde{x}_1}, \ldots, \tilde{x}'_{f_k}/_{\tilde{x}_k}\}\tau \circ \sigma$.

7.5 Cases studies

7.5.1 Postal ballot voting protocol

Consider again the simple "postal ballot" voting protocol (Figure 7.1). Such a protocol should be voter verifiable.

Analysis Let tests R^{IV}, R^{UV} be given as follows:

$$\begin{array}{rcl} R^{IV} & = & \mathsf{eq}(\mathsf{sign}(u, \mathsf{sdec}(x_2, z_1)), x_3) \\ & & \wedge\, \mathsf{eq}(x_1, Pk(\mathsf{sdec}(x_2, z_1))) \\ R^{UV} & = & \bigwedge_{i=1}^{n} \mathsf{eq}(\mathsf{getmsg}(x_{3,i}), u_i) \end{array}$$

and hence the variable x_1 is expected to correspond to the voter's public key published by the key distribution process; x_2 is the voter's private key distributed on a private channel; and x_3 should correspond to the voter's signed vote.

Suppose $\widetilde{VP}(1,\ldots,n)(\rightarrow^* \xrightarrow{\alpha} \rightarrow^*)^* B = \nu\, \tilde{n}.(\sigma \mid Q)$ such that B is irreducible and $\mathrm{dom}(\sigma) = \{x'_1, \ldots, x'_{3\cdot n}\}$. Without loss of generality we have:

$$\begin{array}{l} \sigma = \{^{Pk(sk_{V_1})}/_{x'_{l_1}}, \ldots, {}^{Pk(sk_{V_n})}/_{x'_{l_n}}, \\ \qquad {}^{\mathsf{senc}(sk_{V_1}, kpc_1)}/_{x'_{l_{n+1}}}, \ldots, {}^{\mathsf{senc}(sk_{V_n}, kpc_n)}/_{x'_{l_{2n}}}, \\ \qquad\qquad {}^{\mathsf{sign}(\bar{v}_1, sk_{V_1})}/_{x'_{l_{2n+1}}}, \ldots, {}^{\mathsf{sign}(\bar{v}_n, sk_{V_n})}/_{x'_{l_{3n}}}\} \end{array}$$

Let f_1, f_2, f_3 be given by $f_1(j) = l_j$, $f_2(j) = l_{n+j}$, $f_3(j) = l_{2n+j}$. It follows that:

1. Individual verifiability. We observe $R^{IV}\Phi$ is defined as $\mathsf{sign}(v', \mathsf{sdec}(\mathsf{senc}(sk_{V_{i_2}}, kpc_{i_2}), kpc_j)) = \mathsf{sign}(\bar{v}_{i_3}, sk_{V_{i_3}}) \wedge Pk(sk_{V_{i_1}}) = Pk(\mathsf{sdec}(\mathsf{senc}(sk_{V_{i_2}}, kpc_{i_2}), kpc_j))$. For the $\Rightarrow$ implication suppose $R^{IV}\Phi$ holds. It must be the case that $i_1 = i_2 = i_3 = j$ and $v' = \bar{v}_j$. For the $\Leftarrow$ implication suppose $i_1 = i_2 = i_3 = j$ and $v' = \bar{v}_j$. The result follows immediately.

2. Universal verifiability. For the $\Rightarrow$ implication suppose $R^{UV}\Phi$ holds. We observe $R^{UV}\Phi = \bigwedge_{i=1}^{n} \mathsf{eq}(\mathsf{getmsg}(\mathsf{sign}(\bar{v}_i, sk_{V_i})), v'_i) =_E \bigwedge_{i=1}^{n} \mathsf{eq}(\bar{v}_i, v'_i)$ and hence it must be the case that $\bar{v}_1 = v'_1 \wedge \ldots \wedge \bar{v}_n = v'_n$. It follows immediately that $\tilde{v}' = (\bar{v}_1, \ldots, \bar{v}_n)$. For the $\Leftarrow$ implication suppose $\tilde{v}' = (\bar{v}_1, \ldots, \bar{v}_n)$. We have $R^{UV}\Phi = \bigwedge_{i=1}^{n} \mathsf{eq}(\mathsf{getmsg}(\mathsf{sign}(\bar{v}_i, sk_{V_i})), v'_i) =_E \bigwedge_{i=1}^{n} \mathsf{eq}(\bar{v}_i, v'_i)$, which holds by our hypothesis.

Moreover, by inspection of the voting protocol specification we have $\widetilde{VP}(\{^{\bar{v}_1}/_u\}, \ldots, \{^{\bar{v}_n}/_u\})(\rightarrow^* \xrightarrow{\alpha} \rightarrow^*)^* \varphi$ and $\mathrm{dom}(\varphi) = \{x'_1, \ldots, x'_{3\cdot n}\}$.

7.5.2 Traditional ballot box voting example

Our second protocol is similar to the traditional ballot box voting system. A voter selects her vote and completes the ballot paper accordingly; she then places the ballot paper anonymously in the ballot box, without recording her name or any other identification information. This system can be transformed into an electronic system by publishing a vote $\bar{v}_i$ on the bulletin board. The specification of this protocol is defined by $\langle V, 0, (), (), () \rangle$ where $V \triangleq \overline{bb}\langle u \rangle$. Hence we have that

$$\widetilde{VP}(\{\bar{v}_1/u\}, \ldots, \{\bar{v}_n/u\}) = \overline{bb}\langle \bar{v}_1 \rangle \mid \ldots \mid \overline{bb}\langle \bar{v}_n \rangle$$

Such a protocol should obviously not be individually verifiable. We will show that our definition is indeed violated. The set of irreducible processes $\{B \mid VP(\rightarrow^* \xrightarrow{\alpha} \rightarrow^*)^* B\}$ consists of the set of frames

$$\{\sigma_p = \{\bar{v}_{p(1)}/x'_1, \ldots, \bar{v}_{p(n)}/x'_n\} \mid \\ p \text{ is a permutation on } \{1, \ldots, n\}\}$$

We suppose that there exist a, b such that $a \neq b$ and $\bar{v}_a = \bar{v}_b$, *i.e.* two voters cast the same vote. Let $k \cdot n$ be the number of $\overline{c}\langle M \rangle$ in $\widetilde{VP}(\{\bar{v}_1/u\}, \ldots, \{\bar{v}_n/u\})$. It follows that $k = 1$ and we simply write f instead of f_1. By contradiction, suppose for all σ_p that there exists f and R^{IV} such that for all i, j and v' we have $R^{IV}\{v'/u, x'_{f(i)}/x\}\sigma_p$ iff $i = j \wedge v' = \bar{v}_j$. However as $\bar{v}_a = \bar{v}_b$ there exist i, j such that $x'_{f(i)}\sigma_p = x'_{f(j)}\sigma_p$. Hence, we obtain $R^{IV}\{v'/u, x'_{f(i)}/x\}\sigma_p$ and $i \neq j \wedge v' = \bar{v}_j$ which contradicts the hypothesis.

7.5.3 Protocol due to Fujioka, Okamoto & Ohta

Description The protocol [115] involves voters, an administrator and a collector. The voter computes her ballot $b = \mathsf{commit}(\bar{v}, r)$ and sends the signature $\mathsf{sign}(\mathsf{blind}(b, r'), sk_V)$ with her blind ballot $\mathsf{blind}(b, r')$ to the administrator. The administrator verifies the signature, signs this blind ballot $\mathsf{sign}(\mathsf{blind}(b, r'), sk_A)$ and returns it. The voter verifies the administrator's signature and unblinds the received message to recover the signature of her ballot by the administrator $\mathsf{sign}(b, sk_A)$. The voter then posts her ballot b and the administrator's signature of her ballot $\mathsf{sign}(b, sk_A)$ to the bulletin board. The collector receives $(b, \mathsf{sign}(b, sk_A))$ from the bulletin board, verifies the signature and if the verification succeeds, places $(l, b, \mathsf{sign}(b, sk_A))$ on the bulletin board. The voter checks the bulletin board for her entry, discovers l and posts on the bulletin board l and r. Finally, the collector opens all of ballots on the bulletin board using the corresponding r's. To capture the protocol we consider the equational theory E defined by:

$$
\begin{array}{lcl}
\mathsf{sdec}(\mathsf{senc}(x,y),y) & = & x \\
\mathsf{open}(\mathsf{commit}(x,y),y) & = & x \\
\mathsf{checksign}(\mathsf{sign}(x,y),x,Pk(y)) & = & \mathsf{true} \\
\mathsf{unblind}(\mathsf{blind}(x,y),y) & = & x \\
\mathsf{unblind}(\mathsf{sign}(\mathsf{blind}(x,y),z),y) & = & \mathsf{sign}(x,z) \\
\mathsf{nth}_i^j((x_1,\ldots,x_j)) & = & x_i \text{ if } i \leq j.
\end{array}
$$

Applied pi formalism The voting specification of this protocol is represented by $\langle \mathtt{voter}, \mathtt{keying}, \tilde{s}, \tilde{t}, \tilde{m} \rangle$ where $\tilde{s} = (r, r')$, $\tilde{t} = (sk_V)$, $\tilde{m} = (a)$. The `voter` and `keying` processes are defined below.

$$
\begin{array}{lcl}
\mathtt{voter} & \triangleq & a(sk_V, pk_A) \\
& & \mathsf{let}\ b = \mathsf{commit}(u,r)\ \mathsf{in} \\
& & \overline{c}\langle Pk(sk_V), \mathsf{blind}(b,r'), \mathsf{sign}(\mathsf{blind}(b,r'), sk_V)\rangle \\
& & c(x) \\
& & \mathsf{if}\ \mathsf{checksign}(x, \mathsf{blind}(b,r'), pk_A) = true\ \mathsf{then} \\
& & \mathsf{let}\ sb = \mathsf{unblind}(x,r')\ \mathsf{in} \\
& & \overline{bb}\langle b, sb\rangle \\
& & bb(l,y,z) \\
& & \mathsf{if}\ y = b \wedge z = sb\ \mathsf{then} \\
& & \overline{bb}\langle l, r\rangle \\
\\
\mathtt{keying} & \triangleq & \overline{a}\langle sk_V, Pk(sk_A)\rangle \mid \overline{bb}\langle Pk(sk_V), Pk(sk_A)\rangle
\end{array}
$$

Analysis Let tests R^{IV} and R^{UV} be given in Figure 7.2. R^{IV} expects that x_1 corresponds to the public keys published by the keying authority; x_2 corresponds to the private/public keys sent to the voter by the keying authority using a private channel; and x_3 is the voter's signed blinded ballot. The variable x_4 should correspond to the voter's blinded ballot signed by the administrator; and x_5 is the unblinded signed ballot. Finally, x_7 is expected to refer to the commitment factor used during the protocol. The test R^{IV} ensures that all values are provided as expected and R^{UV} checks that opening the ballots reveals the votes corresponding to the published outcome.

