Verification and composition of security protocols with applications to electronic voting

Ştefan Ciobâcă Ciobâcă

THÈSE

présentée à l'École Normale Supérieure de Cachan

pour obtenir le grade de

Docteur de l'École Normale Supérieure de Cachan

par Ştefan Ciobâcă

Spécialité: Informatique

Verification and Composition of Security Protocols with Applications to Electronic Voting

Thèse soutenue le 9 Decembre 2011

Composition du jury :

• Bruno Blanchet	rapporteur
• Véronique Cortier	directrice de thèse
• Thomas Genet	examinateur
• Jean Goubault-Larrecq	directeur de thèse
• Steve Kremer	directeur de thèse
• Jean-Jacques Lévy	président du jury
• Luca Viganò	rapporteur

Laboratoire Spécification et Vérification (LSV), ENS Cachan, CNRS & INRIA
61, avenue du Président Wilson, 94235 Cachan Cedex

Abstract

This thesis is about the formal verification and composition of security protocols, motivated by applications to electronic voting protocols. Chapters 3 to 5 concern the verification of security protocols while Chapter 6 concerns composition.

We show in Chapter 3 how to reduce certain problems from a quotient term algebra to the free term algebra via the use of strongly complete sets of variants. We show that, when the quotient algebra is given by a convergent optimally reducing rewrite system, finite strongly complete sets of variants exist and are effectively computable.

In Chapter 4, we show that static equivalence for (classes of) equational theories including subterm convergent equational theories, trapdoor commitment and blind signatures is decidable in polynomial time. We also provide an efficient implementation.

In Chapter 5 we extend the previous decision procedure to handle trace equivalence. We use finite strongly complete sets of variants introduced in Chapter 3 to get rid of the equational theory and we model each protocol trace as a Horn theory which we solve using a refinement of resolution. Although we have not been able to prove that this procedure always terminates, we have implemented it and used it to provide the first automated proof of vote privacy of the FOO electronic voting protocol.

In Chapter 6, we study composition of protocols. We show that two protocols that use arbitrary disjoint cryptographic primitives compose securely if they do not reveal or reuse any shared secret. We also show that a form of tagging is sufficient to provide disjointness in the case of a fixed set of cryptographic primitives.

Résumé

Cette thèse concerne la vérification formelle et la composition de protocoles de sécurité, motivées en particulier par l'analyse des protocoles de vote électronique. Les chapitres 3 à 5 ont comme sujet la vérification de protocoles de sécurité et le Chapitre 6 vise la composition.

Nous montrons dans le Chapitre 3 comment réduire certains problèmes d'une algèbre quotient des termes à l'algèbre libre des termes en utilisant des ensembles fortement complets de variants. Nous montrons que, si l'algèbre quotient est donnée par un système de réécriture de termes convergent et optimalement réducteur (optimally reducing), alors des ensembles fortement complets de variants existent et sont finis et calculables.

Dans le Chapitre 4, nous montrons que l'équivalence statique pour (des classes) de théories équationnelles, dont les théories sous-terme convergentes, la théorie de l'engagement à trappe (trapdoor commitment) et la théorie de signature en aveugle (blind signatures), est décidable en temps polynomial. Nous avons implémenté de manière efficace cette procédure.

Dans le Chapitre 5, nous étendons la procédure de décision précédente à l'équivalence de traces. Nous utilisons des ensembles fortement complets de variants du Chapitre 3 pour réduire le problème à l'algèbre libre. Nous modélisons chaque trace du protocole comme une théorie de Horn et nous utilisons un raffinement de la résolution pour résoudre cette théorie. Meme si nous n'avons pas réussi à prouver que la procédure de résolution termine toujours, nous l'avons implementée et utilisée pour donner la première preuve automatique de l'anonymat dans le protocole de vote électronique FOO.

Dans le Chapitre 6, nous étudions la composition de protocoles. Nous montrons que la composition de deux protocoles qui utilisent des primitives cryptographiques disjointes est sûre s'ils ne révèlent et ne réutilisent pas les secrets partagés. Nous montrons qu'une forme d'etiquettage de protocoles est suffisante pour assurer la disjonction pour un ensemble fixé de primitives cryptographiques.

to my mother,
who has always encouraged and supported me in everything I did

Acknowledgements

First of all, I would like to thank all the members of the jury and Bruno Blanchet and Luca Viganò in particular for having accepted to review the thesis manuscript.

I would like to thank my advisors, Véronique Cortier and Steve Kremer, who have always managed to make time for me and to provide interesting scientific insights, good advice and encouragements. I would also like to thank Jean Goubault-Larrecq for accepting to be my official advisor and Stéphanie Delaune for co-advising my research internship during my master's thesis. I have learned a lot from all of them. I have also learned a lot from my collaboration with Rohit Chadha and from the interesting scientific discussions that I have had with Mark Ryan, Joshua Guttman, Cas Cremers, Vicent Cheval and Hubert Comon-Lundh on security protocols and with Serge Haddad and Florent Jacquemard on algorithms and tree automata.

I would like to thank all the members of the LSV for the great research atmosphere and the administrative staff in particular for their help. I thank Dietmar for always being there and for his valuable advice. Nicolas has always been very friendly, despite my annoying LaTeX questions. My office-mates Adam, Pierre, Elie, Cesar and Sergiu were all great and I had a good time with them.

My passion for computer science started by going to the informatics olympiad during school and continued with the excellent curriculum of the Faculty of Computer Science in Iaşi, the Department of Computer Science of Konstanz University and the *Master Parisien de Recherche en Informatique* at ENS Cachan. I am indebted to all my teachers, colleagues and professors that I have met in all this time.

I thank my friends Adina, Daniel, Loredana, Daniel, Yuki, Sergiu, Oana, Corina, Cornel and Anca who made living in France a joy. I also thank my friends back home and my family for bearing with me during all this time I was away. Carmen was especially supportive and understanding during the final months of writing up the thesis manuscript.

This thesis was financially supported by the ANR SeSur AVOTE research project (`http://www.lsv.ens-cachan.fr/Projects/anr-avote/`).

Contents

Chapter 1

Introduction

As the world is getting more and more connected with the advent of the Internet into everyday life, the need for security and privacy of communication has risen considerably. Security and privacy are achieved today by employing *security protocols* (also called *cryptographic protocols*). Security protocols are rules for communication between participants which make use of cryptographic primitives such as encryption to provide security guarantees such as confidentiality of data.

1.1 Cryptographic Primitives

One of the earliest documented uses of cryptography was by Julius Cæsar, who would employ the *Cæsar cipher* [122] to protect messages of military significance. The Cæsar cipher and the slightly more sophisticated ciphering methods that followed are known today to be trivially broken using statistical methods. As the need for secure communication increased with the advent of electronic computers and electronic communication, the need for strong ciphers (i.e. which cannot easily be attacked) increased. In 1977 the United States Federal Information Processing Standard adopted the Data Encryption Standard (DES) [111] as a standard. DES enjoyed widespread use internationally; however, due to the small key-length, it quickly became vulnerable to brute-force attacks [103]. This led to the adoption of AES (Advanced Encryption Standard) [86] to supersede DES.

These standards are openly available for anyone to implement and analyze. The openness of the standards follows Kerckhoffs' principle [75], widely embraced by the scientific community, which states that a cryptosystem should be secure even if everything about the system, except the key, is public knowledge (i.e. including the algorithms for encrypting or decrypting). Due to the fact that anyone can analyze the security of these ciphers, we gain confidence in their security: if there were a problem with a standard, people would likely find and publish it.

However, ciphers have a big disadvantage in that they require the two parties that are communicating to have the same key; the fact that the same key is used for encryption and decryption is why the key is called a symmetric key and the encryption scheme is called symmetric encryption. Securely sharing a symmetric key can be achieved for example by meeting face-to-face before the sensitive communication takes place and agreeing on the key. However, with the striking development of *public key cryptography* [117, 77], another possibility opened up. In public key cryptography, an agent generates a pair of keys: a *public key* which he shares with everyone and a *private key* which he keeps to himself. If someone wishes to communicate sensitive information to the agent, he encrypts the plaintext with the public key of the recipient; the recipient in turn receives the ciphertext which may be decrypted using the private key to obtain the original plaintext. Therefore, in theory, there is no more need to meet face-to-face to agree on a shared key. It is sufficient to *authentically* know the public key of the person which is to receive the sensitive information. Public/private key encryption is also called asymmetric encryption due to the fact that different keys are used for encryption and respectively decryption.

However, in practice, public key encryption is computationally too expensive. Therefore *key-establishment protocols* have been developed, in which two participants usually communicate using public key encryption in order to establish a fresh *symmetric session key*, which they are going to use for the remainder of the session to exchange encrypted messages. This has the advantage of not having to meet in person in order to agree on a symmetric key. Furthermore, the computationally expensive asymmetric cryptography operations are performed only once; after the symmetric session key has been established, messages can be encrypted using symmetric key cryptography, which is computationally less expensive.

Digital signatures [117] also work using private/public key pairs. An agent *signs* a message using his private key to obtain a signed message. Anyone who knows the public key can verify that the resulting message was indeed signed using the associated private key, thereby allowing to establish the identity of the signer, provided that the public key is known authentically. Authenticity of public keys is a particularly important problem [106, p. 27]. Authenticity in this case means to be sure that a public key indeed belongs to the intended recipient and not to an intruder. Digital signatures form the base of public-key infrastructures (PKIs) [122]. PKIs are designed to *bootstrap* authenticity as follows. Trusted third party (TTP) institutions such as a government provide *certificates* which consist of a pair public key - associated identity, pair which is digitally signed by the TTP. By signing the pair, the TTP guarantees that the detainer of the private key associated to the public key has the claimed identity. To check these certificates, the public keys of TTPs are widely distributed via secure channels (for example, every web browser ships with a list of trusted third parties). PKIs usually work in hierarchical way, with higher-level TTPs certifying lower-level TTPs which in their turn provide certificates.

The security of the exchanged messages relies on the security of the cryptographic primitives involved. For ciphers such as AES, confidence in their security results from the fact that even if a large number of person-hours were devoted to trying to break it, no successful practical attack was found. For public key encryption and digital signatures, confidence in their security is higher due to the fact that there are mathematical proofs which show that breaking the encryption or the signature scheme (e.g. by finding the private key associated to a known public key) is equivalent to solving problems which are widely believed to be computationally difficult [88]. Computational difficulty means that the problem requires a lot of computational resources such as time and memory, which would make it intractable for anyone to solve it in reasonable time.

1.2 Security of Protocols

The algorithms for symmetric key encryption, for public key encryption and decryption and for digital signatures are called *cryptographic primitives* since they are used as part of more complex protocols. The security of cryptographic primitives is analyzed individually: in the case of asymmetric key cryptography, the primitives are proven to be secure via a reduction to an intractable problem and in the case of symmetric key encryption, security is established heuristically.

However, even if the security primitives are secure by themselves, combining them to obtain security protocols such as key establishment protocols might prove to be insecure for several reasons. One of the first examples of this is the Kerberos [120] protocol. The Kerberos protocol is based on the Needham-Schroeder Shared Key (NSSK) protocol, introduced by the Needham and Schroeder in [112]. The goal of the protocol is to establish a session key between two agents who already share a secret key with the same trusted server. Denning and Sacco identified [76] in 1981 an attack against the NSSK protocol. The attack allows the intruder to force an agent to accept as session key an already compromised key. As a fix to the NSSK protocol, Denning and Sacco propose *timestamping* certain message of the protocol. Participants are expected to check the timestamp and to ignore messages which have an expired timestamp.

In 1995, Lowe [104] identified a flaw in the Need-Schroeder Public Key (NSPK) protocol, published by Needham and Schroeder in the same article [112] as the NSSK protocol. The goal of the NSPK protocol is to mutually authenticate two parties via a trusted server using public key cryptography. If an honest agent initiates a session with the impostor, the attack identified

by Lowe allows the impostor to make it look to another party that he is the honest agent. What is interesting about the Denning-Sacco attack against NSSK and the Lowe attack against NSPK is that these attacks do not attack the underlying encryption schemes, but the *logical structure* of the protocols. For example the Lowe attack on NSPK will work no matter which public key encryption scheme is used.

A more recent example of a protocol which is insecure (despite the cryptographic primitives employed being secure) is the implementation of a Single Sign-On protocol by Google. A Single Sign-on protocol is designed to ensure the a user authenticates using the same credentials to several service providers. Armando et al. show [11] that Google's implementation of an SSO protocol is insecure by allowing a dishonest service provider to impersonate a user to another service provider. The attack only exploits the expected behavior of the protocol participants and not any weakness of the underlying cryptographic primitives.

1.3 The Dolev-Yao Model

As attacks can have serious economic, military and social repercussions, there is a significant need to gain confidence that the security protocols that we use are not vulnerable to attacks. In their seminal paper [78], Dolev and Yao identify the need to *formally specify* security protocols and to *formally verify* their expected security properties, independently from the security of the underlying cryptographic primitives. This has given the rise to a class of models for security protocols generically called *the Dolev-Yao model*.

The main ingredient of the Dolev-Yao model is that messages exchanged by parties are represented as *terms* in a *term algebra*, with function symbols representing cryptographic primitives. In the Dolev-Yao model, protocols are modeled in various formalisms such as a set of *roles* in role-based models [107], sets of strands in *strand-spaces*-based modes [85], and processes in process algebra based models [5, 3]. The power of the intruder is modeled as a *deduction system* [7, 107], as an equational theory [3] or as the combination of a deduction system with an equational theory [90]. Additionally, the intruder can intercept all messages, block any message, construct new messages according to the deduction rules, impersonate a protocol agent and send messages of his own creation to the agents.

Once the various details described above are fixed, a *model* of the security protocol can be created. It remains to model the security property that we expect of the protocol. Traditionally, two security properties are verified:

1. *secrecy* (or *confidentiality*) of some secret value, which intuitively means that the attacker cannot find the secret value and

2. *authentication*, which means intuitively that an agent is talking to whom he believes to be talking.

Secrecy is usually modeled as non-deducibility [118] and authentication is modeled as an implication in some form of temporal logic [105]. There are other types of security properties such as *vote privacy* which can be verified in the case of electronic voting protocols; we discuss them in greater detail in the next section.

Once models of the security protocol and of the expected security property are built, we can proceed to verify if the model of the protocol satisfies the expected property. The act of formally verifying that a model satisfies a property is called *model checking* (as in Figure 1.1). In our case, the model is the model of a security protocol and the property to be checked is a security property.

1.3.1 Modeling Cryptographic Primitives

In order to gain more confidence in the security of the protocol, the model should in general describe as accurately as possible the protocol. Initially, the Dolev-Yao model allowed only for a fixed set of cryptographic primitives including (symmetric) encryption modeled by a function

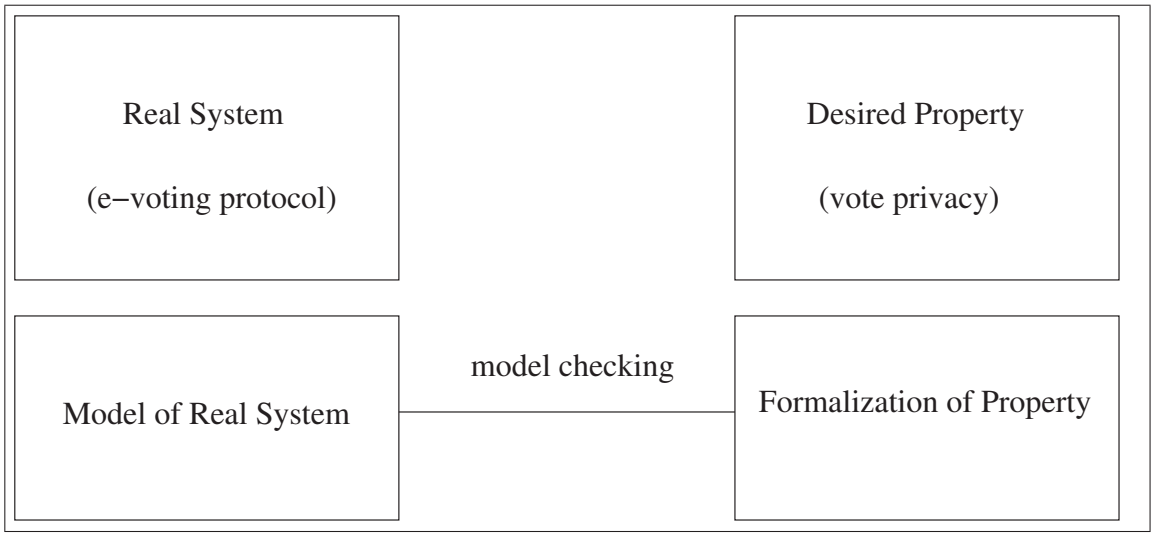

Figure 1.1: Model Checking

symbol enc of arity 2. The term $\mathrm{enc}(x, y)$ would model the encryption of some data x by the key y. The intruder power would be given by a deduction system consisting of the following inference rules:

$$\textsc{Decryption}\ \frac{\mathrm{enc}(x,y) \qquad y}{x} \qquad \textsc{Encryption}\ \frac{x \qquad y}{\mathrm{enc}(x,y)}.$$

The inference rules allow the adversary to construct new messages from messages already known either by encrypting known data x with known key y to obtain $\mathrm{enc}(x, y)$ or by decrypting an already-known encrypted message $\mathrm{enc}(x, y)$ by the already-known associated encryption key y to obtain x (the fact that the intruder can learn new message by observing network traffic is implicit and there is no need to capture this explicitly in the inference rules). Due to the lack of other inference rules, *the only way* for the attacker to find the contents of an encrypted message is by knowing the encryption key. This is known as the *perfect encryption hypothesis* [115] and has been criticized due to the fact that it is not realistic.

Attempts to weaken [61] the perfect encryption assumption included adding an *equational theory* to the term algebra. For example, homomorphic encryption [100], where the *sum* of two ciphertexts is the ciphertext of the sum of the two plaintexts, can be modeled by the following equation:

$$\mathrm{enc}(x, y) + \mathrm{enc}(z, y) = \mathrm{enc}(x + z, y).$$

Terms are then interpreted in the quotient algebra. Another use of equational theories is to model cryptographic primitives such as *exclusive or* (XOR) [42], denoted here as $\oplus$:

$$\begin{array}{ll} x \oplus 0 = x & x \oplus x = 0 \\ x \oplus (y \oplus z) = (x \oplus y) \oplus z & x \oplus y = y \oplus x. \end{array}$$

It turns out that using equational theories allows one to get rid entirely of the deduction system by adding a function symbol dec modeling the decryption algorithm and satisfying the following equation:

$$\mathrm{dec}(\mathrm{enc}(x, y), y) = x.$$

Modeling the decryption algorithm as a function symbol (dec) is called the Dolev-Yao model with explicit destructors [69].

1.3.2 Verifying Security Protocols

Once a model has been fixed, the complexity of the model checking problem should be investigated. In general, model checking of security protocols is undecidable [95, 80]. The undecidability proof

is quite robust in the sense that reasonable variations of the problem are undecidable as well. Therefore, more restricted versions of the model checking problem were considered. The case of a bounded number of sessions (i.e. when each user is only allowed to run the protocol for a bounded number of times) was considered ([118, 43, 42, 41, 54, 8, 32]) and it was shown that for several sets of usual cryptographic primitives, the insecurity problem (i.e. does a message remain secret) is NP-complete [118, 42]. Another approach to get round the undecidability is to verify *sound approximations* of protocols. An approximation of a protocol is sound whenever showing that the approximation is secure, the protocol is also secure. Conversely, it may be that approximations are subject to attacks which are not attacks of the real protocol. A number of theoretical results and tools [63, 12, 25, 82] show how to soundly approximate security protocols in order to prove them secure.

Another line of work is to reinforce security of existing protocols. It was shown that tagging [94, 101] can be added to a protocol to stop type-flaw attacks, i.e. attacks that arise due to messages being confounded for something else. It was also shown that by dynamic tagging [10], a protocol which is secure for one session is secure for an unbounded number of sessions. Finally, there are ways [62] to *compile* a protocol which is secure in a weak sense into a protocol which is secure against a full attacker.

1.3.3 Composing Security Protocols

Yet another aspect is modularity. Even though protocols have been shown to be individually secure, there are typically several of them running at the same time. Undesired interactions among protocols which are individually secure could lead to security problems. Therefore protocol *compositionality* is studied. It was shown for example that protocols which are disjoint in a certain sense compose well [93, 92]. Using different tags for different protocols [57, 60] also leads to secure protocols composition. Another way to ensure compositionality is to make the protocols compositional from the very beginning [33, 34], by the security definition. Compositionality is desired for several reasons:

1. first of all, it is not possible to foresee all of the protocols which may run on a network such as the Internet at the same time. Therefore it is desirable to have protocols which are compositionally secure, i.e. such that the global result is secure;

2. secondly, even if a global model of all protocols were constructible, it would be very difficult to model check it since it would probably be very complex.

We have seen a very brief survey of the state-of-the-art in the literature on verification and composition of protocols in the Dolev-Yao model.

1.4 Electronic Voting Protocols

Recent advances in computer technology have made it possible for electronic voting to be more widely deployed. Electronic voting (e-voting) offers many advantages over classical voting, in that the sum of the votes can be computed very quickly and more accurately (since a computer does the counting of the votes).

However, while electronic voting is more convenient, it comes with several risks as it has the potential to make abuse easier to perform on a large scale. A dishonest agent could try to attack the protocol in order to change the tally of the result. Furthermore, the dishonest agent could be (part of) the voting authorities. Therefore electronic voting protocols should be designed to be secure even in the presence of corrupted officials. Furthermore, the terminal on which the voting is performed could be infected by malware and therefore it cannot be trusted.

These issues can be mitigated by a form of auditing that is called *verification*. E-voting protocols for which the tally can be *verified* to be correct after the election are said to have the *verifiability* [99] property. Verifiability comes in several shapes. *Individual verifiability* means that

every individual should be able to verify that his vote has been taken into account in the final tally. *Universal verifiability* means that anyone should be able to verify that the tally includes votes only from the eligible voters and that each eligible voter voted at most once. Sometimes *eligibility verifiability* (i.e. all votes came from eligible voters) is separated from universal verifiability and is seen as a distinct property [99].

An important aspect in e-voting protocols is *vote privacy*. Vote privacy means that the link between a voter and his vote remains secret to anyone except the voter, including corrupted voting authorities. Vote privacy is essential in e-voting protocols in order to prevent retaliation against political dissidents.

Another property that is desirable of e-voting protocols is *receipt-freeness*. Receipt-freeness essentially prevents vote selling and it means that a voter cannot convince anyone else that he voted a certain way. Such a proof would act as a "receipt" and would allow a voter to sell his vote. Related to receipt-freeness is *coercion-resistance*. A protocol is coercion-resistant if an attacker cannot force a voter to vote a certain way (i.e. by blackmailing him or by holding a gun to his head). The property of *fairness* means that no partial results of the vote are revealed during the election and that only eligible voters can cast a valid vote.

We can see that in contrast to security properties such as confidentiality or authentication which are traditionally verified for cryptographic protocols, the security properties which are expected of electronic voting protocols seem to be much richer.

At this point, it might be tempting to try and reduce vote privacy to a confidentiality problem which was shown to be co-NP-complete for certain cryptographic primitives and a bounded number of sessions [118]. The confidentiality (or secrecy) problem is, given a protocol P and a message s, can the intruder compute s?

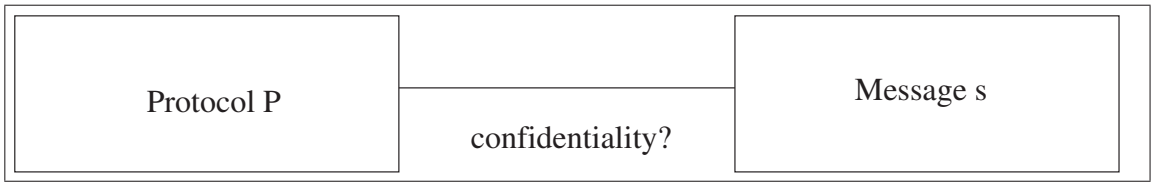

Figure 1.2: The secrecy problem

However, it is not possible to reduce vote privacy to secrecy because all of the votes are known to the intruder! Indeed, the votes are well-known to everybody before the election takes place: they can be either 'yes'/'no' in case of binary election or a set of names in case of electing an official. Furthermore, in some cases, it is not possible to have vote privacy: assume that everyone voted 'yes'. The fact that the vote of every individual must have been 'yes' follows directly from the tally which is supposedly publicly announced at the end of the election.

However, vote privacy can be modeled as an *equivalence* property. Delaune, Kremer and Ryan propose such a definition of vote privacy in [72]: a protocol satisfies vote privacy if the actual run of the protocol is indistinguishable from the run where two participants swap their votes.

Properties such as vote privacy, coercion-resistance and receipt-freeness can be expressed in terms of indistinguishability from the point of view of the attacker. We have seen for example that

$$P\{V_1 \mapsto \text{"}yes\text{"}, V_2 \mapsto \text{"}no\text{"}\} \approx P\{V_1 \mapsto \text{"}no\text{"}, V_2 \mapsto \text{"}yes\text{"}\}$$

is a way of specifying vote privacy, where P is the voting process, V_1 and V_2 represent the votes of two participants and $\approx$ denotes indistinguishability form the point of view of the attacker. In contrast to equivalence properties, confidentiality and authentication are *trace properties* in the sense that it is sufficient to look at any individual run (also called a trace) of the protocol to conclude that they hold or not. In contrast, it is necessary to look at all runs globally in order to establish an equivalence property.

We have shown that electronic voting protocols are special because they are expected to have much richer security properties than the traditional secrecy and authentication properties. Furthermore, to ensure these properties, e-voting protocols make use of rather esoteric cryptographic

primitives [72, 16] such as *blind signatures* [36], *trapdoor commitments* [113] and *homomorphic encryption* [96].

In contrast to the tools which are available to prove and reason about authenticity and confidentiality, there are not so many tools available for reasoning about equivalence properties which are needed for establishing the security of e-voting protocols. Furthermore, existing tools are usually limited to handling a fixed set of cryptographic primitives which usually exclude the more esoteric primitives required by e-voting protocols.

We have shown that properties such as vote privacy are fundamentally different from secrecy properties: vote privacy is an equivalence property while secrecy is a trace property. Furthermore, receipt-freeness and coercion-resistance can also be modeled as equivalence properties [72]. Therefore, in order to verify that protocols satisfy such properties, it is sufficient to devise a procedure for the *equivalence problem*:

Input:	two protocols P and Q
Output:	is P indistinguishable from Q? (i.e. $P \approx Q$).

The relation $\approx$ should model *behavioral equivalence* between processes, i.e. two protocols which are in the $\approx$ relation should be indistinguishable to the intruder. Usually, $\approx$ is taken to be observational equivalence, but it seems that trace equivalence is a better alternative [38]. In this thesis, we will only consider the case where $\approx$ is trace equivalence. It is well-known that in general, the equivalence problem is undecidable since is generalizes the secrecy problem. Therefore various restrictions of the problem are usually considered.

1.5 Contributions

In this thesis, we show that automated verification of security protocols with respect to equivalence properties is feasible for certain classes of bounded protocols and that protocols making use of arbitrary disjoint cryptographic primitives compose well with respect to confidentiality. We detail our contributions below.

1.5.1 The Strong Finite Variant Property

In Chapter 3, we introduce the notion of a *strongly complete set of variants* of a term. Strongly complete sets of variants allow in many cases to get rid of the equational theory and solve problems in the free term algebra. We compare (Section 3.4) the notion of strongly complete set of variants to the notion of complete set of variants, already introduced in [53] and we show that they are different (3.4.1); however, in the presence of free symbols, the problem of finding strongly complete sets of variants can be reduced to the problem of computing complete sets of variants (3.4.2). We show that convergent *optimally reducing* term rewriting systems have finite strongly complete sets of variants. Optimally reducing term rewriting systems include rewrite systems which are relevant to security protocols such as subterm convergent term rewriting systems, blind signatures, trapdoor commitment and others.

Strongly complete sets of variants are motivated by our procedure in Chapter 5 that checks equivalence between processes. In this procedure, strongly complete sets of variants are used to get rid of the equational theory and reduce the problem to the free algebra.

Furthermore, strongly complete sets of variants are of independent interest, because they allow to construct finite complete sets of unifiers for equational unification problems as shown in Section 3.6. This chapter is based on work presented at the UNIF 2011 workshop [47] and it has been implemented in the tool SubVariant [48].

Related Work

The notion of finite set of variants was introduced by Comon and Delaune in [53]. It can be useful for solving unification and disunification properties [53, 84] and in getting rid of the equational

theory for the verification of reachability in security protocols [68]. Methods for proving and for disproving the finite variant property are given in [83]. As opposed to previous work which considers the general case of a convergent term rewriting system modulo an equational theory E, we restrict ourselves to the case where E is empty. In this case (where E is empty) we extend the notion of complete set of variants to the notion of *strongly* complete set of variants. This is needed in order to obtain complete sets of unifiers constructively and to get rid of the equational theory in our procedure (Chapter 5) for verifying trace equivalence.