Suppose $\widetilde{VP}(1,\ldots,n)(\rightarrow^* \xrightarrow{\alpha} \rightarrow^*)^* B$ such that B is irreducible, $\phi(B) = \nu\, \tilde{n}.\sigma$, $\mathrm{dom}(\sigma) = \{x'_1, \ldots, x'_{7\cdot n}\}$ and σ is as defined in Figure 7.2. Let $f_1, \ldots, f_7$ be given by $f_i(j) = l_{(i-1)\cdot n + j}$. It follows that:

1. Individual verifiability. The result follows immediately since $R^{IV}\Phi$ has a single solution for $i_1, \ldots, i_7, j, v'$ namely $i_1 = \ldots = i_7 = j$ and $v' = \bar{v}_j$. The result of $R^{IV}\Phi$ is provided in Figure 7.2.

2. Universal verifiability. We observe that $R^{UV}\Phi = \bigwedge_{i=1}^{n} \mathsf{eq}(\mathsf{open}(\mathsf{nth}_2^3(\langle l_i, \mathsf{commit}(\bar{v}_i, r_i), \mathsf{sign}(\mathsf{commit}(\bar{v}_i, r_i), sk_A)\rangle), r_i), v_i') =_E \bigwedge_{i=1}^{n} \mathsf{eq}(\mathsf{open}(\mathsf{commit}(\bar{v}_i, r_i), r_i), v_i') =_E \mathsf{eq}(\bar{v}_i, v_i')$ and hence has a single solution for $v_1', \ldots, v_n'$ and namely $\tilde{v}' = (\bar{v}_1, \ldots, \bar{v}_n)$, concluding our proof.

Moreover, by inspection of the voting process specification, we have $\widetilde{VP}(\{\bar{v}_1/_u\}, \ldots, \{\bar{v}_n/_u\})(\rightarrow^* \xrightarrow{\alpha} \rightarrow^*)^* B$ and $\mathrm{dom}(\sigma) = \{x_1', \ldots, x_{7 \cdot n}'\}$.

7.5.4 Protocol due to Lee *et al.*

Description The protocol [141] involves an administrator, voters, mixers, talliers and a trusted randomisation service (in practice the randomisation service is implemented as a secure smart card called a tamper resistant randomiser). The voter encrypts her ballot and sends it to the randomisation service using a private channel. The randomisation service re-encrypts the ballot and returns the signed re-encrypted ballot along with a designated verifier proof which demonstrates the re-encrypted ballot is indeed a re-encryption of the voter's encrypted ballot. The additional randomisation ensures the voter cannot reconstruct her ballot and hence is unable to create a receipt for a potential coercer. The voter signs her ballot and posts it on the bulletin board. The administrator verifies the double signed ballots and publishes valid ballots on the bulletin board. The mixers then perform a secret shuffle. Finally the talliers use signature proofs of knowledge to reveal an (t, n)-threshold decryption key for each ballot.

Applied pi formalism The voting specification of this protocol is defined as $\langle \mathtt{voter}, \mathtt{registrar}, \tilde{s}, \tilde{t}, \tilde{m}\rangle$ where $\tilde{s} = (r), \tilde{t} = (sk_V, sk_R), \tilde{m} = (a_{\mathrm{V}})$ and processes `voter`,`registrar` are defined below.

$$\begin{aligned}\mathtt{voter} \triangleq\ & a_{\mathrm{V}}(sk_V, pk_R, pk_T)\\ & \text{let } b = \mathsf{penc}(u, r, pk_T) \text{ in}\\ & \overline{a_{\mathrm{VR}}}\langle b\rangle.a_{\mathrm{VR}}(sb', dvp)\\ & \text{if } \mathsf{checkdvp}(dvp, b, \mathsf{getmsg}(sb'), Pk(sk_V)) \text{ then}\\ & \text{if } \mathsf{checksign}(sb', pk_R) \text{ then}\\ & \overline{bb}\langle \mathsf{sign}(sb', sk_V)\rangle\end{aligned}$$

$$\begin{aligned}\mathtt{registrar} \triangleq\ & \overline{a_{\mathrm{V}}}\langle sk_V, Pk(sk_R), Pk(sk_T)\rangle \mid \overline{a_{\mathrm{R}}}\langle sk_R\rangle \mid\\ & \overline{bb}\langle\langle Pk(sk_V), Pk(sk_R), Pk(sk_T)\rangle\rangle\end{aligned}$$

For simplicity we consider $(1, n)$-threshold decryption. In addition we assume the existence of a secure mixer and hence do not verify the shuffle.

In [141] it is suggested that the protocol makes use of the ElGamal encryption scheme which we model by the following equations for decryption and

re-encryption.

$$\begin{array}{rcl} \mathsf{dec}(\mathsf{penc}(x,y,Pk(z)),z) & = & x \\ \mathsf{renc}(\mathsf{penc}(x,y,z),w) & = & \mathsf{penc}(x,f(y,w),z) \end{array}$$

The ElGamal encryption scheme exhibits the feature expressed by the equation

$$\mathsf{dec}(\mathsf{penc}(x,y,Pk(z)),\mathsf{commit}(\mathsf{penc}(x,y,Pk(z)),z)) = x$$

which is used by the protocol. We also add functions $\mathsf{dvp}, \mathsf{checkdvp}$ to model *designated verifier proofs* of the fact that a message is a re-encryption of another one; we adopt the equations

$$\begin{array}{rcl} \mathsf{checkdvp}(\mathsf{dvp}(x,renc(x,y),y,pk(z)), & & \\ \qquad x,renc(x,y),pk(z)) & = & \mathsf{true} \\ \mathsf{checkdvp}(\mathsf{dvp}(x,y,z,w),x,y,pk(w)) & = & \mathsf{true} \end{array}$$

The second equation models that checkdvp also succeeds for a *fake proof* constructed using the designated verifier's private key. By a slight abuse of notation we also interpret $\mathsf{checkdvp}(t_1,t_2,t_3,t_4)$ as a predicate which evaluates to true whenever $\mathsf{checkdvp}(t_1,t_2,t_3,t_4) =_E \mathsf{true}$.

We adopt the formalism for signature proofs of knowledge due to Backes *et al.* [25]. A signature proof of knowledge is a term $\mathsf{SPK}_{i,j}(\tilde{M},\tilde{N},F)$ where $\tilde{M} = (M_1,\ldots,M_i)$ denotes the witness (or private component), $\tilde{N} = (N_1,\ldots,N_j)$ defines the public parameters and F is a formula over those terms. More precisely F is a term without names or variables, but includes distinguished constants α_k, β_l where $k,l \in \mathbb{N}$. The constants α_k, β_l in F denote placeholders for the terms $M_k \in \tilde{M}$, $N_l \in \tilde{N}$ used within a signature of knowledge $\mathsf{SPK}_{i,j}(\tilde{M},\tilde{N},F)$. For example the signature proof of knowledge used by the Lee *et al.* voting protocol [141] demonstrates possession of a secret key sk_T such that $Pk(sk_T) = pk_T$ and $d = \mathsf{commit}(b',sk_T)$ i.e. the proof shows the public key pk_T is correctly formed and d is a decryption key for the voter's ballot b'. This can be captured as $\mathsf{SPK}_{1,3}((sk_T),(pk_T,\mathsf{commit}(b',sk_T),b'),F_L)$ where $F_L = \mathsf{eq}(\beta_1,Pk(\alpha_1)) \wedge \mathsf{eq}(\beta_2,\mathsf{commit}(\beta_3,\alpha_1))$. A term $\mathsf{SPK}_{i,j}(\tilde{M},\tilde{N},F)$ represents a valid signature if the term obtained by substituting M_k, N_l for the corresponding α_k, β_l evaluates to true. Verification of such a statement is modelled by the function $\mathsf{Ver}_{i,j}$. The equational theory includes the following equations defined over all tuples $\tilde{x} = (x_1,\ldots,x_i), \tilde{y} = (y_1,\ldots,y_j)$ and formula $F \in T_{\Sigma\cup\{\alpha_k,\beta_l|k\leq i,l\leq j\}}$ without names or variables:

$$\begin{array}{rcl} \mathsf{Public_p}(\mathsf{SPK}_{i,j}(\tilde{x},\tilde{y},F)) & = & \mathsf{nth}_p^j(\tilde{y}) \text{ where } p \in [i,j] \\ \mathsf{Formula}(\mathsf{SPK}_{i,j}(\tilde{x},\tilde{y},F)) & = & F \end{array}$$

We also make use of the predicate $\mathsf{Ver}_{i,j}$ defined as $\mathsf{Ver}_{i,j}(F,\mathsf{SPK}_{i,j}(\tilde{M},\tilde{N},F'))$ if and only if $F =_E F'$ and $F\{^{M_1}/_{\alpha_1},\ldots,^{M_i}/_{\alpha_i},^{N_1}/_{\beta_1},\ldots,^{N_j}/_{\beta_j}\}$ holds where $i = |\tilde{M}|, j = |\tilde{N}|$ and $F,F' \in T_{\Sigma\cup\{\alpha_k,\beta_l|k\leq i,l\leq j\}}$ without names or variables.

Analysis Let tests R^{IV}, R^{UV} be given as in Figure 7.3. The tests make the following assumptions. The variable x_1 corresponds to the public/secret keys sent to the voter by the key distribution process; x_2 is the voter's randomised ballot; and x_3 is the signed ballot produced by the randomisation service along with a designated verifier proof demonstrating the correctness of the re-encryption. The value x_4 is the voter's double signed ballot and finally x_5 corresponds to the public keys published by the keying process. Additionally R^{UV} assumes that for all $i \in \{1, \ldots, n\}$ we have $y_{1,i}$ corresponds to the ballot recovered from the double signed ballot produced by the voter; $y_{2,i}$ is a signature proof of knowledge revealing the ballot's decryption key; and finally $y_{3,i}$ is the decryted ballot (i.e. the ith voter's vote).