1.5.2 A Decision Procedure for Static Equivalence

In Chapter 4 we show that the equivalence problem is decidable for a large class of equational theories and for a *passive attacker*. A passive attacker is allowed to listen to network traffic, but it cannot modify, block or send new messages on the network. Therefore, in the case of a passive attacker, the equivalence problem can be reduced to the *static equivalence* problem, which is, given two sequences of (ground) messages S_1 and S_2, can the intruder distinguish between S_1 and S_2? Deciding static equivalence is a prerequisite to deciding more general equivalences which take into account the dynamic behavior of the protocol. Furthermore, static equivalence is interesting in its own right because it helps to partially automate equivalence proofs.

Our procedure is based on the encoding into Horn clauses of the two sequences of messages and of the possible intruder actions. A carefully tuned refinement of resolution can then be used to saturate the set of Horn clauses. From the saturated set of Horn clauses, it is easy to decide if the two sequences of messages are equivalent. Our procedure is sound and complete for any equational theory which is modeled as a convergent term rewriting system. Furthermore, we have shown that our procedure always terminates for several (classes of) equational theories particularly relevant to e-voting protocols, including subterm convergent rewrite systems, trapdoor commitment and blind signatures. Most notably, this is the first decidability result for static equivalence under trapdoor commitment, a cryptographic primitive which is used in e-voting protocols such as Okamoto [114] to ensure receipt-freeness. We also show that by using a *fair* saturation process, we obtain decidability of static equivalence for an equational theory modeling a homomorphic encryption scheme.

As a by-product, the same procedure can be used to decide the intruder deduction problem for the same class of equational theories. The intruder deduction problem is, given a set of (ground) messages S and another term t, can the intruder apply deduction rules in order to derive t from S? The intruder deduction problem can be seen as the restriction of the confidentiality problem to a passive adversary (i.e. an adversary which can intercept messages on the network, but which cannot modify them).

Our procedure for deciding static equivalence and the intruder deduction problem terminates in polynomial time for several equational theories including blind signatures, trapdoor commitment and subterm convergent equational theories. It was implemented in the KiSs tool and it is based on work published in [50, 51].

Related Work

Many decision procedures have been proposed for the intruder deduction problem (e.g. [39, 8, 100, 40]) under a variety of equational theories. We do not discuss these here and we concentrate on work which is directly related to static equivalence. In [2] a decidability result is obtained for a class of equational theories which are locally stable and locally decidable. This class include subterm convergent rewrite systems, for which a polynomial-time procedure is shown to exist. In [58], several decision results are shown for *monoidal* equational theories. A combination result (if deduction and static equivalence are decidable for two disjoint equational theories, then deduction and static equivalence are also decidable for the union of the two theories) was obtained in [13]. An algorithm inspired by [2] which can handle subterm convergent rewrite theories and blind signatures was published in [21, 22] and implemented in [20]. A procedure designed and implemented to obtain asymptotically faster algorithms for static equivalence than [20, 45] is given

in [55], although it cannot handle more cryptographic primitives. Procedures for verifying behavioral equivalences such as implemented in ProVerif [25, 27] can also prove static equivalence; but they are "overkill" in the sense that they were designed for a more difficult problem. Furthermore, none of them can handle the theory of trapdoor commitment.

1.5.3 Automated Verification of Trace Equivalence

In Chapter 5, we show that the automated verification of equivalence properties is feasible by giving a procedure for verifying equivalence between bounded determinate security protocols without else branches. Our procedure is sound and complete for equational theories having the strong finite variant property which we introduce in Chapter 3. Therefore our procedure works for subterm convergent equational theories (which allow to model classical cryptographic primitives such as symmetric and asymmetric encryption, digital signatures, hash functions) but also for blind signatures and trapdoor commitment, as all of these are optimally reducing and have therefore the strong finite variant property.

Our procedure is based on a fully-abstract modeling of each symbolic trace of a process into a set of Horn clauses. This modeling of symbolic traces into Horn clauses extends the Horn clauses in Chapter 4 by taking into account the dynamic behavior of the symbolic trace. We use strongly complete sets of variants and complete sets of equational unifiers to get rid of the underlying equational theory. Therefore the Horn clauses are interpreted in the free term algebra. A carefully crafted refinement of resolution is then used to saturate the set of clauses. From the saturated set of clauses, it is easy to check if two symbolic traces are equivalent. Furthermore, if two determinate processes are given as sets of symbolic traces, we show how to check from the respective saturated sets of clauses if the two processes are trace equivalent.

However, many electronic voting protocols are not determinate and therefore we cannot directly apply this result to reason about them. However, in Section 5.9.1 we present a method for proving that two nondeterminate processes are equivalent, method which uses equivalence checking for two symbolic traces as a subroutine. This allows us to prove (Section 5.9.3) vote privacy of the FOO [87] e-voting protocol. This is the first provably sound and fully automated proof of vote privacy for this protocol. However, our method is incomplete in that two processes could be trace equivalent but the methods fails to prove it.

As we have already explained, our procedure for automatically verifying trace equivalence is based on the saturation of a set of Horn clauses. We conjecture that the saturation always terminates for subterm convergent rewrite systems. Unfortunately, due to the highly complex form of the structures involved in the procedure, we have not been able to show this. However, we have implemented the procedure in the tool AKiSs and we have used it to prove vote privacy of the FOO [87] e-voting protocol. This is the first provably sound automatic proof of vote privacy for this protocol. An article based on this work was published in [35].

Related Work

Undecidability of observational equivalence in the spi calculus, even for the finite control fragment, was shown by Hüttel [97]. In the same paper, a decision result is given for the finite, replication-free, fragment of the spi calculus [97] for a fixed, limited set of cryptographic primitives. An automated tool [79] can check observational equivalence for the finite, replication-free fragment of the spi-calculus with a limited set of cryptographic primitives. Symbolic bisimulations have also been devised for the spi [30, 29] and applied pi calculus [73, 102] to avoid unbounded branching due to adversary inputs. However, only [73] and [30] yield an automated procedure for proving equivalence, but again only approximating observational equivalence. In [121], the authors present a decision procedure for checking open bisimulation for a variant of the spi-calculus with a fixed set of cryptographic primitives.

A highly non-deterministic decision procedure for checking equivalence of constraint systems (or equivalently, of two symbolic traces) is given by Baudet [19]. An alternate procedure achieving

the same goal was proposed by Chevalier and Rusinowitch [44]. Due to the high degree of non-determinism involved, it seems that the two procedures are not implementable. Therefore, Cheval *et al.* [37] have designed a new procedure and a prototype tool to decide the equivalence of constraint systems, but only for a fixed set of primitives. For a restricted class of *simple* processes it was shown [59] that observational equivalence coincides with trace equivalence and that the later is decidable by using as a subroutine a decision procedure for checking equivalence of constraint systems. The procedure for constraint systems in [37] was generalized to trace equivalence [38] for finite processes with a fixed set of primitives, but the new procedure has not been implemented.

ProVerif [25] can prove strong secrecy [26] and diff-equivalence [27], an equivalence between processes that differ only in their choice of terms and which is stronger than observational equivalence. However, ProVerif cannot verify vote privacy of the FOO electronic voting protocol except with the use of the add-on ProSwapper [119] which was not shown to be sound. ProVerif translated protocols into Horn clauses and employs sound abstractions that allow it to prove equivalences for an unbounded number of sessions. However, it may fail to terminate and because of the abstractions it can produce false attacks.

Other approaches [123, 89] based on Horn clauses also perform abstractions and allow for false attacks. For the case of confidentiality properties, precise encodings (without false attacks) into Horn clauses have been proposed and implemented in [6]. However, the encoding is for a fixed set of cryptographic primitives, it does not take into account that processes may block (due to a test failing) and it works only for confidentiality.

1.5.4 Composability

In Chapter 6, we establish a composition result with respect to confidentiality properties. We show that two protocols which use disjoint cryptographic primitives modeled by arbitrary disjoint equational theories can be securely composed as soon as they do not reveal shared secrets and when the shared secrets are not reused. This is the first composition result which allows to make use of arbitrary cryptographic primitives in the active case, all previous results assuming a limited fixed set of primitives or a tagging scheme.

Our work allows us for example to securely compose a key-establishment protocol with another protocol using the established key, as long as the two protocols use disjoint cryptographic primitives. This also holds in the case where key-establishment is performed several times and the respective keys are used. The composition result holds for an unbounded number of sessions. We also show that employing a form of tagging is sufficient to ensure disjointness in the case of encryption and hash functions. We show this by reducing the composition of the tagged protocols to the composition of two protocols which used disjoint equational theories.

One limitation of our approach is that it cannot cope with dishonest agents due to the hypothesis that shared secrets (between the two protocols which are to be composed) are not revealed. Dishonest users would reveal such secrets immediately and therefore our composition result does not give any guarantee in this case. The work in this chapter has been presented at CSF 2010 [49].

Related Work

One of the first papers studying the composition of protocols in the symbolic model is [93]. In the formalism of strand spaces [85], Guttman and Thayer show that two protocols which make use of concatenation and encryption can be safely executed together without damaging interactions, as soon as the protocols are "independent". In [57], Cortier *et al* show that tagging is sufficient to avoid collusion between protocols sharing common keys and making use of standard cryptographic primitives: concatenation, signature, hash functions and encryption. This framework allows to compose processes symmetrically; however, it does not allow to securely compose e.g. a key exchange protocol with another protocol which makes use of the shared key. In [92], Guttman provides a characterization which ensures that two protocols can run securely together when sharing some data such as keys or payloads. The main improvement over [93] is that keys are allowed to be non-atomic.

In [71], Delaune *et al.* use a derivative of the applied-π calculus to model off-line guessing attacks. They show that in the passive case resistance against guessing attacks is preserved by the composition of two protocols which share the weak secret against which the attack is mounted. This result (in the passive case) holds for arbitrary equational theories. However, for the active case this is no longer the case: it is however proven that *tagging* the weak secret enforces secure composition (in the sense of guessing attacks). This framework applies to parallel composition only.

Mödersheim and Viganò [109] have proposed a framework for composing protocols sequentially. In [70], Delaune *et al.* use a simulation based approach inspired from the computational model to provide a framework for securely composing protocols in the applied-π calculus. This involves defining for each sub-protocol an *ideal functionality* and then showing that a certain implementation securely emulates the ideal functionality. In [91], Groß and Mödersheim introduce *vertical* protocol compositions, where a key-exchange protocol is coupled with a protocol (called application protocol) which uses the exchanged key. Vertical compositions are similar to the sequential compositions we have used in this work. Another line of work is represented by the Protocol Composition Logic [67], which can be used to modularly prove security properties of protocols using a fixed set of primitives; however, the proofs may be rather involved and are not automatic.

1.6 Research Publications

The results obtained in this thesis have been partially published:

Journals

- Ş. Ciobâcă, S. Delaune and S. Kremer. Computing knowledge in security protocols under convergent equational theories. Journal of Automated Reasoning, 2011.

Conferences

- Ş. Ciobâcă and V. Cortier. Protocol composition for arbitrary primitives. In Proceedings of the 23rd IEEE Computer Security Foundations Symposium (CSF'10), Edinburgh, Scotland, UK, July 2010, pages 322-336. IEEE Computer Society Press.
- Ş. Ciobâcă, S. Delaune and S. Kremer. Computing knowledge in security protocols under convergent equational theories. In Proceedings of the 22nd International Conference on Automated Deduction (CADE'09), Montreal, Canada, August 2009, LNAI, pages 355-370. Springer.
- R. Chadha, Ş. Ciobâcă and S. Kremer. Automated verification of equivalence properties of cryptographic protocols. In Proceedings of the 21st European Symposium on Programming (ESOP 2012).

Other

- Ş. Ciobâcă. Computing finite variants for subterm convergent rewrite systems. In Proceedings of the 25th International Workshop on Unification (UNIF'11), Wroclaw, Poland, July 2011.

Research Tools

The implementations of the tools that were built as part of our research are available online:

- Ş. Ciobâcă. KiSs, a tool for deciding static equivalence and deduction under a class of convergent equational theories (`http://www.lsv.ens-cachan.fr/~ciobaca/kiss`)

- Ş. Ciobâcă. SubVariant, a tool for computing finite, complete sets of variants under optimally reducing convergent equational theories (`http://www.lsv.ens-cachan.fr/~ciobaca/subvariant`)
- Ş. Ciobâcă. AKiSs, a tool for automatically verifying trace equivalence of determinate, bounded security protocols without else branches under convergent term rewriting systems having the strong finite variant property (`http://www.lsv.ens-cachan.fr/~ciobaca/akiss`)

1.7 Thesis Plan

Chapter 2 introduces notations and concepts standard in the literature. In Chapter 3, we introduce the notion of strongly complete set of variants of a term, we discuss its relation to the notion of complete set of variants, and we show that *optimally reducing* term rewriting systems have finite strongly complete sets of variants. Furthermore, we show how to construct finite complete sets of unifiers for equational unification problems from the strongly complete sets of variants of the terms involved. In Chapter 4, we discuss a decision procedure for the intruder deduction problem and for the static equivalence problem for a class of convergent equational theories including subterm convergent rewrite systems, trapdoor commitment and blind signatures. In Chapter 5, we give a procedure for verifying indistinguishability between finite number of sessions of security protocols. We have implemented the procedure and we have used the implementation, among others, to prove vote privacy of the FOO [87] e-voting protocol. In Chapter 6, we establish a composition result with respect to confidentiality properties. We show that two protocols which use disjoint cryptographic primitives modeled by arbitrary disjoint equational theories can be securely composed as soon as they do not reveal shared secrets and when the shared secrets are not reused. We also show that employing a form of tagging is sufficient to ensure disjointness in the case of encryption and hash functions. Chapter 7 concludes the thesis with a summary of the results obtained and presents perspectives and possible future work.

Chapter 2

Preliminaries

In this chapter, we review several standard concepts and definitions in term algebra and term rewriting theory.

2.1 Term Algebra

Let $\mathcal{F}$ be a *signature*, i.e. a finite set of *function symbols*. Let ar be a function associating to each function symbol a natural number, called the *arity* of the symbol. A function of arity 0 is called a *constant*.

We let $\mathcal{X}, \mathcal{N}, \mathcal{M}, \mathcal{W}, \mathcal{C}$ be pairwise disjoint, countably infinite sets of *variables*, *private names*, *public names*, *parameters* and *public channels*, respectively. We assume furthermore that $\mathcal{X}, \mathcal{N}, \mathcal{M}$, $\mathcal{W}$ and $\mathcal{C}$, which will act as *atoms* in the term algebra, are disjoint from $\mathcal{F}$. We will note by $\mathcal{A}$ the entire set of atoms ($\mathcal{A} = \mathcal{X} \cup \mathcal{N} \cup \mathcal{M} \cup \mathcal{W} \cup \mathcal{C}$).