Suppose $\widetilde{VP}(1, \ldots, n)(\rightarrow^* \xrightarrow{\alpha} \rightarrow^*)^* \nu\, \tilde{n}.(\sigma \mid Q)$ such that Q is irreducible and $\mathrm{dom}(\sigma) = \{x'_1, \ldots, x'_{6 \cdot n}\}$. Without loss of generality we have σ as specified in Figure 7.3. Let $\tilde{f}$ be given by $f_i(j) = l_{(i-1)\cdot n+j}$. It follows that:

1. Individual verifiability. The expansion of $R^{IV}\Phi$ is provided in Figure 7.3. The result follows immediately since $R^{IV}\Phi$ has a single solution for $i_1, \ldots, i_6, j$ namely $i_1 = i_2 = \ldots = i_6 = j$ and $v' = \bar{v}_j$.

2. Universal verifiability. We observe $R^{UV}\{\tilde{v}'/\tilde{u}, \tilde{x}'_{f_1}/\tilde{x}_1, \ldots, \tilde{x}'_{f_k}/\tilde{x}_k\}\sigma$ evaluates to the following for all $i \in \{1, \ldots, n\}$:

$$y_{1,i} = b'_i \wedge \mathsf{Ver}_{1,3}(F_L, y_{2,i}) \wedge y_{1,i} = \mathsf{Public}_3(y_{2,i}) \wedge Pk(sk_T) = \mathsf{Public}_1(y_{2,i}) \wedge y_{3,i} = \mathsf{dec}(y_{1,i}, \mathsf{Public}_2(y_{2,i})) \wedge v'_i = y_{3,i}$$

   where $b'_i = \mathsf{penc}(\bar{v}_i, f(r_i, r'_i), Pk(sk_T))$. It follows that $R^{UV}\Phi$ holds when $\tilde{v}' = (\bar{v}_1, \ldots, \bar{v}_n)$ and

$$\tau \quad =_E \quad \{b'_1/y_{1,1}, \ldots, b'_n/y_{1,n}, s_1/y_{2,1}, \ldots, s_n/y_{2,n} s^{\bar{v}_1}/y_{3,1}, \ldots, \bar{v}_n/y_{3,n}\}$$

   with $s_i = \mathsf{SPK}_{1,3}((sk_T), (pk_T, \mathsf{commit}(b'_i, sk_T), b'_i), F_L)$, concluding our proof.

Moreover, we have $\widetilde{VP}(\{\bar{v}_1/u\}, \ldots, \{\bar{v}_n/u\})(\rightarrow^* \xrightarrow{\alpha} \rightarrow^*)^* \varphi$ such that $\mathrm{dom}(\varphi) = \{x'_1, \ldots, x'_{6 \cdot n}\}$.

7.6 Relates works

The literature is rich in works dealing with formal verification of security protocols. However, there are only few formal works [30, 77, 97, 196, 136, 149] related to electronic voting protocols. This is mainly due to their lack of maturity compared to other ones such as key distribution or authentication protocols, and to

the complexity of the involved techniques. Indeed, they involve advanced cryptographic primitives, for example bit commitment, blind signature, zero knowledge proofs, and rely on complex channels, for example anonymous channels. Eligibility, fairness, receipt-freeness, coercion-resistance, vote-privacy, inalterability and declared tally have been studied in [97, 136, 30, 25].

In this chapter, we analyse the individual and universal verifiability properties. Juels , Catalano and Jacobson [126, 127] presented the first definition of universal verifiability in terms of game semantics in the provable security model. Their definition assumes voting protocols produce signature proofs of knowledge demonstrating the correctness of tallying. Automated analysis is not discussed.

In the context of formal methods, Chevallier-Mames et al. [76] provide the first formalisation of universal verifiability. However, their definition is incompatible with vote-privacy, and hence with coercion-resistance. To see this, note that they require functions f and f' such that, for any bulletin board bb and list of eligible voters L, $f(bb, L)$ returns the list of actual voters, and $f'(bb, L)$ returns the tally (see Definition 1 of [76]). From these functions, one could consider any single bulletin board entry b and compute $f(\{b\}, L)$ and $f'(\{b\}, L)$ to reveal a voter and her vote.

Baskar, Ramanujan & Suresh [30] and subsequently Talbi et al. [193] have formalised verifiability with respect to the FOO [115] electronic voting protocol. Their definitions are tightly coupled to that particular protocol and cannot easily be generalised to other protocols. Moreover, their definitions characterise individual executions as verifiable or not; whereas verifiability properties should be considered with respect to every execution (i.e. the entire protocol).

7.7 Conclusion

In this chapter, we have formally defined the properties of *individual and universal verifiability* for electronic voting protocols, and show that they are satisfied by the protocols due to Fujioka, Okamoto & Ohta [115] and Lee *et al.* [141]. Since the second of these protocols also satisfies vote-privacy, receipt-freeness and coercion-resistance [97], our definition is compatible with those properties, in contrast with the definition of [76]. Our definition represents a sufficient condition, but not a necessary one.

A more recent version of this work, in which, in addition to the formalisation of *individual and universal verifiability*, we formalise the *eligibility verifiability* will be presented at WISSec 2009 workshop [191].

Since the *voter verifiability properties* can be seen as *equivalence properties*, we plan, in future works, to extend the decidability results obtained on the saturated systems in order to prove such equivalence properties.

Figure 7.1 Postal ballot voting protocol

The voting specification of our *"postal ballot"* voting protocol is defined as $\langle V, P, (), (sk_V), (a)\rangle$ for processes:

$$\begin{array}{rcl} V & \triangleq & a(sk_V).\overline{bb}\langle \mathsf{sign}(u, sk_V)\rangle \\ P & \triangleq & \overline{a}\langle sk_V\rangle \mid \overline{bb}\langle Pk(sk_V)\rangle \end{array}$$

The process $\widetilde{VP}(\{\bar{v}_1/_u\}, \{\bar{v}_2/_u\})$ is defined as:

$$\begin{array}{l} \widetilde{VP}(\{\bar{v}_1/_u\}, \{\bar{v}_2/_u\}) \triangleq \nu\, a_1, sk_{V_1}, kpc_1, a_2, sk_{V_2}, kpc_2. \\ \qquad (\overline{a_1}\langle sk_{V_1}\rangle \mid \overline{bb}\langle Pk(sk_{V_1})\rangle \mid \overline{a_2}\langle sk_{V_2}\rangle \mid \overline{bb}\langle Pk(sk_{V_1})\rangle \mid \\ \qquad\quad a_1(sk_V).\overline{bb}\langle \mathsf{senc}(sk_V, kpc_1)\rangle.\overline{bb}\langle \mathsf{sign}(\bar{v}_1, sk_V)\rangle \mid \\ \qquad\qquad a_2(sk_V).\overline{bb}\langle \mathsf{senc}(sk_V, kpc_1)\rangle.\overline{bb}\langle \mathsf{sign}(\bar{v}_2, sk_V)\rangle) \end{array}$$

Figure 7.2 Fujioka *et al.* protocol verification artifacts

$$\begin{array}{rcl} R^{IV} & = & \mathsf{eq}(Pk(\mathsf{nth}_1^2(\mathsf{sdec}(x_2, z_3))), \mathsf{nth}_1^2(x_1)) \wedge \mathsf{eq}(\mathsf{nth}_2^3(x_3), \mathsf{blind}(\mathsf{commit}(v, z_1), z_2)) \wedge \\ & & \mathsf{eq}(\mathsf{checksign}(x_4, \mathsf{blind}(\mathsf{commit}(v, z_1), z_2), \mathsf{nth}_2^2(x_1)), true) \wedge \\ & & \mathsf{eq}(\mathsf{nth}_1^2(x_5), \mathsf{nth}_2^3(x_6)) \wedge \mathsf{eq}(\mathsf{nth}_1^2(x_5), \mathsf{commit}(v, z_1)) \wedge \mathsf{eq}(\mathsf{nth}_2^2(x_7), z_1) \\ R^{UV} & = & \bigwedge_{i=1}^{n} \mathsf{eq}(\mathsf{open}(\mathsf{nth}_2^3(x_{6,i}), \mathsf{nth}_2^2(x_{7,i})), v_i) \end{array}$$

$$\begin{array}{l} \sigma = \{(Pk(sk_{V_1}), Pk(sk_A))/_{x'_{l_1}}, \ldots, (Pk(sk_{V_n}), Pk(sk_A))/_{x'_{l_n}}, \\ \qquad \mathsf{senc}((sk_{V_1}, Pk(sk_A)), Kpc_1)/_{x'_{l_{n+1}}}, \ldots, \mathsf{senc}((sk_{V_n}, Pk(sk_A)), Kpc_n)/_{x'_{l_{2n}}}, \\ (Pk(sk_{V_1}), \mathsf{blind}(b_1, r'_1), \mathsf{sign}(\mathsf{blind}(b_1, r'_1), sk_{V_1}))/_{x'_{l_{2n+1}}}, \ldots, (Pk(sk_{V_n}), \mathsf{blind}(b_n, r'_n), \mathsf{sign}(\mathsf{blind}(b_n, r'_n), sk_{V_n}))/_{x'_{l_{3n}}}, \\ \mathsf{sign}(\mathsf{blind}(b_1, r'_1), sk_A)/_{x'_{l_{3n+1}}}, \ldots, \mathsf{sign}(\mathsf{blind}(b_n, r'_n), sk_A)/_{x'_{l_{4n}}}, (b_1, \mathsf{sign}(b_1, sk_A))/_{x'_{l_{4n+1}}}, \ldots, (b_n, \mathsf{sign}(b_n, sk_A))/_{x'_{l_{5n}}}, \\ \qquad (l_1, b_1, \mathsf{sign}(b_1, sk_A))/_{x'_{l_{5n+1}}}, \ldots, (l_n, b_n, \mathsf{sign}(b_n, sk_A))/_{x'_{l_{6n}}}, (l_1, r_1)/_{x'_{l_{6n+1}}}, \ldots, (l_n, r_n)/_{x'_{l_{7n}}}\} \\ \qquad\qquad \text{where } b_i = \mathsf{commit}(\bar{v}_i, r_i) \end{array}$$

$$\begin{array}{rcl} R^{IV}\Phi & = & \mathsf{eq}(Pk(\mathsf{nth}_1^2(\mathsf{sdec}(\mathsf{senc}((sk_{V_{i_2}}, Pk(sk_A)), Kpc_{i_2}), Kpc_j))), Pk(sk_{V_{i_1}})) \wedge \\ & & \mathsf{eq}(\mathsf{blind}(\mathsf{commit}(\bar{v}_{i_3}, r_{i_3}), r'_{i_3}), \mathsf{blind}(\mathsf{commit}(u, r_j), r'_j)) \wedge \\ & & \mathsf{eq}(\mathsf{checksign}(\mathsf{sign}(\mathsf{blind}(\mathsf{commit}(\bar{v}_{i_4}, r_{i_4}), r'_{i_4}), sk_A), \\ & & \mathsf{blind}(\mathsf{commit}(u, r_j), r'_j), Pk(sk_A)), true) \wedge \\ & & \mathsf{eq}(\mathsf{commit}(\bar{v}_{i_5}, r_{i_5}), \mathsf{commit}(\bar{v}_{i_6}, r_{i_6})) \wedge \\ & & \mathsf{eq}(\mathsf{commit}(\bar{v}_{i_5}, r_{i_5}), \mathsf{commit}(u, r_j)) \wedge \mathsf{eq}(r_{i_7}, r_j). \end{array}$$