Given a signature $\mathcal{F}'$ that is a subset of $\mathcal{F}$ ($\mathcal{F}' \subseteq \mathcal{F}$) and a set of atoms $\mathcal{A}'$ that is a subset of the global set of atoms ($\mathcal{A}' \subseteq \mathcal{A}$), we define the set $T(\mathcal{F}', \mathcal{A}')$ of all $\mathcal{F}'$*-terms* over $\mathcal{A}'$ as the smallest set such that:

1. all atoms in $\mathcal{A}'$ are terms ($\mathcal{A}' \subseteq T(\mathcal{F}', \mathcal{A}')$),
2. for all function symbols $f \in \mathcal{F}'$ of arity $\mathsf{ar}(f) = k$ and all terms $t_1, \ldots, t_k \in T(\mathcal{F}', \mathcal{A}')$, we have that $f(t_1, \ldots, t_k) \in T(\mathcal{F}', \mathcal{A}')$ (i.e. application of function symbols to terms yields terms).

The set of *positions* of a term $t \in T(\mathcal{F}, \mathcal{A})$ is a set $\mathcal{P}os(t)$ of strings over the alphabet of positive integers, defined inductively such that:

1. if $t \in \mathcal{A}$, then $\mathcal{P}os(t) = \{\epsilon\}$, where ϵ denotes the empty string, and
2. if $t = f(t_1, \ldots, t_k)$ with $f \in \mathcal{F}$, then

$$\mathcal{P}os(t) = \{\epsilon\} \cup \bigcup_{i=1}^{k} \{ip \mid p \in \mathcal{P}os(t_i)\}.$$

where juxtaposition denotes string concatenation. The *size* $|t|$ of the term t, is the cardinality of $\mathcal{P}os(t)$.

For every position $p \in \mathcal{P}os(t)$ of the term t, the *subterm* $t|_p$ of the term t at the position p is defined by induction on the length of p to be:

$$\begin{aligned} t|_\epsilon &= t \\ f(t_1, \ldots, t_k)|_{iq} &= t_i|_q \end{aligned}$$

For all terms $s \in T(\mathcal{F}, \mathcal{A})$ and for every position $p \in \mathcal{P}os(t)$ of the term t, we denote by $t[s]_p$ the term that is obtained by placing the term s at position p in the term t, i.e.

$$\begin{array}{rcl} t[s]_\epsilon & = & s \\ f(t_1, \ldots, t_k)[s]_{iq} & = & f(t_1, \ldots, t_{i-1}, t_i[s]_q, t_{i+1}, \ldots, t_k) \end{array}$$

The *set of subterms* $\mathsf{st}(t)$ of the term t is the set

$$\mathsf{st}(t) = \{s \mid \text{ there exists } p \in \mathcal{P}os(t) \text{ such that } t|_p = s\}.$$

The set $\mathcal{V}ar(t)$ is the set of variables of the term t, i.e.

$$\mathcal{V}ar(t) = \{x \mid \text{there exists } p \in \mathcal{P}os(t) \text{ such that } t|_p = x \in \mathcal{X}\}.$$

The set $\mathcal{N}ames(t)$ is the set of names of the term t, i.e.

$$\mathcal{N}ames(t) = \{n \mid \text{there exists } p \in \mathcal{P}os(t) \text{ such that } t|_p = a \in \mathcal{N} \cup \mathcal{M}\}.$$

The set $\mathcal{S}ymbols(t)$ is the set of function symbols of the term t, i.e.

$$\mathcal{S}ymbols(t) = \{f \mid \text{there exists } p \in \mathcal{P}os(t) \text{ and } t_1, \ldots, t_n \text{ such that } t|_p = f(t_1, \ldots, t_n)\}.$$

We extend the functions $\mathcal{V}ar$, $\mathcal{N}ames$ and $\mathcal{S}ymbols$ on tuples of terms, sets of terms and combinations thereof in the expected manner. We say that the term t is *ground* if $\mathcal{V}ar(t) = \emptyset$.

A *substitution* is a function $\sigma : \mathcal{X} \mapsto T(\mathcal{F}, \mathcal{A})$ such that $\sigma(x) \neq x$ for only finitely many variables $x \in \mathcal{X}$. The finite set of variables $\mathcal{D}om(\sigma) = \{x \mid \sigma(x) \neq x\}$ is called the *domain* of the substitution σ. The *range* $\mathcal{R}ange(\sigma)$ of the substitution σ is the set $\mathcal{R}ange(\sigma) = \{\sigma(x) \mid x \in \mathcal{D}om(\sigma)\}$. If $\mathcal{D}om(\sigma) \subseteq \{x_1, \ldots, x_n\}$, we may write σ as

$$\sigma = \{x_1 \mapsto \sigma(x_1), \ldots, x_n \mapsto \sigma(x_n)\}.$$

The homomorphic extension of a substitution σ to the set of terms $T(\mathcal{F}, \mathcal{A})$ is the function $\hat{\sigma} : T(\mathcal{F}, \mathcal{A}) \to T(\mathcal{F}, \mathcal{A})$, where

$$\begin{array}{rcll} \hat{\sigma}(x) & = & \sigma(x) & \text{for all variables } x \in \mathcal{X} \\ \hat{\sigma}(a) & = & a & \text{for all non-variable atoms } a \in \mathcal{A} \setminus \mathcal{X} \\ \hat{\sigma}(f(t_1, \ldots, t_k)) & = & f(\hat{\sigma}(t_1), \ldots, \hat{\sigma}(t_k)) & \text{for all other terms } f(t_1, \ldots, t_k). \end{array}$$

We identify as usual substitutions with their homomorphic extensions and we may write substitution application in suffix form: the term $t\sigma$ is the application of the homomorphic extension $\hat{\sigma}$ of the substitution σ to the term t, i.e. $t\sigma = \hat{\sigma}(t)$.

The *identity substitution* $\sigma = \{\}$ is the only substitution with empty domain $\mathcal{D}om(\sigma) = \{\}$ and, when applied to a term t, it does not change it: $t = t\{\}$. If $X \subseteq \mathcal{X}$ is a set of variables and σ is a substitution, then $\sigma[X]$ is the restriction of σ to X, i.e. the substitution such that $\sigma[X](x) = \sigma(x)$ if $x \in X$ and $\sigma[X](x) = x$ if $x \in \mathcal{X} \setminus X$. The restriction operator enjoys the following property, which we are going to use several times later on:

Lemma 2.1. *Let t be a term and let σ be a substitution. If $X \supseteq \mathcal{V}ar(t)$ then $t\sigma = t(\sigma[X])$.*

We let $\uplus$ denote the disjoint union of two sets. In particular, if σ and τ are two substitutions with disjoint domains ($\mathcal{D}om(\sigma) \cap \mathcal{D}om(\tau) = \emptyset$), we denote by $\gamma = \sigma \uplus \tau$ the substitution of domain $\mathcal{D}om(\gamma) = \mathcal{D}om(\sigma) \cup \mathcal{D}om(\tau)$ such that $\gamma(x) = \sigma(x)$ if $x \in \mathcal{D}om(\sigma)$ and $\gamma(x) = \tau(x)$ if $x \in \mathcal{X} \setminus \mathcal{D}om(\sigma)$. The *composition* $\sigma\tau$ of two substitutions σ and τ is defined to be the substitution which associates to any variable x the term $(x\sigma)\tau$, i.e. $\sigma\tau(x) = \hat{\tau}(\sigma(x))$. Substitution composition enjoys the following property, which we are going to use several times later on:

Lemma 2.2. *Let $\sigma_1, \sigma_2, \sigma_3, \sigma_4$ be substitutions such that $\sigma_1[X] = (\sigma_2\sigma_3)[X]$ and $\sigma_3[X \cup \mathcal{V}ar(\mathcal{R}ange(\sigma_2))] = \sigma_4[X \cup \mathcal{V}ar(\mathcal{R}ange(\sigma_2))]$ for some set of variables X. Then $\sigma_1[X] = (\sigma_2\sigma_4)[X]$.*

Proof. Let $x \in X$ be an arbitrary variables in the set X. If $x \in \mathcal{D}om(\sigma_2)$, we have that $\mathcal{V}ar(\sigma_2(x)) \subseteq \mathcal{V}ar(\mathcal{R}ange(\sigma_2))$. If $x \notin \mathcal{D}om(\sigma_2)$, we have that $\mathcal{V}ar(\sigma_2(x)) = \mathcal{V}ar(x) = \{x\} \subseteq X$. In either case, we have that $\mathcal{V}ar(\sigma_2(x)) \subseteq X \cup \mathcal{V}ar(\mathcal{R}ange(\sigma_2))$. Therefore

$$\begin{array}{rcl}(\sigma_3[X \cup \mathcal{V}ar(\mathcal{R}ange(\sigma_2))])(\sigma_2(x)) & = & (\sigma_4[X \cup \mathcal{V}ar(\mathcal{R}ange(\sigma_2))])(\sigma_2(x)) \\ & = & x\sigma_2(\sigma_4[X \cup \mathcal{V}ar(\mathcal{R}ange(\sigma_2))]).\end{array}$$

As $\mathcal{V}ar(x\sigma_2) \subseteq X \cup \mathcal{V}ar(\mathcal{R}ange(\sigma_2))$, it follows by Lemma 2.1 that $x\sigma_2(\sigma_4[X \cup \mathcal{V}ar(\mathcal{R}ange(\sigma_2))] = x\sigma_2\sigma_4$.

We have shown that $x\sigma_2\sigma_3 = x\sigma_2\sigma_4$. As $x \in X$ was chosen arbitrarely in X, it follows that $\sigma_2\sigma_3[X] = \sigma_2\sigma_4[X]$. But $\sigma_1[X] = \sigma_2\sigma_3[X]$ and therefore, by transitivity, $\sigma_1[X] = \sigma_2\sigma_4[X]$.

□

A substitution σ is called a *variable renaming* if $\mathcal{D}om(\sigma) = \mathcal{R}ange(\sigma)$. We say that σ is a variable renaming from the set of variables $X \subseteq \mathcal{X}$ to the set of variables $Y \subseteq \mathcal{X}$ if $\mathcal{D}om(\sigma) \subseteq X \cup Y$ and $\sigma(x) \in Y$ for all variables $x \in X$.

If $\mathcal{A}_0$ is a set of atoms, a *replacement* of terms is a function $\sigma : D \to T(\mathcal{F}, \mathcal{A}_0)$ such that D is a finite set of terms $D \subseteq T(\mathcal{F}, \mathcal{A}_0)$. The *application of a replacement* $\sigma : D \to T(\mathcal{F}, \mathcal{A}_0)$ to a term t is written in suffix notation $t\sigma$ and is defined inductively to be:

$$\begin{array}{rcll} t\sigma & = & \sigma(t) & \text{if the term } t \in D \text{ is in the domain } D \text{ of } \sigma \\ t\sigma & = & a & \text{if the term } t = a \in \mathcal{A} \setminus D \text{ is an atom not in the domain } D \text{ of } \sigma \\ t\sigma & = & f(t_1\sigma, \ldots, t_k\sigma) & \text{if the term } t = f(t_1, \ldots, t_k) \notin D \text{ is not in the domain of } \sigma \end{array}$$

A (k-)context $C \in T(\mathcal{F}, \mathcal{A} \cup \{\_^1, \ldots, \_^k\})$ is a term with distinguished variables $\_^1, \ldots, \_^k$ such that each variable $\_^i$ appears at most once in C. We write $C[t_1, \ldots, t_k]$ for the term $C\{\_^1 \mapsto t_1, \ldots, \_^k \mapsto t_k\}$.

2.1.1 Unification

Two terms $s, t \in T(\mathcal{F}, \mathcal{A})$ are (syntactically) *unifiable* if there exists a substitution σ such that $s\sigma = t\sigma$. The substitution σ is then a *unifier* of s and t. A unifier σ of s and t is called more general than another unifier τ if there exists a substitution ω such that $\tau = \sigma\omega$.

It is well known [15] that any two unifiable terms s and t admit a *most general unifier* $\mathsf{mgu}(s, t)$, i.e. a unifier which is more general than any other unifier. Furthermore, any two most general unifiers σ and τ of two unifiable terms s and t can be obtained from each other by composition with a variable renaming. We make therefore a slight abuse of language and speak of *the* most general unifier $\mathsf{mgu}(s, t)$ of s and t (which is unique up to variable renaming), as is common in term rewriting literature.

2.2 Rewriting

2.2.1 Equational Theories

An *identity* over $\mathcal{F}' \subseteq \mathcal{F}$ is a pair $(s, t) \in T(\mathcal{F}', \mathcal{X}) \times T(\mathcal{F}', \mathcal{X})$, where $\times$ denotes the cartesian product of sets. We will write identities $(s, t) \in T(\mathcal{F}, \mathcal{X}) \times T(\mathcal{F}, \mathcal{X})$ as $s \approx t$.

If $\mathsf{E} = \{s_1 \approx t_1, \ldots, s_n \approx t_n\}$ is a set of identities, we will denote by $=_\mathsf{E}$ the smallest relation on terms containing E and such that:

1. $=_\mathsf{E}$ is reflexive: $t =_E t$ for all terms $t \in T(\mathcal{F}, \mathcal{A})$,
2. $=_\mathsf{E}$ is transitive: $s =_E r$ and $r =_E t$ implies $s =_E t$ for all terms $s, r, t \in T(\mathcal{F}, \mathcal{A})$,
3. $=_\mathsf{E}$ is symmetric: $s =_E t$ implies $t =_E s$ for all terms $s, t \in T(\mathcal{F}, \mathcal{A})$,

4. $=_{\mathsf{E}}$ is closed under substitutions: $s =_E t$ implies $s\sigma =_E t\sigma$ for all terms $s, t \in T(\mathcal{F}, \mathcal{A})$ and all substitutions σ and

5. $=_{\mathsf{E}}$ is closed under $\mathcal{F}$-operations: $s_1 =_E t_1, \ldots, s_k =_{\mathsf{E}} t_k$ implies $f(s_1, \ldots, s_k) =_{\mathsf{E}} f(t_1, \ldots, t_k)$ for all function symbols f of arity $\mathsf{ar}(f) = k$ and for all terms $s_1, \ldots, s_k, t_1, \ldots, t_k \in T(\mathcal{F}, \mathcal{A})$.

The relation $=_{\mathsf{E}}$ is called the *equational theory* generated by the set of identities E. We will sometimes make an abuse of language and call E an equational theory.

We write $s \rightarrow_{\mathsf{E}} t$ if there exist a context C, an identity $s_i \approx t_i \in \mathsf{E}$ and a substitution σ such that $s = C[s_i\sigma]$ and $t = C[t_i\sigma]$ or vice-versa, if $s = C[t_i\sigma]$ and $t = C[s_i\sigma]$. It is well known that $\rightarrow^*_{\mathsf{E}}$, the transitive and reflexive closure of $\rightarrow_{\mathsf{E}}$, coincides with $=_{\mathsf{E}}$ [14]. Two substitutions σ and τ are equal modulo E, written $\sigma =_{\mathsf{E}} \tau$, if $\sigma(x) =_{\mathsf{E}} \tau(x)$ for all $x \in \mathcal{X}$.

Equational Unification

Two terms $s, t \in T(\mathcal{F}, \mathcal{A})$ are *unifiable in the equational theory* E (or E-*unifiable*) if there exists a substitution σ such that $s\sigma =_{\mathsf{E}} t\sigma$. The substitution σ is then an E-*unifier* of s and t. A set of substitutions $\mathsf{mgu}_{\mathsf{E}}(s, t)$ is called a *complete set of* E-*unifiers* of s and t if:

1. any substitution $\sigma \in \mathsf{mgu}_{\mathsf{E}}(s, t)$ is an E-unifier of s and t: $s\sigma =_{\mathsf{E}} t\sigma$ and

2. any E-unifier τ of s and t is an instance modulo E of some substitution $\sigma \in \mathsf{mgu}_{\mathsf{E}}(s, t)$ on the variables of s and t: there exist a substitution $\sigma \in \mathsf{mgu}_{\mathsf{E}}(s, t)$ and a substitution ω such that $\tau[\mathcal{V}ar(s, t)] =_{\mathsf{E}} \sigma\omega[\mathcal{V}ar(s, t)]$.

An equational theory is called *finitary* if, for any two terms $s, t \in T(\mathcal{F}, \mathcal{A})$ there exists a finite complete set of E-unifiers ($\mathsf{mgu}_{\mathsf{E}}(s, t)$ is finite).

2.2.2 Rewrite Systems

A *rewrite rule* is an identity $l \approx r$ such that $l \notin \mathcal{X}$ and $\mathcal{V}ar(r) \subseteq \mathcal{V}ar(l)$ [1]. In this case we may write $l \rightarrow r$ instead of $l \approx r$. A *term rewriting system* $R = \{l_1 \rightarrow r_1, \ldots, l_n \rightarrow r_n\}$ is a finite set of rewrite rules. We write $s \rightarrow_{\mathsf{R}} t$ if there exist a position $p \in \mathcal{P}os(s)$, a rewrite rule $l \rightarrow r \in R$ and a substitution σ of domain $\mathcal{D}om(\sigma) \subseteq \mathcal{V}ar(l)$ such that $s|_p = l\sigma$ and $t = s[r\sigma]_p$. The relation $\rightarrow_{\mathsf{R}}$ is then called the *one-step rewrite relation* of R, the reflexive transitive closure of $\rightarrow_{\mathsf{R}}$ is denoted by $\rightarrow^*_{\mathsf{R}}$ and the reflexive transitive symmetric closure of $\rightarrow_{\mathsf{R}}$ is denoted $\leftrightarrow^*_{\mathsf{R}}$. We also write $=_{\mathsf{R}}$ instead of $\leftrightarrow^*_{\mathsf{R}}$.

A rewrite system R *implements* an equational theory E if the relations $=_{\mathsf{R}}$ and $=_{\mathsf{E}}$ coincide. If $R = \{l_1 \rightarrow r_1, \ldots, l_n \rightarrow r_n\}$ is a term rewriting system and $\mathsf{E} = \{l_1 \approx r_1, \ldots, l_n \approx r_n\}$ is the associated equational theory, it is known that R implements E (see for example [14]).

A rewrite system R is *terminating* if there is no infinite rewrite chain $t_1 \rightarrow_{\mathsf{R}} t_2 \rightarrow_{\mathsf{R}} \ldots$. A rewrite system R is *confluent* if for all terms s, u, v such that $s \rightarrow^*_{\mathsf{R}} u$ and $s \rightarrow^*_{\mathsf{R}} v$ we have that there exists a term t such that $u \rightarrow^*_{\mathsf{R}} t$ and $v \rightarrow^*_{\mathsf{R}} t$. A rewrite system is *convergent* if it is both terminating and confluent. An equational theory is *convergent* if it is implementable by a convergent rewrite system. A term t is *in normal form* or *normalized* (with respect to a rewrite system R) if $t \not\rightarrow_{\mathsf{R}} s$ for any term s. A term t is a normal form of a term s (with respect to a rewrite system R) if t is in normal form with respect to R and $s \rightarrow^*_{\mathsf{R}} t$. It is well known (see e.g. [14]) that if R is a convergent rewrite system, any term t has a unique normal form with respect to R. In this case (when R is convergent) we will denote with $t{\downarrow_{\mathsf{R}}}$ the normal form of t.

We say that a rewrite rule $l \rightarrow r$ is in R up to renaming, and we write $l \rightarrow r \bar{\in} \mathsf{R}$ if there exist a rewrite rule $l' \rightarrow r' \in \mathsf{R}$ and a variable renaming τ such that $l = l'\tau$ and $r = r'\tau$.

If σ is a substitution, we will denote by $\sigma{\downarrow_{\mathsf{R}}}$ the substitution which assigns to any variable x the term $(x\sigma){\downarrow}$. We say that a substitution σ is in normal form (with respect to R) if $\sigma = \sigma{\downarrow_{\mathsf{R}}}$. If R is clear from the context, we sometimes omit R and use $\downarrow$ instead of $\downarrow_{\mathsf{R}}$.

[1] these restrictions on rewrite rules follow the lines of [14]. Rewrite rules not obeying them are necessarily non-terminating and therefore not very interesting.

Subterm Convergent Rewrite Systems

Definition 2.1. A term rewriting system R is *subterm convergent* if it is convergent and every rewrite rule $l \to r \in \mathsf{R}$ is such that

subterm rule $r \in \mathsf{st}(l)$ or

extended rule r is a ground term in normal form with respect to R.

The first type of rewrite rule justifies the name of this class of rewrite systems and was originally the only type of rewrite rule allowed [1] in subterm convergent rewrite systems. The second type of rule represents a small but useful extension introduced in [21]. In this thesis, we consider the extended version of subterm convergent rewrite systems as defined above. We say that an equational theory E is subterm convergent if there is a subterm convergent rewrite system R that implements the equational theory E. Subterm convergent rewrite systems allow the modeling of many usual cryptographic primitives such as symmetric encryption:

Example 2.1. Let $\mathsf{R} = \{\mathrm{dec}(\mathrm{enc}(x, y), y) \to x\}$. It is easy to see that R is subterm convergent. The function symbol enc models symmetric encryption and the function symbol dec symmetric decryption. The only rewrite rule models the fact that decryption cancels out encryption when the correct key is used.

Optimally Reducing Rewrite Systems

Definition 2.2. A rewrite rule $l \to r \in \mathsf{R}$ is called *optimally reducing* if, for any substitution θ such that all strict subterms of $l\theta$ are in normal form, we have that $r\theta$ is in normal form. A term rewriting system R is *optimally reducing* if all rewrite rules $l \to r \in \mathsf{R}$ are optimally reducing.

Optimally reducing term rewriting systems were introduced in [110]. It is easy to see that any subterm convergent rewrite system is an optimally reducing rewrite system. Additionally, optimally reducing rewrite systems allow to model cryptographic primitives which cannot be modeled by subterm convergent rewrite systems, as shown in the following example.

Example 2.2. The term rewriting system R defined bellow is convergent and optimally reducing. However, R is not subterm convergent because of its second rewrite rule.

$$\mathsf{R} = \left\{ \begin{array}{l} \mathrm{check}(\mathrm{sign}(x, y), \mathrm{pk}(y)) \to x \\ \mathrm{unblind}(\mathrm{sign}(\mathrm{blind}(x, y), z), y) \to \mathrm{sign}(x, z) \end{array} \right\}$$

The term rewriting system R models *blind signatures*, especially useful in modeling electronic voting protocols [72].

The function symbol sign represents the signing algorithm: the term $\mathrm{sign}(s, t)$ denotes the signature of the message t with the private key t. The function symbol check allows to verify a signature using the public key $\mathrm{pk}(y)$ of the party having signed the message.

The blind function symbol allows a party to "blind" a message x with a blinding factor y, thereby obtaining the blob $\mathrm{blind}(x, y)$. The blob can then be signed by a party using the private key z to obtain $\mathrm{sign}(\mathrm{blind}(x, y), z)$ and then, using the unblinding function symbol unblind, to obtain via the second rewrite rule the message $\mathrm{sign}(x, z)$. This allows the party that has blinded the message to obtain a signature of x without revealing what x is. This is particularly useful in voting protocols, where a voter can have his vote signed by the voting authority without revealing the vote itself [72].

Free symbols

A function symbol $f \in \mathcal{F}$ is *free* with respect to a rewrite system $R = \{l_1 \to r_1, \ldots, l_n \to r_n\}$ if the function symbol $f \notin \mathcal{S}ymbols(l_1, r_1, \ldots, l_n, r_n)$ does not appear in any of the identities of R. We have that:

Theorem 2.1. *If* R *is a convergent rewrite system and* $f \in \mathcal{F}$ *is a free function symbol of arity* $\mathsf{ar}(f) = k$*, then*

$$f(t_1, \ldots, t_k)\downarrow = f(t_1\downarrow, \ldots, t_k\downarrow).$$

Proof. We will first show by induction on the number of rewrite steps that if $f(t_1, \ldots, t_k) \to_{\mathsf{R}}^* t$, then $t = f(s_1, \ldots, s_k)$ for some terms $s_1, \ldots, s_k$ such that $t_1 \to_{\mathsf{R}}^* s_1, \ldots, t_k \to_{\mathsf{R}}^* s_k$. Indeed, for the base case, if the number of rewrite steps is 0, we choose $s_1 = t_1, \ldots, s_k = t_k$ and we have that $t_1 \to_{\mathsf{R}}^* s_1, \ldots, t_k \to_{\mathsf{R}}^* s_k$ as $\to_{\mathsf{R}}^*$ is reflexive.

For the inductive case, we assume by the induction hypothesis that $f(t_1, \ldots, t_k) \to_{\mathsf{R}}^* f(s'_1, \ldots, s'_k)$ such that $t_i \to_{\mathsf{R}}^* s'_i$ for all $1 \leq i \leq k$ and we show that if $f(s'_1, \ldots, s'_k) \to_{\mathsf{R}} t$, then $t = f(s_1, \ldots, s_k)$ such that $t_i \to_{\mathsf{R}}^* s_i$ for all $1 \leq i \leq k$. As $f(s'_1, \ldots, s'_k) \to_{\mathsf{R}} t$, it follows that there exists a rewrite rule $l \to r \in \mathsf{R}$, a position $p \in \mathcal{P}os(t)$ and a substitution σ such that $f(s'_1, \ldots, s'_k)|_p = l\sigma$ and $t = f(s'_1, \ldots, s'_k)[r\sigma]_p$. As f is free with respect to R, it follows that $f \notin \mathcal{S}ymbols(l)$. Furthermore, by the definition of a rewrite system, $l \notin \mathcal{X}$. Therefore l and $f(s'_1, \ldots, s'_k)$ are not unifiable. Therefore $p \neq \epsilon$ cannot be the empty position. Therefore $p = jq$ for some index $1 \leq j \leq k$ and some position q.

We have therefore that $t = f(s'_1, \ldots, s'_k)[r\sigma]_p = f(s'_1, \ldots, s'_k)[r\sigma]_{jq}$ is equal, by the definition of $\cdot[\cdot]\cdot$, to $f(s'_1, \ldots, s'_{j-1}, s'_j[r\sigma]_q, s'_{j+1}, \ldots, s'_k)$. Let $s_i = s'_i$ for all $1 \leq i \leq k$ such that $i \neq j$ and let $s_j = s'_j[r\sigma]_q$. By the definition of $\to_{\mathsf{R}}$, we have that $s'_j \to_{\mathsf{R}} s_j$. By reflexivity of $\to_{\mathsf{R}}^*$, we have that $s'_i \to_{\mathsf{R}}^* s_i$ for all $1 \leq i \leq k$ if $i \neq j$. Therefore $s'_i \to_{\mathsf{R}}^* s_i$ for all $1 \leq i \leq k$. But $t_i \to_{\mathsf{R}}^* s'_i$ for $1 \leq i \leq k$ and, by transitivity of $\to_{\mathsf{R}}^*$, we have $t_i \to_{\mathsf{R}}^* s_i$ for all $1 \leq i \leq k$.

We have shown that $t = f(s_1, \ldots, s_k)$ such that $t_i \to_{\mathsf{R}}^* s_i$ for all $1 \leq i \leq k$, thereby concluding the inductive step.

We have just proved by induction on the number of rewrite steps that if $f(t_1, \ldots, t_k) \to_{\mathsf{R}}^* t$, then $t = f(s_1, \ldots, s_k)$ for some terms $s_1, \ldots, s_k$ such that $t_i \to_{\mathsf{R}}^* s_i$ for all $1 \leq i \leq k$. Furthermore, if $f(s_1, \ldots, s_k)$ is in normal form, we have that $s_1, \ldots, s_k$ are also in normal form. Therefore, if $f(s_1, \ldots, s_k)$ is the normal form of $f(t_1, \ldots, t_k)$, it follows that s_i is the normal form of t_i for all $1 \leq i \leq k$. Therefore $f(t_1, \ldots, t_k)\downarrow = f(t_1\downarrow, \ldots, t_k\downarrow)$. □

2.3 Modeling Messages as Terms

In order to formally assess the security of cryptographic protocols, we use a mathematical model of such protocols. First of all, we model messages exchanged between the parties during the protocol run as terms in a term algebra. Cryptographic primitives such as encryption are modeled as function symbols and the cryptographic properties of the primitives are modeled as an equational theory.

> *Example* 2.3. We let $\mathcal{F} = \{\mathrm{enc}, \mathrm{dec}\}$, where $\mathsf{ar}(\mathrm{enc}) = \mathsf{ar}(\mathrm{dec}) = 2$. The term $\mathrm{enc}(m_1, m_2)$ represents the message denoting the encryption of the message modeled by the term m_1 with the key modeled by the term m_2.
>
> Let $\mathsf{E} = \{\mathrm{dec}(\mathrm{enc}(x, y), y) = x\}$ be an equational theory. We have that the term $\mathrm{dec}(\mathrm{enc}(m_1, m_2), m_3)$ represents the result of decrypting the message $\mathrm{enc}(m_1, m_2)$ by the key m_3. If $m_2 =_{\mathsf{E}} m_3$, we have that $\mathrm{dec}(\mathrm{enc}(m_1, m_2), m_3) =_{\mathsf{E}} m_1$ (the decryption succeeds).

Base data such as atomic keys are modeled as *names*. Names can be private (the set $\mathcal{N}$ is the set of private names) if the intruder does not *a priori* know them, or they can be public (the set $\mathcal{M}$ is the set of public names) in case the intruder knows them *a priori*.