Figure 7.3 Lee *et al.* protocol verification artifacts

$$
\begin{aligned}
R^{IV} \;=\;& \mathsf{eq}(x_2, \mathsf{penc}(u, z_1, \mathsf{nth}_3^3(\mathsf{sdec}(x_1, z_2)))) \wedge \mathsf{eq}(x_4, \mathsf{sign}(\mathsf{nth}_1^2(\mathsf{sdec}(x_3, z_2)),\\
& \mathsf{nth}_1^3(\mathsf{sdec}(x_1, z_2))))\wedge\\
& \mathsf{checkdvp}(\mathsf{nth}_2^2(\mathsf{sdec}(x_3, z_2)), x_2, \mathsf{getmsg}(\mathsf{nth}_1^2(\mathsf{sdec}(x_3, z_2))), Pk(\mathsf{nth}_1^3(\mathsf{sdec}(x_1, z_2))))\wedge\\
& \mathsf{eq}(\mathsf{nth}_1^3(x_6), Pk(\mathsf{nth}_1^3(\mathsf{sdec}(x_1, z_2)))) \wedge \mathsf{eq}(Pk(x_5), \mathsf{nth}_2^3(\mathsf{sdec}(x_1, z_2)))\\
R^{UV} \;=\;& \textstyle\bigwedge_{i=1}^{n} \Big(y_{1,i} = \mathsf{getmsg}(\mathsf{getmsg}(x_{4,i})) \wedge \mathsf{nth}_3^3(x_{6,i}) = \mathsf{Public}_1(y_{2,i}) \wedge y_{1,i} = \mathsf{Public}_3(y_{2,i})\wedge\\
& \mathsf{Ver}_{1,3}(F_L, y_{2,i}) \wedge y_{3,i} = \mathsf{dec}(y_{1,i}, \mathsf{Public}_2(y_{2,i})) \wedge u_i = y_{3,i}\Big)
\end{aligned}
$$

$$
\begin{aligned}
\sigma =_E \{&\mathsf{senc}(\langle sk_{V_1}, Pk(sk_{R_1}), Pk(sk_T)\rangle, kpc_1)/_{x'_{l_1}}, \ldots, \mathsf{senc}(\langle sk_{V_n}, Pk(sk_{R_n}), Pk(sk_T)\rangle, kpc_n)/_{x'_{l_n}},\\
& b_1/_{x'_{l_{n+1}}}, \ldots, b_n/_{x'_{l_{2n}}}\\
& \mathsf{senc}(\langle \mathsf{sign}(b'_1, sk_{R_1}), \mathsf{dvp}(b_1, b'_1, r'_1, Pk(sk_{V_1}))\rangle, kpc_1)/_{x'_{l_{2n+1}}}, \ldots,\\
& \mathsf{senc}(\langle \mathsf{sign}(b'_n, sk_{R_n}), \mathsf{dvp}(b_n, b'_n, r'_n, Pk(sk_{V_n}))\rangle, kpc_n)/_{x'_{l_{3n}}},\\
& \mathsf{sign}(\mathsf{sign}(b'_1, sk_{R_1}), sk_{V_1})/_{x'_{l_{3n+1}}}, \ldots, \mathsf{sign}(\mathsf{sign}(b'_n, sk_{R_n}), sk_{V_n})/_{x'_{l_{4n}}},\\
& sk_{R_1}/_{x'_{l_{4n+1}}}, \ldots, sk_{R_n}/_{x'_{l_{5n}}},\\
& \langle Pk(sk_{V_1}), Pk(sk_{R_1}), Pk(sk_T)\rangle/_{x'_{l_{5n+1}}}, \ldots, \langle Pk(sk_{V_n}), Pk(sk_{R_n}), Pk(sk_T)\rangle/_{x'_{l_{6n}}}\}\\
& \text{where } b_i = \mathsf{penc}(\bar{v}_i, r_i, Pk(sk_T)) \text{ and } b'_i = \mathsf{penc}(\bar{v}_i, f(r_i, r'_i), Pk(sk_T))
\end{aligned}
$$

$$
\begin{aligned}
R^{IV}\Phi \;=\;& \mathsf{eq}(b_{i_2}, \mathsf{penc}(v', r_j, \mathsf{nth}_3^3(\mathsf{sdec}(\mathsf{senc}(\langle sk_{V_{i_1}}, Pk(sk_{R_{i_1}}), Pk(sk_T)\rangle, kpc_{i_1}), kpc_j))))\\
& \wedge\mathsf{eq}(\mathsf{sign}(\mathsf{sign}(b'_{i_4}, sk_{R_{i_4}}), sk_{V_{i_4}}), \mathsf{sign}(\mathsf{nth}_1^2(\mathsf{sdec}(\mathsf{senc}(\langle \mathsf{sign}(b'_{i_3}, sk_{R_{i_3}}),\\
& \mathsf{dvp}(b_{i_3}, b'_{i_3}, r'_{i_3}, Pk(sk_{V_{i_3}}))\rangle, kpc_{i_3}), kpc_j)),\\
& \qquad \mathsf{nth}_1^3(\mathsf{sdec}(\mathsf{senc}(\langle sk_{V_{i_1}}, Pk(sk_{R_{i_1}}), Pk(sk_T)\rangle, kpc_{i_1}), kpc_j))))\\
& \wedge\mathsf{checkdvp}(\mathsf{nth}_2^2(\mathsf{sdec}(\mathsf{senc}(\langle \mathsf{sign}(b'_{i_3}, sk_{R_{i_3}}),\\
& \mathsf{dvp}(b_{i_3}, b'_{i_3}, r'_{i_3}, Pk(sk_{V_{i_3}}))\rangle, kpc_{i_3}), kpc_j)), b_{i_2},\\
& \mathsf{getmsg}(\mathsf{nth}_1^2(\mathsf{sdec}(\mathsf{senc}(\langle \mathsf{sign}(b'_{i_3}, sk_{R_{i_3}}), \mathsf{dvp}(b_{i_3}, b'_{i_3}, r'_{i_3}, Pk(sk_{V_{i_3}}))\rangle, kpc_{i_3}), kpc_j))),\\
& Pk(\mathsf{nth}_1^3(\mathsf{sdec}(\mathsf{senc}(\langle sk_{V_{i_1}}, Pk(sk_{R_{i_1}}),\\
& Pk(sk_T)\rangle, kpc_{i_1}), kpc_j))))\\
& \wedge\mathsf{eq}(\mathsf{nth}_1^3(\langle Pk(sk_{V_{i_6}}), Pk(sk_{R_{i_6}}), Pk(sk_T)\rangle), Pk(\mathsf{nth}_1^3(\mathsf{sdec}(\mathsf{senc}(\langle sk_{V_{i_1}}, Pk(sk_{R_{i_1}}),\\
& Pk(sk_T)\rangle, kpc_{i_1}), kpc_j))))\\
& \wedge\mathsf{eq}(Pk(sk_{R_{i_5}}), \mathsf{nth}_2^3(\mathsf{sdec}(\mathsf{senc}(\langle sk_{V_{i_1}}, Pk(sk_{R_{i_1}}), Pk(sk_T)\rangle, kpc_{i_1}), kpc_j)))
\end{aligned}
$$

Chapter 8

Conclusion and Perspectives

In our society, the use of electronic applications such as e-communication, e-voting, e-banking, e-commerce, *etc* is increasing. Among several important requirements, security figures as one crucial aspect. To guarantee security, such applications use cryptographic protocols. It is well-known that design of cryptographic protocols is not sufficient to their deployment, they need to be formally analysed. While the insecurity problem of cryptographic protocols has been shown to be undecidable in the general case [111], several restrictions led to decidable results with perfect and unperfect cryptography hypotheses.

In this thesis, we have relaxed the perfect cryptography hypothesis by taking into account several algebraic properties of cryptographic primitives. Following the symbolic approach (in particular the method based on the resolution of constraint solving) to analyse cryptographic protocols, we provided decision procedures for the insecurity problem of cryptographic protocols with a bounded number of sessions.

In Chapter 3, we considered the *collision vulnerability property of hash functions*, and we analysed the class of cryptographic protocols employing hash functions having this property. We reduced the insecurity problem of our class of cryptographic protocols to the ordered satisfiability problem for the intruder exploiting the collision vulnerability property of hash functions. We provided sufficient arguments that allowed us to conjecture that, following [74], the ordered satisfiability problem for the intruder exploiting the collision vulnerability property of hash functions can be reduced to the ordered satisfiability problem for an intruder operating on words. We then proved the decidability of the last problem. A natural extension of this work would be to prove the above conjecture.

In Chapter 4, we considered the *destructive exclusive ownership vulnerability* and the *constructive exclusive ownership vulnerability properties for digital signature schemes*, and we showed the decidability of the insecurity problem for the two classes of cryptographic protocols using signature schemes having respectively

these two properties. We first reduced the insecurity problem of our two classes of protocols to the reachability problem for our respective intruder deduction systems. We then provided a decision procedure, based on the *saturation of intruder deduction systems*, for the reachability problem.

This decision procedure was then extended in Chapter 5 in order to capture a general class of cryptographic protocols: *the class of protocols using cryptographic primitives represented by convergent equational theories with finite variant property*. A simple generalisation of the decision procedure given in Chapter 4 allowed us to prove the decidability of the ground reachability problem for our class of deduction systems. Actually, we slightly modified the *saturation algorithm introduced in Chapter 4*, and showed that the termination of its application on our class of intruder deduction systems implies the decidability of the ground reachability problem. We also showed that this termination is not sufficient to prove the decidability of the general reachability problem. We then gave an additional syntactic condition on the deduction systems that allowed us to prove the decidability of the general reachability problem. As an application of our result, we gave an alternative proof for the decidability of the ground reachability problem for the blind signature theory. This decidability result has been initially proved in [9]. We also gave an alternative proof for the decidability of the general reachability problem for the subterm convergent theory, which is already proved in [31].