Example 2.4. Continuing the previous example, a freshly generated key can be modeled as the term $k \in \mathcal{N}$ and the public name $yes \in \mathcal{M}$ can be used to model a well-known representation of the vote 'yes'.

The set of terms representing messages is defined to be $\mathsf{Messages} = T(\mathcal{F}, \mathcal{N} \cup \mathcal{M})$.

Example 2.5. Continuing the previous example, if we let $m_1 = yes$ and $m_2 = k$, we have that the term $\mathrm{enc}(m_1, m_2) = \mathrm{enc}(yes, k)$ represents the message constructed by encrypting the vote 'yes' with a fresh key k that is a priori unknown to the intruder.

2.4 Frames

We fix an enumeration $w_1, w_2, \ldots$ of the elements of the set $\mathcal{W}$. The atoms $w_1, w_2, \ldots$ are not used as building blocks in messages, but are used by the attacker to *point* to messages exchanged during the protocol. The parameter w_1 refers to the first message exchanged, w_2 to the second, etc. This suggests a way of modeling sequences of messages exchanged during a protocol:

Definition 2.3. A *frame* φ is a substitution $\{w_1 \mapsto t_1, \ldots, w_n \mapsto t_n\}$ where $t_i \in \mathsf{Messages}$ $(1 \leq i \leq n)$ are terms representing messages.

Note that in our definition, every frame with $|\mathcal{D}om(\varphi)| = n$ has $\mathcal{D}om(\varphi) = \{w_1, \ldots, w_n\}$.

2.4.1 Deduction

The adversary can use the messages learnt from the run of a protocol to construct new messages. Messages can be constructed by applying function symbols to already known messages and by referring to the parameters. The adversary knows a priori all public names. We denote by $\mathsf{Recipes}_{\mathcal{F}} = T(\mathcal{F}, \mathcal{M} \cup \mathcal{W})$ the set of *recipes* constructible over a given signature $\mathcal{F}$, i.e. terms used by the intruder to construct messages. If the signature $\mathcal{F}$ is obvious from context, we also write $\mathsf{Recipes}$ instead of $\mathsf{Recipes}_{\mathcal{F}}$. The relation between recipes and the resulting messages is modeled as the deducibility relation.

Definition 2.4. We say that *the term t is deducible from φ with a recipe r in the equational theory* E (written as $\varphi \vdash^{r}_{\mathsf{E},\mathcal{F}} t$) if $r \in \mathsf{Recipes}_{\mathcal{F}}$ and $r\varphi =_{\mathsf{E}} t$.

If the signature $\mathcal{F}$ is obvious from context, we write $\vdash_{\mathsf{E}}$ instead of $\vdash_{\mathsf{E},\mathcal{F}}$. If the equational theory is also obvious from context, it can be omitted. If the equational theory E is implemented by a rewrite system R, then we also write $\vdash_{\mathsf{R}}$ instead of $\vdash_{\mathsf{E}}$.

Example 2.6. Continuing the previous example, let $\varphi = \{w_1 \mapsto \mathrm{enc}(yes, k), w_2 \mapsto k\}$ be a frame representing the sequence of messages from a toy voting protocol. The first message exchanged during the protocol was the encryption of the vote 'yes' under a fresh key k. The second message is the value of the key that was used to protect the vote in the first message.

The fact that the intruder does not a priori know k is modeled by the fact that $\varphi \nvdash^{k} k$, since $k \in \mathcal{N}$ and therefore $k \notin \mathsf{Recipes}$. However, the intruder can deduce the value of k indirectly, by referring to w_2: $\varphi \vdash^{w_2} k$.

We also have that the intruder can get the value of the vote 'yes' either indirectly, as $\varphi \vdash^{\mathrm{dec}(w_1,w_2)} yes$ or directly, since $\varphi \vdash^{yes} yes$ (as $yes \in \mathcal{M}$, a public name, can be used in the recipe).

2.4.2 Static Equivalence

The deduction relation described previously is enough to model secrecy of data: we can model the fact that a private name s remains secret after the protocol run if $\varphi \not\vdash^r s$ for any recipe r, where φ represents the frame collecting all messages from the protocol. However, for more subtle properties, deduction is not enough.

> *Example* 2.7. Consider for example a voting protocol where the possible votes are 'yes' and 'no'. The values of any vote are a priori known to the intruder (e.g. $\varphi \vdash^{yes} yes$ for any frame φ).
>
> Therefore, the fact that the vote of an individual voter remains private cannot be modeled by the fact that the name *yes* (representing a 'yes' vote) or that the name $no \in \mathcal{M}$ (representing a 'no' vote) cannot be deduced from the frame.

Instead, to express the fact that the value of the vote is not revealed, we ask that the intruder is not be able to distinguish a run of the protocol where 'yes' was voted from a run of the protocol where 'no' was voted. This indistinguishability relation is modeled as static equivalence:

Definition 2.5. A *test* $r_1 \stackrel{?}{=} r_2$ *holds* in a frame φ (written $(r_1 = r_2)\varphi$) if $\varphi \vdash^{r_1} t$ and $\varphi \vdash^{r_2} t$ for some t, *i.e.*, r_1 and r_2 are recipes for the same term in φ. If $(r_1 = r_2)\varphi$, we sometimes say that $r_1 = r_2$ is an *identity* that holds in φ.

A frame φ_1 is *statically included* in φ_2 (written $\varphi_1 \sqsubseteq_s \varphi_2$) iff for all r_1, r_2 we have that $(r_1 = r_2)\varphi_1$ implies $(r_1 = r_2)\varphi_2$.

Two frames φ_1 and φ_2 are *statically equivalent* (written $\varphi_1 \approx_s \varphi_2$) iff $\varphi_1 \sqsubseteq_s \varphi_2$ and $\varphi_2 \sqsubseteq_s \varphi_1$.

> *Example* 2.8. Let $\varphi = \{w_1 \mapsto \mathrm{enc}(yes, k), w_2 \mapsto k\}$ and let $\varphi' = \{w_1 \mapsto \mathrm{enc}(no, k), w_2 \mapsto k\}$. The frame φ can represent the messages in a run of a protocol where 'yes' was voted, while φ' can represent the messages in a the run of the same protocol where 'no' was voted. We have that $\varphi \not\approx_s \varphi'$, since the following test
>
> $$\mathrm{dec}(w_1, w_2) \stackrel{?}{=} yes$$
>
> holds in φ but not in φ'. We can conclude that the privacy of the vote is not preserved, since the two frames are distinguishable.
>
> On the other hand, the frames $\varphi_a = \{w_1 \mapsto \mathrm{enc}(yes, k)\}$ and $\varphi_b = \{w_1 \mapsto \mathrm{enc}(no, k)\}$ are statically equivalent. Intuitively, this is because $k \in \mathcal{N}$ is not known to the intruder and therefore there is no way to "open" the encryption.

Chapter 3

The Strong Finite Variant Property

3.1 Introduction

Given a term (e.g. $t = \mathrm{dec}(x, y)$) and a convergent rewrite system (e.g. $\mathsf{R} = \{\mathrm{dec}(\mathrm{enc}(x, y), y) \to x\}$), we are interested in having a convenient symbolic representation of all normal forms $t\sigma\!\downarrow$ of instantiations $t\sigma$ of the term t. In the above case, the normal form $t\sigma\!\downarrow$ of $t\sigma$ falls into one of the following two cases:

1. either $\sigma(x)\!\downarrow = \mathrm{enc}(s, \sigma(y)\!\downarrow)$ is an encryption of some term s with the term $\sigma(y)\!\downarrow$, in which case $\sigma\!\downarrow[\{x, y\}] = \sigma_1\tau_1[\{x, y\}]$ for $\sigma_1 = \{x \mapsto \mathrm{enc}(z, y)\}$ and $\tau_1 = \{z \mapsto s, y \mapsto \sigma(y)\!\downarrow\}$ and therefore
$$t\sigma\!\downarrow = (t\sigma_1)\!\downarrow\tau_1.$$

2. or $\sigma(x)\!\downarrow$ is not such an encryption ($\sigma(x)\!\downarrow \neq \mathrm{enc}(s, \sigma(y)\!\downarrow)$ for any term s), in which case $\sigma\!\downarrow[\{x, y\}] = \sigma_2\tau_2[\{x, y\}]$ for $\sigma_2 = \{\}$ (the identity substitution) and $\tau_2 = \sigma\!\downarrow$ and therefore
$$t\sigma\!\downarrow = (t\sigma_2)\!\downarrow\tau_2.$$

The set of terms $\{t\sigma_i\!\downarrow\}$ (in this example, $1 \leq i \leq 2$) is then called a complete set of variants of t since any normal form $t\sigma\!\downarrow$ of an instantiation $t\sigma$ of the term t will be a syntactic instantiation of a term $t\sigma_i\!\downarrow$ (for some $1 \leq i \leq 2$).

Rewrite systems for which finite complete sets of variants can be found for any term t are said to have the *finite variant property* (from hereon FVP). The FVP is useful in symbolic analysis of security protocols [53] and in solving unification and disunification problems [53, 84], since they allow to get rid of the rewrite system and solve the resulting problems syntactically.

Complete sets of variants were introduced in [53] by Comon and Delaune. In their work, the normal form of a term is defined modulo some equational theory (typically AC). Their work includes showing that several rewrite systems have the finite variant property: the standard Dolev Yao theory with explicit descriptors (modulo the empty equational theory), a presentation of abelian groups (modulo AC) and a presentation of the Diffie-Hellman cryptographic primitives (modulo AC). Another line of work [83] proposes algorithms for proving and for disproving that a certain rewrite theory has or does not have the finite variant property.

In this thesis, we restrict ourselves to working modulo the empty equational theory and therefore we consider syntactic normal forms. Furthermore, since the complete set of variants $\{t\sigma_i\!\downarrow\}$ of t is fully determined by t and the set of substitutions $\{\sigma_i\}$, we will from now speak of substitutions σ_i as being the variants of t and we will say that the set $\{\sigma_i\}$ is a complete set of variants of t. In Section 3.2 we recall the definition of FVP in our setting.

Our work in this chapter is motivated by the procedure for verifying security protocols that we describe in Chapter 5. We define a notion of *strongly complete set of variants* (Section 3.3), which is a stronger version of the *complete set of variants*. Rewrite systems having finite strongly complete sets of variants for any term are said to have the strong finite variant property (from hereon SFVP).

We discuss the differences between the SFVP and the FVP in Section 3.4. In particular, in the presence of free symbols of arity ≥ 2, the two notions coincide. In the same section, we also compare strongly complete sets of variants with complete sets of variants and we show a link between the two.

In Section 3.5, we investigate the SFVP for convergent optimally reducing rewrite systems, which are useful for modeling many cryptographic primitives. We show that convergent optimally reducing rewrite systems have the strong finite variant property and we give an algorithm to compute a finite strongly complete set of variants for a given term and a given optimally reducing rewrite system. In chapter 5, strongly complete sets of variants allow us to get rid of the underlying rewrite system in our procedure for verifying security protocols.

In Section 3.6, we show that having the strong finite variant property is sufficient for equational unification problems to have finite complete sets of unifiers. The proof is constructive: a finite complete set of unifiers of an equation modulo the rewrite system is built from the strongly complete sets of variants of the two terms in the equation.

The two applications of our work, namely the procedure for verifying security protocols in Chapter 5 and the computation of finite complete sets of unifiers in Section 3.6 show that the notion of strongly complete set of variants is more natural and more useful than the notion of a complete set of variants. This chapter is based on work [47] that has been presented at the UNIF 2011 workshop. The algorithms presented here have been implemented in the tool SubVariant [48].

3.2 The Finite Variant Property (FVP)

Formally, the finite variant problem is to find finite complete sets of variants for a term:

Definition 3.1. A *complete set of variants* of a term t with respect to a convergent term rewriting system R is a set of substitutions $\{\sigma_1, \ldots, \sigma_n\}$ such that for any substitution π, we have that $(t\pi)\downarrow = ((t\sigma_i)\downarrow)\tau$ for some index $1 \leq i \leq n$ and some substitution τ.

In the above definition, the difficulty of finding a complete set of variants lies in the fact that the term $(t\sigma_i)\downarrow\tau$ is not normalized anymore after the application of the substitution τ to $(t\sigma_i)\downarrow$. Therefore the set consisting of the identity substitution is in general not a complete set of variants.

> *Example* 3.1. Let $t = \mathrm{dec}(x, y)$ and $\mathsf{R} = \{\mathrm{dec}(\mathrm{enc}(x, y), y) \to x\}$. We have that the set consisting of the substitution $\sigma_1 = \{\}$ (the identity substitution) and of the substitution $\sigma_2 = \{x \mapsto \mathrm{enc}(z, y)\}$ is a complete set of variants of t. We prove that $\{\sigma_1, \sigma_2\}$ is a complete set of variants of t by choosing a substitution τ for any substitution π as follows:
>
> 1. if the decryption does not succeed (i.e. $x\pi\downarrow \neq \mathrm{enc}(t', y\pi\downarrow)$), then we let $\tau = \pi\downarrow$ and we have that $(\mathrm{dec}(x, y)\pi)\downarrow = \mathrm{dec}(x, y)(\pi\downarrow) = (\mathrm{dec}(x, y)\sigma_1)\downarrow\tau$.
> 2. if $x\pi\downarrow = \mathrm{enc}(t', y\pi\downarrow)$ (i.e. the decryption succeeds), then we let $\tau = \{z \mapsto t'\}$ and we have that $(\mathrm{dec}(x, y)\pi)\downarrow = t' = z\{z \mapsto t'\} = (\mathrm{dec}(x, y)\sigma_2)\downarrow\{z \mapsto t'\}$.

A term rewriting system R has the *FVP* if any term $t \in T(\mathcal{F}, \mathcal{A})$ has a finite complete set of variants. The following example illustrates that not all rewrite systems have the FVP.

> *Example* 3.2. We consider the term rewriting system $\mathsf{R} = \{f(g(x)) \to g(f(x))\}$ and the term $t = f(x)$. By analyzing the sequence of substitutions $\pi_i = \{x \mapsto g^i(y)\}$ (for all $i \in \mathbb{N}$)

and the normal forms $(t\pi_i)\downarrow = g^i(f(y))$ (for all $i \in \mathbb{N}$), it can be proven that any complete set of variants of t will contain all of the substitutions:

$$\sigma_i = \{x \mapsto g^i(y)\}$$

for all $i \in \mathbb{N}$ and up to renaming of the variable y. Therefore this term rewriting system does not have the FVP.