We note that in Chapter 5, we have only considered the class of convergent equational theories with finite variant property. We plan to extend the results of Chapter 5 to the class of AC-convergent equational theories with finite variant property. As showed in [86], such extention may include several theories such as the theory of *exclusive or* and the theory of *abelian groups*. We also plan to weaken the syntactic condition we assume on the deduction systems in order to prove the decidability of the general reachability problem.

In Chapter 6, we showed the decidability of the *ground entailment problem for a new class of clauses*. This decidability result is obtained by generalising the *saturation procedure* given in Chapter 5, and it relies on the use of *selected resolution* and on some syntactic conditions on the atom and term orderings. We also showed how to use this result in order to decide the insecurity problem of cryptographic protocols in the case of bounded number of sessions. In future works, we plan to:

1. implement the decision procedure: with respect to the result of D. Basin and H. Ganzinger [28], we remark that for ground entailment problem, it suffices to consider inferences by *selected resolution* in which one of the atoms is ground. Thus we do not need to construct the set of ground atoms before solving an entailment problem.

2. extend this result to the case where the clauses are considered modulo an

equational theory, *i.e.* we reason modulo an equational theory on atoms and terms.

3. extend the application on cryptographic protocols from search of proofs to proof of correctness, and that by including the clauses describing the protocol to the set of clauses.

4. extend the application on cryptographic protocols to the unperfect cryptography hypothesis, and that by including the clauses describing the algebraic properties of primitives and the clauses representing the congruence relation to the set of clauses.

In Chapter 7, we studied a new class of cryptographic protocols: the *electronic-voting protocols*, and in particular the *voter verifiability property*, which includes the *individual verifiability* and the *universal verifiability* properties. We formally defined these two properties using the *applied pi calculus*, and we applied our definition to the *e-voting protocol due to Fujioka, Okamoto & Ohta* [115] and the *e-voting protocol due to Lee et al.* [141]. In future works, we plan to:

1. extend the scope of e-voting protocols on which we apply our definition in order to capture more relevant protocols such as the *protocol due to Okamoto* [165], and the *protocol due to D. Sandler et al.* [182].

2. extend the decidability results obtained on the saturated systems in order to prove the voter verifiability property.

3. analyse the e-voting protocols by taking into consideration algebraic properties of cryptographic primitives.

Bibliography

[1] Logic in Computer Science. www.cs.sunysb.edu/ cse541/Spring2009/lectures.html.

[2] Security Protocols Open Repository. http://www.lsv.ens-cachan.fr/spore/.

[3] Digital signature standard. *National Institute of Standards and Technology, Federal information processing standards publication 186*, 1994.

[4] Secure hash standard. *U.S. department of commerce/National Institute of Standards and Technology, Federal information processing standards publication 180-1*, April 1995.

[5] Ansi x9.62, the elliptic curve digital signature algorithm (ecdsa), working draft. August 1998.

[6] Secure/multipurpose internet mail extensions (s/mime) version 3.1 certificate handling. *B. Ramsdell, RFC 3850*, July 2004.

[7] Secure hash standard. *U.S. department of commerce/National Institute of Standards and Technology, Federal information processing standards publication 180*, May 1993.

[8] M. Abadi, B. Blanchet, and C. Fournet. Just fast keying in the pi calculus. In *D. A. Schmidt editor, ESOP, volume 2986 of Lecture Notes in Computer Science, pages 340-354. Springer*, 2004.

[9] M. Abadi and V. Cortier. Deciding knowledge in security protocols under (many more) equational theories. In *CSFW*, pages 62–76. IEEE Computer Society, 2005.

[10] M. Abadi and V. Cortier. Deciding knowledge in security protocols under equational theories. *Theor. Comput. Sci.*, 367(1-2):2–32, 2006.

[11] M. Abadi and C. Fournet. Mobile values, new names, and secure communication. In *POPL'01: Proceedings of the 28th ACM Symposium on Principles of Programming Languages*, pages 104–115. ACM, 2001.

[12] M. Abadi and A.D. Gordon. A calculus for cryptographic protocols: The spi calculus. *Information and Computation*, pages 148(1):1–70, January 1999.

[13] M. Abadi and Ph. Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). *J. Cryptology*, 15(2):103–127, 2002.

[14] R. M. Amadio and W. Charatonik. On name generation and set-based analysis in the dolev-yao model. In L. Brim, P. Jancar, M. Kretínský, and A. Kucera, editors, *CONCUR*, volume 2421 of *Lecture Notes in Computer Science*, pages 499–514. Springer, 2002.

[15] R. M. Amadio and D. Lugiez. On the reachability problem in cryptographic protocols. In C. Palamidessi, editor, *CONCUR*, volume 1877 of *Lecture Notes in Computer Science*, pages 380–394. Springer, 2000.

[16] R. M. Amadio, D. Lugiez, and V. Vanackère. On the symbolic reduction of processes with cryptographic functions. *Theor. Comput. Sci.*, 290(1):695–740, 2003.

[17] A. Armando, D. A. Basin, M. Bouallagui, Y. Chevalier, L. Compagna, S. Mödersheim, M. Rusinowitch, M. Turuani, L. Viganò, and L. Vigneron. The aviss security protocol analysis tool. In E. Brinksma and K. Guldstrand Larsen, editors, *CAV*, volume 2404 of *Lecture Notes in Computer Science*, pages 349–353. Springer, 2002.

[18] G. Ateniese and B. de Medeiros. A provably secure nyberg-rueppel signature variant with applications. *Technical Report, In Cryptology ePrint Archive, Report 2004/93*, 2004.

[19] AVISPA Tool Set. `http://www.avispa-project.org/`.

[20] F. Baader. The theory of idempotent semigroups is of unification type zero. *J. Autom. Reasoning*, 2(3):283–286, 1986.

[21] F. Baader and T. Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.

[22] F. Baader and W. Snyder. Unification theory. In *J. A. Robinson and A. Voronkov editors, Handbook of Automated Reasoning, Elsevier and MIT Press*, pages 445–532. 2001.

[23] L. Bachmair. Canonical equational proofs. *Progress in Theoretical Computer Science, Birkhauser*, 1991.

[24] L. Bachmair and H. Ganzinger. On restrictions of ordered paramodulation with simplification. In M. E. Stickel, editor, *CADE*, volume 449 of *Lecture Notes in Computer Science*, pages 427–441. Springer, 1990.

[25] M. Backes, C. Hritcu, and M. Maffei. Automated verification of remote electronic voting protocols in the applied pi-calculus. In *CSF'08: Proceedings of the 21st IEEE Computer Security Foundations Symposium*, pages 195–209. IEEE, 2008.

[26] M. Backes and B. Pfitzmann. A cryptographically sound security proof of the needham-schroeder-lowe public-key protocol. In Pandya and Radhakrishnan [166], pages 1–12.

[27] J. Baek, K. Kim, and T. Matsumoto. On the significance of unknown key-share attacks: how to cope with them? *In proc. of Symposium on Cryptography and Information Security (SCIS 2000)*, January 2000.

[28] D. A. Basin and H. Ganzinger. Automated complexity analysis based on ordered resolution. *J. ACM*, 48(1):70–109, 2001.

[29] D. A. Basin, S. Mödersheim, and L. Viganò. An on-the-fly model-checker for security protocol analysis. In *E. Snekkenes and D. Gollmann editors, ESORICS, volume 2808 of Lecture Notes in Computer Science*, pages 253–270, Springer, 2003.

[30] A. Baskar, R. Ramanujam, and S. P. Suresh. Knowledge-based modelling of voting protocols. In *TARK'07: Proceedings of the 11th International Conference on Theoretical Aspects of Rationality and Knowledge*, pages 62–71. ACM, 2007.

[31] M. Baudet. Deciding security of protocols against off-line guessing attacks. In V. Atluri, C. Meadows, and A. Juels, editors, *ACM Conference on Computer and Communications Security*, pages 16–25. ACM, 2005.

[32] M. Baudet. *Sécurité des protocoles cryptographiques : aspects logiques et calculatoires*. Thèse de doctorat, Laboratoire Spécification et Vérification, ENS Cachan, France, 2007.

[33] M. Baudet, V. Cortier, and S. Kremer. Computationally sound implementations of equational theories against passive adversaries. In L. Caires, G. F. Italiano, L. Monteiro, C. Palamidessi, and M. Yung, editors, *ICALP*, volume 3580 of *Lecture Notes in Computer Science*, pages 652–663. Springer, 2005.

[34] Mathieu Baudet. Random polynomial-time attacks and Dolev-Yao models. In Siva Anantharaman, editor, *Proceedings of the Workshop on Security of Systems: Formalism and Tools (SASYFT'04)*, Orléans, France, June 2004.

[35] L. D. Baxter. A practically linear unification algorithm. *Reseach Report CS-76-13, Dep. of Applied Analysis and Computer Science, University of Waterloo, Waterloo, Ontario, Canada*, 1976.

[36] M. Bellare, A. Boldyreva, and S. Micali. Public-key encryption in a multi-user setting: Security proofs and improvements. In *EUROCRYPT*, pages 259–274, 2000.

[37] M. Bellare and Ph. Rogaway. Entity authentication and key distribution. In D. R. Stinson, editor, *CRYPTO*, volume 773 of *Lecture Notes in Computer Science*, pages 232–249. Springer, 1993.

[38] M. Bellare and Ph. Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In S. Vaudenay, editor, *EUROCRYPT*, volume 4004 of *Lecture Notes in Computer Science*, pages 409–426. Springer, 2006.

[39] J. Cohen Benaloh and D. Tuinstra. Receipt-free secret-ballot elections (extended abstract). In *STOC*, pages 544–553, 1994.

[40] V. Bernat and H. Comon-Lundh. Normal proofs in intruder theories. In *M. Okada and I. Satoh editors, ASIAN, volume 4435 of Lecture Notes in Computer Science*, pages 151–166, Springer, 2006.

[41] K. Bhargavan, C. Fournet, and A. D. Gordon. A semantics for web services authentication. In N. D. Jones and X. Leroy, editors, *POPL*, pages 198–209. ACM, 2004.

[42] E. Biham and R. Chen. Near-collisions of sha-0. In M. K. Franklin, editor, *CRYPTO*, volume 3152 of *Lecture Notes in Computer Science*, pages 290–305. Springer, 2004.

[43] S. Blake-Wilson, D. Johnson, and A. Menezes. Key agreement protocols and their security analysis. In M. Darnell, editor, *IMA Int. Conf.*, volume 1355 of *Lecture Notes in Computer Science*, pages 30–45. Springer, 1997.

[44] S. Blake-Wilson and A. Menezes. Authenticated diffie-hellman key agreement protocols. In *Selected Areas in Cryptography*, pages 339–361, 1998.

[45] S. Blake-Wilson and A. Menezes. Unknown key-share attacks on the station-to-station (sts) protocol. In H. Imai and Y. Zheng, editors, *Public Key Cryptography*, volume 1560 of *Lecture Notes in Computer Science*, pages 154–170. Springer, 1999.

[46] B. Blanchet. An efficient cryptographic protocol verifier based on prolog rules. In *CSFW'01: Proceedings of the 14th IEEE Computer Security Foundations Workshop*, pages 82–96. IEEE Computer Society Press, 2001.

[47] B. Blanchet. From secrecy to authenticity in security protocols. In M. V. Hermenegildo and G. Puebla, editors, *SAS*, volume 2477 of *Lecture Notes in Computer Science*, pages 342–359. Springer, 2002.

[48] B. Blanchet. Computationally sound mechanized proofs of correspondence assertions. In *CSF*, pages 97–111. IEEE Computer Society, 2007.

[49] B. Blanchet. *Vérification automatique de protocoles cryptographiques : modèle formel et modèle calculatoire.* Mémoire d'habilitation à diriger des recherches, Université Paris-Dauphine, November 2008.

[50] B. Blanchet, M. Abadi, and C. Fournet. Automated verification of selected equivalences for security protocols. In *LICS*, pages 331–340. IEEE Computer Society, 2005.

[51] Bruno Blanchet. Automatic proof of strong secrecy for security protocols. In *IEEE Symposium on Security and Privacy*, pages 86–, 2004.

[52] D. Bolignano. Towards the formal verification of electronic commerce protocols. In *CSFW*, pages 133–147. IEEE Computer Society, 1997.

[53] M. Boreale. Symbolic trace analysis of cryptographic protocols. In *F. Orejas and P. G. Spirakis and J. van Leeuwen editors, ICALP, volume 2076 of Lecture Notes in Computer Science*, pages 667–681, Springer, 2001.

[54] M. Boreale and M. G. Buscemi. A method for symbolic analysis of security protocols. *Theor. Comput. Sci.*, 338(1-3):393–425, 2005.

[55] R.S. Boyer and J.S. Moore. The sharing of structure in theorem-proving programs. *In Machine Intelligence, vol. 7, B. Meltzer and D. Michie (Eds.). Edinburgh University Press, Edinburgh, Scotland*, pages 101–116, 1972.

[56] E. Bresson, Y. Lakhnech, L. Mazaré, and B. Warinschi. A generalization of ddh with applications to protocol analysis and computational soundness. In Alfred Menezes, editor, *CRYPTO*, volume 4622 of *Lecture Notes in Computer Science*, pages 482–499. Springer, 2007.

[57] M. Burrows, M. Abadi, and R. M. Needham. A logic of authentication. *ACM Trans. Comput. Syst.*, 8(1):18–36, 1990.

[58] S. Bursuc, H. Comon-Lundh, and S. Delaune. Associative-commutative deducibility constraints. In W. Thomas and P. Weil, editors, *STACS*, volume 4393 of *Lecture Notes in Computer Science*, pages 634–645. Springer, 2007.

[59] R. Canetti and H. Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In B. Pfitzmann, editor, *EUROCRYPT*, volume 2045 of *Lecture Notes in Computer Science*, pages 453–474. Springer, 2001.

[60] U. Hustadt Ch. G. Fermüller, A. Leitsch and T. Tammet. Resolution decision procedures. In *Handbook of Automated Reasoning*, pages 1791–1849. 2001.

[61] D. Chaum. Blind signatures for untraceable payments. In *CRYPTO*, pages 199–203, 1982.

[62] D. Chaum. Elections with unconditionally-secret ballots and disruption equivalent to breaking rsa. In *EUROCRYPT*, pages 177–182, 1988.

[63] D. Chaum. Secret-ballot receipts: True voter-verifiable elections. *IEEE Security & Privacy*, 2(1):38–47, 2004.

[64] D. Chaum, C. Crépeau, and I. Damgård. Multiparty unconditionally secure protocols (extended abstract). In *STOC*, pages 11–19. ACM, 1988.

[65] D. Chaum, P. Y. A. Ryan, and S. A. Schneider. A practical voter-verifiable election scheme. In S. De Capitani di Vimercati, P. F. Syverson, and D. Gollmann, editors, *ESORICS*, volume 3679 of *Lecture Notes in Computer Science*, pages 118–139. Springer, 2005.

[66] Y. Chevalier and M. Kourjieh. Key substitution in the symbolic analysis of cryptographic protocols. In V. Arvind and S. Prasad, editors, *FSTTCS*, volume 4855 of *Lecture Notes in Computer Science*, pages 121–132. Springer, 2007.

[67] Y. Chevalier and M. Kourjieh. A symbolic intruder model for hash-collision attacks. In *M. Okada and I. Satoh editors, ASIAN, volume 4435 of Lecture Notes in Computer Science*, pages 13–27, Springer, 2006.

[68] Y. Chevalier, R. Küsters, M. Rusinowitch, and M. Turuani. Deciding the security of protocols with diffie-hellman exponentiation and products in exponents. In Pandya and Radhakrishnan [166], pages 124–135.

[69] Y. Chevalier, R. Küsters, M. Rusinowitch, and M. Turuani. Deciding the security of protocols with commuting public key encryption. *Electr. Notes Theor. Comput. Sci.*, 125(1):55–66, 2005.

[70] Y. Chevalier, R. Küsters, M. Rusinowitch, and M. Turuani. An np decision procedure for protocol insecurity with xor. *Theor. Comput. Sci.*, 338(1-3):247–274, 2005.

[71] Y. Chevalier, R. Küsters, M. Rusinowitch, M. Turuani, and L. Vigneron. Extending the dolev-yao intruder for analyzing an unbounded number of sessions. In M. Baaz and J. A. Makowsky, editors, *CSL*, volume 2803 of *Lecture Notes in Computer Science*, pages 128–141. Springer, 2003.

[72] Y. Chevalier, D. Lugiez, and M. Rusinowitch. Towards an automatic analysis of web service security. In B. Konev and F. Wolter, editors, *FroCos*, volume 4720 of *Lecture Notes in Computer Science*, pages 133–147. Springer, 2007.

[73] Y. Chevalier and M. Rusinowitch. Combining intruder theories. In *L. Caires and G. F. Italiano and L. Monteiro and C. Palamidessi and M. Yung editors, ICALP, volume 3580 of Lecture Notes in Computer Science*, pages 639–651, 2005.

[74] Y. Chevalier and M. Rusinowitch. Hierarchical combination of intruder theories. In F. Pfenning, editor, *RTA*, volume 4098 of *Lecture Notes in Computer Science*, pages 108–122. Springer, 2006.

[75] Y. Chevalier and L. Vigneron. A tool for lazy verification of security protocols. In *ASE*, pages 373–376. IEEE Computer Society, 2001.

[76] B. Chevallier-Mames, P.A. Fouque, D. Pointcheval, J. Stern, and J. Traore. On Some Incompatible Properties of Voting Schemes. In *WOTE'06: Proceedings of the International Association for Voting Systems Sciences Workshop on Trustworthy Elections*, 2006.

[77] T. Chothia, S. Orzan, J. Pang, and M. Torabi Dashti. A framework for automatically checking anonymity with *mu* crl. In U. Montanari, D. Sannella, and R. Bruni, editors, *TGC*, volume 4661 of *Lecture Notes in Computer Science*, pages 301–318. Springer, 2006.

[78] J. Clark and J. Jacob. A survey of authentication protocol literature. November 1997.

[79] E. M. Clarke, S. Jha, and W. R. Marrero. Verifying security protocols with brutus. *ACM Trans. Softw. Eng. Methodol.*, 9(4):443–487, 2000.

[80] H. Comon-Lundh. Intruder theories (ongoing work). In I. Walukiewicz, editor, *FoSSaCS*, volume 2987 of *Lecture Notes in Computer Science*, pages 1–4. Springer, 2004.

[81] H. comon Lundh and V. Cortier. Tree automata with one memory, set constraints and cryptographic protocols. *Research Report, LSV-01-13, Laboratoire Spécification et Vérification, ENS de Cachan, France*, 2001.

[82] H. Comon-Lundh and V. Cortier. Security properties: Two agents are sufficient. In P. Degano, editor, *ESOP*, volume 2618 of *Lecture Notes in Computer Science*, pages 99–113. Springer, 2003.

[83] H. Comon-Lundh and V. Cortier. Computational soundness of observational equivalence. In *CCS'08: Proceedings of the 15th ACM Conference on Computer and Communications Security*, pages 109–118. ACM Press, 2008.

[84] H. Comon-Lundh and V. Cortier. New decidability results for fragments of first-order logic and application to cryptographic protocols. In *R. Nieuwenhuis editor, RTA, volume 2709 of Lecture Notes in Computer Science*, pages 148–164, Springer, 2003.

[85] H. Comon-Lundh, V. Cortier, and J. Mitchell. Tree automata with one memory, set constraints, and ping-pong protocols. In *F. Orejas and P. G. Spirakis and J. van Leeuwen editors, ICALP, volume 2076 of Lecture Notes in Computer Science*, pages 682–693, Springer, 2001.

[86] H. Comon-Lundh and S. Delaune. The finite variant property: How to get rid of some algebraic properties. In J. Giesl, editor, *RTA*, volume 3467 of *Lecture Notes in Computer Science*, pages 294–307. Springer, 2005.

[87] H. Comon-Lundh and V. Shmatikov. Intruder deductions, constraint solving and insecurity decision in presence of exclusive or. In *LICS*, pages 271–, 2003.

[88] R. Corin. *Analysis model for security protocols*. Phd thesis, University of Twente, Twente, The Netherlands, 2006.

[89] V. Cortier. *Vérification automatiques des protocoles cryptographiques*. Thèse de doctorat, Laboratoire Spécification et Vérification, ENS Cachan, France, 2003.