3.3 The Strong Finite Variant Property

The notion of FVP was introduced in [53]. In the procedure for verifying security protocols that we describe in Chapter 5, we require a slightly stronger notion, which motivates the following definition:

Definition 3.2. Let t be a term with the set of variables $\mathcal{V}ar(t) = X$. A set of substitutions $\{\sigma_1, \ldots, \sigma_n\}$ is called a strongly complete set of variants of t if the substitutions σ_i $(1 \leq i \leq n)$ are of domain $\mathcal{D}om(\sigma_i) \subseteq X$ for all $1 \leq i \leq n$ and for any substitution π, we have that $(\pi[X])\downarrow = ((\sigma_i\downarrow)\tau)[X]$[1] for some substitution τ and some index $1 \leq i \leq n$ such that $(t\pi)\downarrow = ((t\sigma_i)\downarrow)\tau$.

The difference between a complete set of variants and a strongly complete set of variants is that we require not only that $(t\pi)\downarrow$ be equal to $(t\sigma_i)\downarrow\tau$, but also that $\pi\downarrow$ and $(\sigma_i\downarrow)\tau$ coincide on the variables of t.

Note that in the above definition the condition $\pi[X]\downarrow = (\sigma_i\downarrow\tau)[X]$ is not sufficient in the sense that it does not in general imply $(t\pi)\downarrow = (t\sigma_i)\downarrow\tau$: take $\mathsf{R} = \{\mathrm{dec}(\mathrm{enc}(x, y), y) \to x\}$, $t = \mathrm{dec}(x, y)$, $\pi = \{x \mapsto \mathrm{enc}(z, y)\}$, $\sigma_i = \{\}$ (the identity substitution). We have that $\tau = \pi$ is such that $\pi[X]\downarrow = (\sigma_i\downarrow\tau)[X] = \tau[X]$ but $(t\pi)\downarrow = z \neq (t\sigma_i)\downarrow\tau = \mathrm{dec}(\mathrm{enc}(z, y), y)$.

As with complete sets of variants, a finite strongly complete set of variants does not exist in general. As expected, rewrite systems R for which any term t has a finite strongly complete set of variants are said to have the *strong finite variant property* (from hereon SFVP).

The notion of strongly complete set of variants is stronger than the notion of complete set of variants. It is obvious that a strongly complete set of variants is always a complete set of variants. The following example illustrates that the reverse is not true and reveals the subtlety between a *complete set of variants* and a *strongly complete set of variants*.

Example 3.3. We consider the (subterm convergent) term rewriting system

$$\mathsf{R} = \{h(f(x), y) \to y, h(g(x), y) \to y\}$$

and the term

$$t = h(x, y).$$

It is easy to see that the following set S is a complete finite set of variants of t:

$$S = \{\sigma_1 = \{\}, \sigma_2 = \{x \mapsto f(z)\}\}.$$

Note that S does not contain the substitution $\sigma_3 = \{x \mapsto g(z)\}$. Even if S is a finite complete set of variants of t it is not a *strongly* complete set of variants of t: if we consider the substitution $\pi = \{x \to g(a)\}$ for some constant a, we have that:

1. $\pi\downarrow[X] = (\sigma_1\downarrow)\tau_1[X]$ (with $\tau_1 = \{x \mapsto g(a)\}$), but $(t\pi)\downarrow \neq ((t\sigma_1)\downarrow)\tau_1$.
2. $\pi\downarrow[X] \neq (\sigma_2\downarrow)\tau_2[X]$ for any substitution τ_2.

where $X = \mathcal{V}ar(t)$ is the set of variables of t. However, the set $S \cup \{\sigma_3\}$ is a strongly complete set of variants of t.

[1] Recall that the notation $\pi[X]$ denotes the restriction of the substitution π to the variables in X.

3.4 Relating the SFVP and the FVP

3.4.1 Strict Containment

It is easy to see that a term rewriting system having the SFVP also has the FVP. The reverse is not true: term rewriting systems having the FVP need not have the SFVP. Let us consider the signature $\mathcal{F} = \{f/1, g/1\}$ and the following convergent term rewriting system

$$\mathsf{R} = \{f(g(x)) \to f(x)\}.$$

It is easy to see that any term t has a normal form which is either the term $t{\downarrow} = g^n(f^m(x))$ or the term $t{\downarrow} = g^n(f^m(a))$ for some variable $x \in \mathcal{X}$, non-variable atom $a \in \mathcal{A} \setminus \mathcal{X}$ and integers $n, m \in \mathbb{N}$. We will show that the identity substitution $\{\}$ forms by itself a complete set of variants of any term t built over the signature $\mathcal{F}$.

Proof. Let σ be an arbitrary substitution and t be an arbitrary term.

1. If $t{\downarrow} = g^n(f^m(a))$ for some non-variable atom $a \in \mathcal{A} \setminus X$, then $(t\sigma){\downarrow} = ((t{\downarrow})\sigma){\downarrow} = t{\downarrow} = (t\{\}){\downarrow}\{\}$.
2. If $t{\downarrow} = g^n(f^m(x))$ for some variable $x \in \mathcal{X}$, let the term $s = \sigma(x)$ be the value of σ at x. We have that $s{\downarrow} = g^p(f^q(y))$ or $s{\downarrow} = g^p(f^q(b))$ for some variable $y \in \mathcal{X}$, non-variable atom $b \in \mathcal{A} \setminus \mathcal{X}$ and some integers $p, q \in \mathbb{N}$.
   (a) if $s{\downarrow} = g^p(f^q(y))$ then
      i. if $m = 0$, we have $t\sigma{\downarrow} = g^{n+p}(f^q(y)) = (t\{\}){\downarrow}\sigma$ and
      ii. otherwise, if $m > 0$, we have $t\sigma{\downarrow} = g^n(f^{m+q}(y)) = (t\{\}){\downarrow}\{x \mapsto f^q(y)\}$.
   (b) if $s{\downarrow} = g^p(f^q(b))$ then
      i. if $m = 0$, we have $t\sigma{\downarrow} = g^{n+p}(f^q(b)) = (t\{\}){\downarrow}\sigma$ and
      ii. otherwise, if $m > 0$, we have $(t\sigma){\downarrow} = g^n(f^{m+q}(b)) = (t\{\}){\downarrow}\{x \mapsto f^q(b)\}$.

We have shown that the identity substitution forms by itself a complete set of variants of any term t over $\mathcal{F}$. □

However, R does not have the SFVP. This is illustrated by the following example.

Example 3.4. Let $t = f(x)$ be a term where $x \in \mathcal{X}$ is a variable. By analyzing the instantiations $t\pi_i$ where the substitutions π_i are defined as $\pi_i = \{x \mapsto g^i(y)\}$ for all $i \in \mathbb{N}$, it can be shown that any strongly complete set of variants must contain substitutions σ_i for all $i \in \mathbb{N}$ such that $\sigma_i[\{x\}] = \{x \mapsto g^i(z)\}$ for all $i \in \mathbb{N}$ (up to renaming of z). Therefore any strongly complete set of variants of t is infinite.

3.4.2 In the Presence of Free Symbols

We have shown that in general the SFVP is a strictly stronger property than the FVP.

However, if the signature contains at least a free symbol of arity strictly greater than 1, we show that the two notions coincide. Let $f \in \mathcal{F}$ be the free symbol of arity $\mathsf{ar}(f) = k \geq 2$. From hereon, we will denote by $\mathsf{tuple}(t_1, \ldots, t_n)$ the term

$$\mathsf{tuple}(t_1, \ldots, t_n) = f(t_1, t, \ldots, t, f(t_2, t, \ldots, t, f(\cdots f(t_n, t, \ldots, t))))$$

where t is an arbitrary ground term in normal form (for example a constant in normal form or a name).

Example 3.5. If $k = 3$, then $\mathsf{tuple}(a, b, c) = f(a, t, f(b, t, f(c, t, t)))$.

Because f is a free symbol, we have by Theorem 2.1 that

$$\mathsf{tuple}(t_1, \ldots, t_n)\downarrow = \mathsf{tuple}(t_1\downarrow, \ldots, t_n\downarrow). \tag{3.1}$$

The following theorem shows that in the presence of at least a free symbol f of arity $k \geq 2$ the two notions (FVP and SFVP) coincide. Note that the presence of a free symbol does not make the notions of complete sets of variants and of strongly complete sets of variants coincide; however a strongly complete set of variants of some term can be constructed by computing a complete set of variants of another, more complex, term.

Theorem 3.1. *Let t be a term and let $X = \{x_1, \ldots, x_n\} = \mathcal{V}ar(t)$ be the set of its variables. Let S be a complete set of variants of $\mathsf{tuple}(t, x_1, \ldots, x_n)$. Then*

$$S' = \{\sigma[X] \mid \sigma \in S\}$$

is a strongly complete set of variants of t.

Proof. Let π be an arbitrary substitution. We need to show that there exist a substitution $\sigma' \in S'$ and a substitution τ' such that $\pi[X]\downarrow = ((\sigma'\downarrow)\tau')[X]$ and $(t\pi)\downarrow = (t\sigma')\downarrow\tau'$. Because S is a complete set of variants of $\mathsf{tuple}(t, x_1, \ldots, x_n)$, we have that there exist a substitution $\sigma \in S$ and a substitution τ such that

$$(\mathsf{tuple}(t, x_1, \ldots, x_n)\pi)\downarrow = ((\mathsf{tuple}(t, x_1, \ldots, x_n)\sigma)\downarrow)\tau. \tag{3.2}$$

By applying Equation 3.1 to each side of the equality we obtain

$$(\mathsf{tuple}(t, x_1, \ldots, x_n)\pi)\downarrow = \mathsf{tuple}((t\pi)\downarrow, (x_1\pi)\downarrow, \ldots, (x_n\pi)\downarrow)$$

$$((\mathsf{tuple}(t, x_1, \ldots, x_n)\sigma)\downarrow)\tau = \mathsf{tuple}(((t\sigma)\downarrow)\tau, ((x_1\sigma)\downarrow)\tau, \ldots, ((x_n\sigma)\downarrow)\tau)$$

and, as the two terms are equal by Equation (3.2), it follows that every every pair of terms at the same position in the two tuples are equal: $(t\pi)\downarrow = ((t\sigma)\downarrow)\tau$ and $x_i\pi\downarrow = x_i\sigma\downarrow\tau$ for all $1 \leq i \leq n$. But as $\{x_1, \ldots, x_n\} = X$, it follows that $\pi[X]\downarrow = ((\sigma\downarrow)\tau)[X]$. As $X = \mathcal{V}ar(t) = \{x_1, \ldots, x_n\}$, it immediately follows that $(t\pi)\downarrow = ((t(\sigma[X]))\downarrow)\tau$ and $\pi[X]\downarrow = (((\sigma[X])\downarrow)\tau)[X]$.

As $\sigma \in S$, we have that $\sigma[X] \in S'$. We have shown therefore that there exist a substitution $\sigma' = \sigma[X] \in S'$ and a substitution $\tau' = \tau$ such that $\pi[X] = ((\sigma'\downarrow)\tau')[X]$ and $(t\pi)\downarrow = (t\sigma')\downarrow\tau'$, thereby proving that S' is a strongly complete set of variants of t. □

3.5 Computing Strongly Complete Sets of Variants for Convergent Optimally Reducing Rewrite Systems

In this section, we present an algorithm for computing strongly complete sets of variants for terms under a convergent optimally reducing rewrite system.

The algorithm for computing a strongly complete finite set of variants is based on a refinement of *narrowing* which is sound and complete for convergent optimally reducing rewrite systems. Each narrowing step (denoted hereafter $\hookrightarrow$) works on a configuration $(t, \mathcal{P}, X)$ consisting of a term t, a set of positions $\mathcal{P}$ of t at which narrowing is allowed and a set of *forbidden* variables X in the sense that fresh variables should be chosen outside of X. The main difference between our refinement of narrowing and standard narrowing is that positions at which a narrowing step is performed are restricted by the set $\mathcal{P}$.

In order to describe the refinement of narrowing that we use in our algorithm we first need a new notation: if p is a position, by $\mathsf{down}(p)$ we denote the set of positions which is the downward closure of p, i.e. all positions that are descendants of p, including p itself:

$$\mathsf{down}(p) = \{pq \mid \text{ for all positions } q\}.$$

$$\frac{\begin{array}{ll} q \in \mathcal{P} & \\ l \to r \bar{\in} \mathsf{R} & \mathcal{V}ar(\{l,r\}) \cap X = \emptyset \\ \theta = \mathsf{mgu}(l, t|_q) & \end{array}}{(t, \mathcal{P}, X) \overset{\theta}{\hookrightarrow} (t\theta[r\theta]_q, \mathcal{P} \setminus \mathsf{down}(q), X \cup \mathcal{V}ar(\mathcal{R}ange(\theta)))}$$

Figure 3.1: Narrowing step (complete for convergent optimally reducing rewrite systems)

To compute a complete finite set of variants of some term t and a convergent optimally reducing rewrite system R, we will begin with the initial configuration

$$\mathcal{C}_0 = (t, \mathcal{P}os_{init}(t), \mathcal{V}ar(t))$$

where $\mathcal{P}os_{init}(t) = \mathcal{P}os(t) \setminus \{p \mid t|_p \in \mathcal{X}\}$ is the set of all non-variable positions of t and non-deterministically apply narrowing steps.

Each narrowing step non-deterministically chooses a rewrite rule $l \to r$ and a position p from $\mathcal{P}$ where narrowing is performed. The initial choice of $\mathcal{P} = \mathcal{P}os_{init}(t)$ in the initial configuration is a way to enforce the *basic restriction*, that is, narrowing is only performed strictly inside t (and not inside the variables of t). Furthermore, if we have performed narrowing at a position p and because of the specificity of convergent optimally reducing rewrite systems, there is no need to consider this position or any of its descendants anymore and therefore they are removed from $\mathcal{P}$.