[90] V. Cortier, S. Delaune, and P. Lafourcade. A survey of algebraic properties used in cryptographic protocols. *Journal of Computer Security*, 14(1):1–43, 2006.

[91] V. Cortier, M. Rusinowitch, and E. Zalinescu. A resolution strategy for verifying cryptographic protocols with cbc encryption and blind signatures. In P. Barahona and A. P. Felty, editors, *PPDP*, pages 12–22. ACM, 2005.

[92] R. Cramer, M. K. Franklin, B. Schoenmakers, and M. Yung. Multi-autority secret-ballot elections with linear work. In *EUROCRYPT*, pages 72–83, 1996.

[93] R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In H. Krawczyk, editor, *CRYPTO*, volume 1462 of *Lecture Notes in Computer Science*, pages 13–25. Springer, 1998.

[94] M. Daum and S. Lucks. Attacking hash functions by poisoned messages. *rump session, Eurocrypt*, 2005.

[95] S. Delaune and F. Jacquemard. Narrowing−based constraint solving for the verification of security protocols. In *Proceedings of the 18th International Workshop of Unification (UNIF'04)*, Cork, Ireland, 2004.

[96] S. Delaune and F. Jacquemard. A theory of dictionary attacks and its complexity. In *CSFW*, pages 2–15. IEEE Computer Society, 2004.

[97] S. Delaune, S. Kremer, and M. Ryan. Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security*, 17(4):435–487, 2009.

[98] B. den Boer and A. Bosselaers. Collisions for the compressin function of md5. In *EUROCRYPT*, pages 293–304, 1993.

[99] D. E. Denning and G. M. Sacco. Timestamps in key distribution protocols. *Commun. ACM*, 24(8):533–536, 1981.

[100] N. Dershowitz and J.P Jouannaud. Rewrite systems. In *Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B)*, pages 243–320. 1990.

[101] W. Diffie and M.E. Hellman. Nex directions in cryptography. *IEEE Trans. Info. Theory IT-22*, pages 644–654, November 1976.

[102] W. Diffie, P.l C. van Oorschot, and M.l J. Wiener. Authentication and authenticated key exchanges. *Des. Codes Cryptography*, 2(2):107–125, 1992.

[103] H. Dobbertin. Cryptanalysis of md5 compress. *rumps session, Eurocrypt*, 1996.

[104] H. Dobbertin. Cryptanalysis of md4. In *D. Gollmann editor, FSE, volume 1039 of Lecture Notes in Computer Science*, pages 53–69, Springer, 1996.

[105] H. Dobbertin, A. Bosselaers, and B. Preneel. Ripemd-160: A strengthened version of ripemd. In *D. Gollmann editor, FSE, volume 1039 of Lecture Notes in Computer Science*, pages 71–82, Springer, 1996.

[106] D. Dolev, S. Even, and R. M. Karp. On the security of ping-pong protocols. *Information and Control*, 55(1-3):57–68, 1982.

[107] D. Dolev and A. Chi-Chih Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–207, 1983.

[108] B. Donovan, P. Norris, and G. Lowe. Analyzing a library of security protocols using casper and fdr. *Workshop on Formal Methods and Security Protocols, Trento, Italy*, 1999.

[109] L. Durante, R. Sisto, and A. Valenzano. Automatic testing equivalence verification of spi calculus specifications. *ACM Trans. Softw. Eng. Methodol.*, 12(2):222–284, 2003.

[110] N. A. Durgin, P. D. Lincoln, J. C. Mitchell, and A. Scedrov. Undecidability of bounded security protocols. In *In Workshop on Formal Methods and Security Protocols, Trento, Italia*, 1999.

[111] S. Even and O. Goldreich. On the security of multi-party ping-pong protocols. In *FOCS*, pages 34–39. IEEE, 1983.

[112] S. Even, O. Goldreich, and A. Shamir. On the security of ping-pong protocols when implemented using the rsa. In Hugh C. Williams, editor, *CRYPTO*, volume 218 of *Lecture Notes in Computer Science*, pages 58–72. Springer, 1985.

[113] M. Fay. First-order unification in an equational theory, in proc. 4th workshop on automated deduction. pages 161–167, Austin, Texas, 1979.

[114] C. Fournet and M. Abadi. Hiding names: Private authentication in the applied pi calculus. In M. Okada, B. C. Pierce, A. Scedrov, H. Tokuda, and A. Yonezawa, editors, *ISSS*, volume 2609 of *Lecture Notes in Computer Science*, pages 317–338. Springer, 2002.

[115] A. Fujioka, T. Okamoto, and K. Ohta. A Practical Secret Voting Scheme for Large Scale Elections. In *AUSCRYPT'92: Proceedings of the Workshop on the Theory and Application of Cryptographic Techniques*, volume 718 of *LNCS*, pages 244–251. Springer, 1993.

[116] T. El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.

[117] S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2):281–308, 1988.

[118] J. Goubault-Larrecq. A method for automatic cryptographic protocol verification. In J. D. P. Rolim, editor, *IPDPS Workshops*, volume 1800 of *Lecture Notes in Computer Science*, pages 977–984. Springer, 2000.

[119] S. Hirose and S. Yoshida. An authenticated diffie-hellman key agreement protocol secure against active attacks. In H. Imai and Y. Zheng, editors, *Public Key Cryptography*, volume 1431 of *Lecture Notes in Computer Science*, pages 135–148. Springer, 1998.

[120] J. Hsiang and M. Rusinowitch. On word problems in equational theories. In *T. Ottmann editor, ICALP, volume 267 of Lecture Notes in Computer Science*, pages 54–71, Springer, 1987.

[121] G. P. Huet. A complete proof of correctness of the knuth-bendix completion algorithm. *J. Comput. Syst. Sci.*, 23(1):11–21, 1981.

[122] A. Huima. Efficient infinite-state analysis of security protocols. In *In* Proc. FLOC'99 Workshop on Formal Methods and Security Protocols, Trento, Italy, 1999.

[123] J.M. Hullot. Canonical forms and unification. In W. Bibel and R. A. Kowalski, editors, *CADE*, volume 87 of *Lecture Notes in Computer Science*, pages 318–334. Springer, 1980.

[124] J.M. Hullot. A catalogue of canonical term rewriting systems. *Technical Report CSL-114, Computer Science Laboratory, SRI, CA, USA*, 1980.

[125] A. Evans Jr., W. Kantrowitz, and E. Weiss. A user authentication scheme not requiring secrecy in the computer. *Commun. ACM*, 17(8):437–442, 1974.

[126] A. Juels, D. Catalano, and M. Jacobsson. Coercion-resistant electronic elections. *Cryptology ePrint Archive*, Report 2002/165, 2002.

[127] A. Juels, D. Catalano, and M. Jacobsson. Coercion-resistant electronic elections. *In WPES'05: Proceedings of the 2005 ACM workshop on Privacy in the electronic society, pages 61-70, New York, NY, USA*, ACM, 2005.

[128] D. Kähler, R. Küsters, and T. Wilke. Deciding properties of contract-signing protocols. In V. Diekert and B. Durand, editors, *STACS*, volume 3404 of *Lecture Notes in Computer Science*, pages 158–169. Springer, 2005.

[129] V. Klìma. Finding md5 collisions on a notebook pc using multi-message modifications. Cryptology ePrint Archive, Report 2005/102, 2005. `http://eprint.iacr.org/`.

[130] V. Klìma. Finding md5 collisions a toy for a notebook. Cryptology ePrint Archive, Report 2005/075, 2005. `http://eprint.iacr.org/`.

[131] D.E. Knuth and P.B. Bendix. Simple word problems in universal algebra. *Computational Algebra, J. Leach, Pergamon Press*, pages 263–297, 1970.

[132] T. Kohno, A. Stubblefield, A. D. Rubin, and D. S. Wallach. Analysis of an electronic voting system. In *IEEE Symposium on Security and Privacy*, pages 27–, 2004.

[133] R. A. Kowalski. Predicate logic as programming language. In *IFIP Congress*, pages 569–574, 1974.

[134] R. A. Kowalski and D. Kuehner. Linear resolution with selection function. *Artif. Intell.*, 2(3/4):227–260, 1971.

[135] S. Kremer and J.F. Raskin. Game analysis of abuse-free contract signing. In *CSFW*, pages 206–, 2002.

[136] S. Kremer and M. Ryan. Analysis of an electronic voting protocol in the applied pi calculus. In *S. Sagiv editor, ESOP, volume 3444 of Lecture Notes in Computer Science*, pages 186–200, Springer, 2005.

[137] D. G. Kuehner. *Strategies for improving the efficiency of automatic theorem-proving*. Phd thesis, University of Edinburgh, United Kingdom, 1971.

[138] K. O. Kürtz, R. Küsters, and T. Wilke. Selecting theories and nonce generation for recursive protocols. In P. Ning, V. Atluri, V. D. Gligor, and H. Mantel, editors, *FMSE*, pages 61–70. ACM, 2007.

[139] L. Lamport. Constructing digital signatures from a one-way functions. *Technical Report, CSL-98, SRI International*, October 1979.

[140] D. Lankford. Canonical algebraic simplification in computational logic. *Technical Report ATP-25 , Department of Mathematics, University of Texas, Austin*, 1975.

[141] B. Lee, C. Boyd, K. Kim, J. Yang, and S. Yoo. Providing receipt-freeness in mixnet-based voting protocols. In *ICISC'03: Proceedings of the 6th International Conference on Information Security and Cryptology*, volume 2791 of *LNCS*, pages 245–258. Springer, 2004.

[142] P. Lincoln, J. C. Mitchell, M. Mitchell, and A. Scedrov. A probabilistic poly-time framework for protocol analysis. In *ACM Conference on Computer and Communications Security*, pages 112–121, 1998.

[143] D. W. Loveland. A simplified format for the model elimination theorem-proving procedure. *J. ACM*, 16(3):349–363, 1969.

[144] G. Lowe. An attack on the needham-schroeder public key authentication protocol. *Information processing letters*, 1995.

[145] G. Lowe. Breaking and fixing the needham-schroeder public-key protocol using fdr. *Software - Concepts and Tools*, 17(3):93–102, 1996.

[146] Ch. Lynch. Schematic saturation for decision and unification problems. In F. Baader, editor, *CADE*, volume 2741 of *Lecture Notes in Computer Science*, pages 427–441. Springer, 2003.

[147] A. Martelli and U. Montanari. An efficient unification algorithm. *ACM Trans. Program. Lang. Syst.*, 4(2):258–282, 1982.

[148] A. Martelli and V. Montanari. Unification in linear time and space: A structured presentation. *Internal Rep. B76-16, Ist. di Elaborazione delle Informazione, Consiglio Nazionale delle Ricerche, Pisa, Italy*, July 1976.

[149] S. Mauw, J. Verschuren, and E. P. de Vink. Data anonymity in the foo voting scheme. *Electr. Notes Theor. Comput. Sci.*, 168:5–28, 2007.

[150] D. A. McAllester. Automatic recognition of tractability in inference relations. *J. ACM*, 40(2):284–303, 1993.

[151] C. Meadows. The nrl protocol analyzer: An overview. *J. Log. Program.*, 26(2):113–131, 1996.

[152] A. Menezes and N. P. Smart. Security of signature schemes in a multi-user setting. *Des. Codes Cryptography*, 33(3):261–274, 2004.

[153] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone. Handbook of applied cryptography, CRC press. 1996.

[154] R. C. Merkle. Secure communications over insecure channels. *Commun. ACM*, 21(4):294–299, 1978.

[155] D. Micciancio and B. Warinschi. Soundness of formal encryption in the presence of active adversaries. In Moni Naor, editor, *TCC*, volume 2951 of *Lecture Notes in Computer Science*, pages 133–151. Springer, 2004.

[156] J. K. Millen and V. Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *ACM Conference on Computer and Communications Security*, pages 166–175, 2001.

[157] J. K. Millen and V. Shmatikov. Symbolic protocol analysis with an abelian group operator or diffie-hellman exponentiation. *Journal of Computer Security*, 13(3):515–564, 2005.

[158] R. Milner. Communication and concurrency. *International Series in Computer Science*, Prentice Hall, 1989.

[159] R. Milner. Communicating and mobile systems: the π-calculus. *Cambridge University Press*, 1999.

[160] J. C. Mitchell, M. Mitchell, and U. Stern. Automated analysis of cryptographic protocols using mur-phi. In *IEEE Symposium on Security and Privacy*, pages 141–151. IEEE Computer Society, 1997.

[161] D. Monniaux. Abstracting cryptographic protocols with tree automata. *Sci. Comput. Program.*, 47(2-3):177–202, 2003.

[162] P. Narendran, F. Pfenning, and R. Statman. On the unification problem for cartesian closed categories. *J. Symb. Log.*, 62(2):636–647, 1997.

[163] R. M. Needham and M. D. Schroeder. Using encryption for authentication in large networks of computers. *Commun. ACM*, 21(12):993–999, 1978.

[164] H. De Nivelle. *Ordering refinements of resolution*. Phd thesis, Delft University of Technology, the Netherlands, 1995.

[165] T. Okamoto. Receipt-free electronic voting schemes for large scale elections. In B. Christianson, B. Crispo, T. M. A. Lomas, and M. Roe, editors, *Security Protocols Workshop*, volume 1361 of *Lecture Notes in Computer Science*, pages 25–35. Springer, 1997.

[166] Paritosh K. Pandya and Jaikumar Radhakrishnan, editors. *FST TCS 2003: Foundations of Software Technology and Theoretical Computer Science, 23rd Conference, Mumbai, India, December 15-17, 2003, Proceedings*, volume 2914 of *Lecture Notes in Computer Science*. Springer, 2003.

[167] M. Paterson and M. N. Wegman. Linear unification. *J. Comput. Syst. Sci.*, 16(2):158–167, 1978.

[168] L. C. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6(1-2):85–128, 1998.

[169] O. Pereira and J.J. Quisquater. On the perfect encryption assumption. *Proc. of the 1st Workshop on Issues in the Theory of Security (WITS'00)*, pages 42–45, Geneva (Switzerland), 2000.

[170] T. Pornin and J. P. Stern. Digital signatures do not guarantee exclusive ownership. In J. Ioannidis, A. D. Keromytis, and M. Yung, editors, *ACNS*, volume 3531 of *Lecture Notes in Computer Science*, pages 138–150, 2005.

[171] M. O. Rabin. Digitalized signatures and public-key functions as intractable as factorization. *MIT/LCS/TR-212, MIT Laboratory for Computer Science*, 1979.

[172] R. L. Rivest. The md4 message digest algorithm. In A. Menezes and S. A. Vanstone, editors, *CRYPTO*, volume 537 of *Lecture Notes in Computer Science*, pages 303–311. Springer, 1990.

[173] R. L. Rivest. The md5 message digest algorithm. *RFC*, 1321, April, 1992.

[174] R. L. Rivest, A. Shamir, and L. M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.

[175] R. L. Rivest, A. Shamir, and L. M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.

[176] J.A. Robinson. A machine-oriented logic based on the resolution principle. *J. ACM*, 12(1):23–41, 1965.

[177] J.A. Robinson. Fast unification. *In Theorem Proving Workshop, Oberwolfach, W. Germany*, 1976.

[178] M. Rusinowitch and M. Turuani. Protocol insecurity with finite number of sessions is np-complete. In *CSFW*, pages 174–. IEEE Computer Society, 2001.

[179] P. Ryan, S. Schneider, M. Goldsmith, G. Lowe, and B. Roscoe. Modelling and analysis of security protocols. *Livres aux éditions Addison-Wesley*, 2001.

[180] H. Lin S. Delaune and Ch. Lynch. Protocol verification via rigid/flexible resolution. In N. Dershowitz and A. Voronkov, editors, *LPAR*, volume 4790 of *Lecture Notes in Computer Science*, pages 242–256. Springer, 2007.

[181] K. Sako and J. Kilian. Receipt-free mix-type voting scheme - a practical solution to the implementation of a voting booth. In *EUROCRYPT*, pages 393–403, 1995.

[182] D. Sandler, K. Derr, and D. S. Wallach. Votebox: A tamper-evident, verifiable electronic voting system. In P. C. van Oorschot, editor, *USENIX Security Symposium*, pages 349–364. USENIX Association, 2008.

[183] Y. Sasaki, Y. Naito, N. Kunihiro, and K. Ohta. Wang's sufficient conditions of md5 are not sufficient. 2005.

[184] M. Schmidt-Schauß. Unification under associativity and idempotence is of type nullary. *J. Autom. Reasoning*, 2(3):277–281, 1986.

[185] C.P. Schnorr. Efficient signature generation by smart cards. *J. Cryptology*, 4(3):161–174, 1991.

[186] K. U. Schulz. Makanin's algorithm for word equations - two improvements and a generalization. In K. U. Schulz, editor, *IWWERT*, volume 572 of *Lecture Notes in Computer Science*, pages 85–150. Springer, 1990.

[187] H. Seidl and K. Neeraj Verma. Flat and one-variable clauses: Complexity of verifying cryptographic protocols with single blind copying. In F. Baader and A. Voronkov, editors, *LPAR*, volume 3452 of *Lecture Notes in Computer Science*, pages 79–94. Springer, 2004.

[188] V. Shmatikov. Decidable analysis of cryptographic protocols with products and modular exponentiation. In *D. A. Schmidt editor, ESOP, volume 2986 of Lecture Notes in Computer Science*, pages 355–369, Springer, 2004.

[189] V. Shoup. On formal models for secyre key exchange. *Cryptology ePrint Archive Report 1999/012*, 1999.

[190] J. R. Slagle. Automated theorem-proving for theories with simplifiers commutativity, and associativity. *J. ACM*, 21(4):622–642, 1974.

[191] B. Smyth, M.D. Ryan, S. Kremer, and M. Kourjieh. Election verifiability in electronic voting protocols. *In the 4th Benelux Workshop on Information and System Security, Louvain-la-Neuve, Belgium*, November 2009.

[192] P. Syverson, C. Meadows, and I. Cervesato. Dolev-yao is no better than machiavelli. In *First Workshop on Issues in the Theory of Security WITS00*, pages 87–92, 2000.

[193] M. Talbi, B. Morin, V. Viet Triem Tong, A. Bouhoula, and M. Mejri. Specification of Electronic Voting Protocol Properties Using ADM Logic: FOO Case Study. In *ICICS'08: Proceedings of the 10th International Conference on Information and Communications Security Conference*, volume 5308 of *LNCS*, pages 403–418. Springer, 2008.

[194] I. Tsahhirov and P. Laud. Application of dependency graphs to security protocol analysis. In G. Barthe and C. Fournet, editors, *TGC*, volume 4912 of *Lecture Notes in Computer Science*, pages 294–311. Springer, 2007.

[195] M. turuani. *Sécurité des protocoles cryptographiques: décidabilité et complexité*. Thèse de doctorat, INRIA, University of Henri Pointcaré, Nancy, 2003.

[196] J. van Eijck and S. Orzan. Epistemic verification of anonymity. *Electr. Notes Theor. Comput. Sci.*, 168:159–174, 2007.

[197] M. VenturiniZelli. Complexity of the unification algorithm for first-order expressions. *Calcolo 12, 4*, pages 361–372, 1975.

[198] R. J. Waldinger and K. N. Levitt. Reasoning about programs. *Artif. Intell.*, 5(3):235–316, 1974.

[199] X. Wang, D. Feng, X. Lai, and H. Yu. Collisions for hash functions md4, md5, haval-128 and ripemd. 2004.

[200] X. Wang, X. Lai, D. Feng, H. Chen, and X. Yu. Cryptanalysis of the hash functions md4 and ripemd. In *R. Cramer editor, EUROCRYPT, volume 3494 of Lecture Notes in Computer Science*, pages 1–18, Springer, 2005.

[201] X. Wang, Y. L. Yin, and H. Yu. Finding collisions in the full sha-1. In V. Shoup, editor, *CRYPTO*, volume 3621 of *Lecture Notes in Computer Science*, pages 17–36. Springer, 2005.

[202] X. Wang and H. Yu. How to break md5 and other hash functions. In *R. Cramer editor, EUROCRYPT, volume 3494 of Lecture Notes in Computer Science*, pages 19–35, Springer, 2005.

[203] B. Warinschi. A computational analysis of the needham-schroeder-(lowe) protocol. *Journal of Computer Security*, 13(3):565–591, 2005.

[204] J. Yajima and T. Shimoyama. Wang's sufficient conditions of md5 are not sufficient. 2005.

[205] E. Zalinescu. *Sécurité des protocoles cryptographiques: décidabilité et résultats de transfert*. Thèse de doctorat, INRIA, University of Henri Pointcaré, Nancy, 2007.