We write $\overset{\theta_1 \dots \theta_n}{\hookrightarrow}$ for the relation $\overset{\theta_1}{\hookrightarrow}\overset{\theta_2}{\hookrightarrow} \dots \overset{\theta_n}{\hookrightarrow}$. In particular, if $n = 0$, $\hookrightarrow$ denotes the identity relation. We show that our narrowing procedure allows to compute a finite strongly complete set of variants for any term t:

Theorem 3.2 (Correctness)**.** *If* R *is a convergent optimally reducing rewrite system, the set*

$$\Sigma = \{\sigma = \theta_1 \dots \theta_n \mid (t, \mathcal{P}os_{init}(t), \mathcal{V}ar(t)) \overset{\theta_1, \dots, \theta_n}{\hookrightarrow} (t', \mathcal{P}', X')\}$$

is a finite strongly complete set of variants of t*.*

The proof of correctness of the above theorem relies on the following soundness and completeness lemmas:

Lemma 3.1 (Partial Soundness)**.** *Let* t, t' *be terms,* $\mathcal{P}, \mathcal{P}'$ *sets of positions,* X, X' *sets of variables and* θ *a substitution. If*

$$(t, \mathcal{P}, X) \overset{\theta}{\hookrightarrow} (t', \mathcal{P}', X'),$$

we have that $t\theta \to_{\mathsf{R}} t'$ *and* $X' = X \cup \mathcal{V}ar(\mathcal{R}ange(\theta))$*.*

Proof. By the definition of $\overset{\theta}{\hookrightarrow}$, we have that if $(t, \mathcal{P}, X) \overset{\theta}{\hookrightarrow} (t', \mathcal{P}', X')$, there exist a rewrite rule $l \to r \bar{\in} \mathsf{R}$, a position q and a substitution θ such that $t|_q\theta = l\theta$. We have that $t\theta = t\theta[l\theta]_q \to_{\mathsf{R}} t\theta[r\theta]_q = t'$. The fact that $X' = X \cup \mathcal{V}ar(\mathcal{R}ange(\theta))$ is immediate from the definition of $\overset{\theta}{\hookrightarrow}$. □

By iterating Lemma 3.1, we obtain:

Lemma 3.2 (Soundness)**.** *Let* t, t' *be terms,* $\mathcal{P}, \mathcal{P}'$ *sets of positions,* X, X' *sets of variables and* $\theta_1, \dots, \theta_n$ *substitutions. If*

$$(t, \mathcal{P}, X) \overset{\theta_1 \dots \theta_n}{\hookrightarrow} (t', \mathcal{P}', X'),$$

we have that $t\theta_1 \dots \theta_n \to^*_{\mathsf{R}} t'$ *and* $X' = X \cup \mathcal{V}ar(\mathcal{R}ange(\theta_1)) \cup \dots \cup \mathcal{V}ar(\mathcal{R}ange(\theta_n))$*.*

Proof. We proceed by induction on n. If $n = 0$, we have that $\overset{\theta_1 \dots \theta_n}{\hookrightarrow} = \hookrightarrow$ is the identity relation. Therefore $X' = X$ and $t\theta_1 \dots \theta_n = t = t'$, which implies $t\theta_1 \dots \theta_n \to^*_{\mathsf{R}} t'$.

If $n > 0$, we have that

$$(t, \mathcal{P}, X) \overset{\theta_1 \ldots \theta_{n-1}}{\hookrightarrow} (t'', \mathcal{P}'', X'') \overset{\theta_n}{\hookrightarrow} (t', \mathcal{P}', X')$$

for some term t'', some set of positions $\mathcal{P}''$ and some set of variables X''.

By the induction hypothesis, we have that

$$t\theta_1 \ldots \theta_{n-1} \rightarrow_{\mathsf{R}}^* t'' \tag{3.3}$$

and that

$$X'' = X \cup \mathcal{V}ar(\mathcal{R}ange(\theta_1)) \cup \ldots \cup \mathcal{V}ar(\mathcal{R}ange(\theta_{n-1})). \tag{3.4}$$

As $(t'', \mathcal{P}'', X'') \overset{\theta_n}{\hookrightarrow} (t', \mathcal{P}', X')$, we have by Lemma 3.1 that

$$X' = X'' \cup \mathcal{V}ar(\mathcal{R}ange(\theta_n)) \tag{3.5}$$

and that

$$t''\theta_n \rightarrow_{\mathsf{R}} t'. \tag{3.6}$$

By combining Equation (3.3) with Equation (3.6) we obtain that $t\theta_1 \ldots \theta_n \rightarrow_{\mathsf{R}}^* t'$ and by combining Equation (3.4) with Equation (3.5), we obtain that $X' = X \cup \mathcal{V}ar(\mathcal{R}ange(\theta_1)) \cup \ldots \cup \mathcal{V}ar(\mathcal{R}ange(\theta_n))$, which is what we had to show.

□

Now we describe what completeness means for our narrowing rule:

Lemma 3.3 (Partial Completeness)**.** *Let t be a term, ω a substitution such that $t\omega$ is not in normal form, X a set of variables such that $\mathcal{V}ar(t) \subseteq X$ and $\mathcal{P} \subseteq \mathcal{P}os(t)$ a set of positions in t such that $t\omega|_p$ is in normal form for any position $p \subseteq \mathcal{P}os(t\omega) \setminus \mathcal{P}$.*

If R is a convergent optimally reducing rewrite system, there exist substitutions θ and ω', a term t', a set of positions $\mathcal{P}' \subseteq \mathcal{P}os(t')$ and a set of variables X' such that

$$(t, \mathcal{P}, X) \overset{\theta}{\hookrightarrow} (t', \mathcal{P}', X') \text{ and}$$

$$\omega[X] = \theta\omega'[X].$$

Furthermore, $\mathcal{V}ar(t') \subseteq X'$ and $t'\omega'|_p$ is in normal form for every $p \in \mathcal{P}os(t'\omega') \setminus \mathcal{P}'$.

Proof. As $t\omega$ is not in normal form, there exists a position q, a rewrite rule $l' \rightarrow r' \in \mathsf{R}$ and a substitution σ' of domain $\mathcal{D}om(\sigma') \subseteq \mathcal{V}ar(l')$ such that $t\omega|_q = l'\sigma'$. Let q be maximal (w.r.t. to its length) with this property.

Let τ be a variable renaming such that $\mathcal{V}ar(\{l'\tau, r'\tau\}) \cap X = \emptyset$, let $l = l'\tau$, $r = r'\tau$ and $\sigma = \tau^{-1}\sigma'$. We have that $l \rightarrow r \bar{\in} \mathsf{R}$ with $\mathcal{V}ar(\{l, r\}) \cap X = \emptyset$, $t\omega|_q = l'\sigma' = l'\tau\tau^{-1}\sigma' = l\sigma$.

Because $t\omega|_q$ is an instance of the left-hand side of a rewrite rule, we have that $t\omega|_q$ is not in normal form. As $t\omega|_p$ is in normal form for any position $p \in \mathcal{P}os(t\omega) \setminus \mathcal{P}$ by hypothesis, it must be that $q \in \mathcal{P}$.

We have shown that $q \in \mathcal{P}$; as $\mathcal{P} \subseteq \mathcal{P}os(t)$ by hypothesis, we have that $q \in \mathcal{P}os(t)$. Therefore $t\omega|_q = t|_q\omega$.

As $\mathcal{V}ar(t) \subseteq X$, we have that $t|_q(\omega[X]) = t|_q\omega = t\omega|_q = l\sigma$. As the domains of the substitutions $\omega[X]$ ($\mathcal{D}om(\omega[X]) \subseteq X$) and σ ($\mathcal{D}om(\sigma) \subseteq \mathcal{V}ar(l)$ and $\mathcal{V}ar(l) \cap X = \emptyset$) are disjoint, we have that $\omega[X] \uplus \sigma$ is a unifier of $t|_q$ and l.

As $t|_q$ and l are unifiable, they have a most general unifier. Let $\theta = \mathsf{mgu}(t|_q, l)$ be a most general unifier of $t|_q$ and l. As $\omega[X] \uplus \sigma$ is a unifier of $t|_q$ and l, it follows that it is an instance of their most general unifier: there exists a substitution ω' such that $\omega[X] \uplus \sigma = \theta\omega'$. It easily follows that

$$\omega[X] = \theta\omega'[X] \text{ and} \tag{3.7}$$

$$\sigma[\mathcal{V}ar(l)] = \theta\omega'[\mathcal{V}ar(l)]. \tag{3.8}$$

Let $t' = t\theta[r\theta]_q$, $\mathcal{P}' = \mathcal{P} \setminus \mathsf{down}(q)$ and $X' = X \cup \mathcal{V}ar(\mathcal{R}ange(\theta))$. We have that $\mathcal{P}' \subseteq \mathcal{P}os(t')$. By the definition of $\overset{\theta}{\hookrightarrow}$, we have that

$$(t, \mathcal{P}, X) \overset{\theta}{\hookrightarrow} (t', \mathcal{P}', X'). \tag{3.9}$$

As $\mathcal{V}ar(r\theta) \subseteq \mathcal{V}ar(l\theta)$ by the definition of a rewrite rule, we have that $\mathcal{V}ar(t') = \mathcal{V}ar(t\theta[r\theta]_q) \subseteq \mathcal{V}ar(t\theta[l\theta]_q) = \mathcal{V}ar(t\theta) \subseteq \mathcal{V}ar(t) \cup \mathcal{V}ar(\mathcal{R}ange(\theta))$ and therefore $\mathcal{V}ar(t') \subseteq X'$.

To finish the proof, it remains to be shown that $t'\omega'|_p$ is in normal form for every position $p \in \mathcal{P}os(t'\omega') \setminus \mathcal{P}'$. Let $p \in \mathcal{P}os(t'\omega') \setminus \mathcal{P}'$ be an arbitrary position. We will show that $t'\omega'|_p$ is in normal form by the following case distinction:

1. if q is a prefix of p (i.e. there exists a position q' such that $p = qq'$), we have that $t'\omega'|_p = (t\theta[r\theta]_q)\omega'|_p = r\theta\omega'|_{q'}$. By the definition of a rewrite rule, we have that $\mathcal{V}ar(r) \subseteq \mathcal{V}ar(l)$. Furthermore, by Equation (3.8), we have that $\sigma[\mathcal{V}ar(l)] = \theta\omega'[\mathcal{V}ar(l)]$ and therefore $r\theta\omega' = r\sigma$.

   As R is an optimally reducing rewrite system, we have that $l \to r \;\bar{\in}\; \mathsf{R}$ is an optimally reducing rewrite rule. Since q was chosen to be maximal such that $t\omega|_q$ is not in normal form, we have that all strict subterms of $l\sigma = t\omega|_q$ are in normal form. Therefore, as $l \to r$ is optimally reducing, it follows that $r\sigma$ is in normal form. Therefore any subterm of $r\sigma = r\theta\omega'$ is in normal form; in particular the subterm of $r\sigma$ at position q', $t'\omega'|_p = r\theta\omega'|_{q'}$, is in normal form.

2. p cannot be a prefix of q: we have that $t\omega|_p$ is not in normal form (since it has as subterm $l\sigma$). Therefore $p \in \mathcal{P}$. But $\mathcal{P}' = \mathcal{P} \setminus \mathsf{down}(q)$ and therefore $p \in \mathcal{P}'$. But p was chosen arbitrarily in $\mathcal{P}os(t'\omega') \setminus \mathcal{P}'$.

3. if p and q are incomparable (w.r.t. the prefix ordering), we have that $t'\omega'|_p = t\omega|_p$ and, as $p \notin \mathcal{P}'$ and $\mathcal{P}' = \mathcal{P} \setminus \mathsf{down}(q)$, it must be that $p \notin \mathcal{P}$. Therefore $t'\omega'|_p = t\omega|_p$ is in normal form by hypothesis.

We have shown that our choice of $\theta, t', X', \mathcal{P}'$ satisfies each of the properties we wanted and therefore we conclude.

□

By applying Lemma 3.3 iteratively, we obtain the follow completeness lemma:

Lemma 3.4 (Completeness)**.** *Let t be a term, ω a substitution, X a set of variables such that $\mathcal{V}ar(t) \subseteq X$ and $\mathcal{P} \subseteq \mathcal{P}os(t)$ a set of positions such that for all $p \in \mathcal{P}os(t\omega) \setminus \mathcal{P}$, we have that $t\omega|_p$ is in normal form.*

If R is a convergent optimally reducing rewrite system, there exist an integer $n \geq 0$, substitutions $\theta_1, \ldots, \theta_n$, a substitution ω', a term t', a set of positions $\mathcal{P}'$ and a set of variables X' such that

$$(t, \mathcal{P}, X) \overset{\theta_1 \ldots \theta_n}{\hookrightarrow} (t', \mathcal{P}', X'),$$

$\omega[X] = \theta_1 \ldots \theta_n\omega'[X]$ and $t'\omega'$ is in normal form. Furthermore, $\mathcal{V}ar(t') \subseteq X'$.

Proof. We proceed by induction on $t\omega$ equipped with the well-founded order $\to_{\mathsf{R}}$ given by the terminating term rewriting relation.

Base case. If $t\omega$ is already in normal form, we choose $n = 0$, $\omega' = \omega$, $t' = t$, $X' = X$ and $\mathcal{P}' = \mathcal{P}$. It trivially follows that $(t, \mathcal{P}, X) \overset{\theta_1 \ldots \theta_n}{\hookrightarrow} (t', \mathcal{P}', X')$ (as $\hookrightarrow = \overset{\theta_1 \ldots \theta_n}{\hookrightarrow}$ is the identity relation), that $\omega[X] = \omega'[X]$, that $\mathcal{V}ar(t') \subseteq X'$, that $t'\omega' = t\omega$ is in normal form.

Inductive case. If $t\omega$ is not in normal form, we have by Lemma 3.3 that there exists t'', θ, ω'', $\mathcal{P}''$ and X'' such that

$$(t, \mathcal{P}, X) \overset{\theta}{\hookrightarrow} (t'', \mathcal{P}'', X''), \tag{3.10}$$

$\omega[X] = \theta\omega''[X]$, $t''\omega''|_p$ is in normal form for all $p \in \mathcal{P}os(t''\omega'') \setminus \mathcal{P}''$ and $\mathcal{V}ar(t'') \subseteq X''$. By Lemma 3.1, we have that $X'' = X \cup \mathcal{V}ar(\mathcal{R}ange(\theta))$. As $\mathcal{V}ar(t) \subseteq X$ and $\omega[X] = \theta\omega''[X]$, we have by Lemma 2.1 that $t\omega = t\theta\omega''$. By Lemma 3.1, we have that $t\theta \to t''$. Therefore $t\theta\omega'' \to t''\omega''$, which implies $t\omega \to t''\omega''$.

We can therefore apply the induction hypothesis on t'' and ω'' to obtain that there exists $m \geq 0$, substitutions $\omega'_1, \ldots, \omega'_m$, a substitution ω', a term t', a set of positions $\mathcal{P}'$ and a set of variables X' such that

$$(t'', \mathcal{P}'', X'') \overset{\theta'_1 \ldots \theta'_m}{\hookrightarrow} (t', \mathcal{P}', X'), \tag{3.11}$$

$\omega''[X''] = \theta'_1 \ldots \theta'_m\omega'[X'']$, $\mathcal{V}ar(t') \subseteq X'$ and $t'\omega'$ is in normal form.

Let $n = m + 1$, $\theta_1 = \theta$ and $\theta_i = \theta'_{i-1}$ for $2 \leq i \leq n$. By Equation (3.10) and Equation (3.11), we have that $(t, \mathcal{P}, X) \overset{\theta_1 \ldots \theta_n}{\hookrightarrow} (t', \mathcal{P}', X')$.

We have that $\omega[X] = \theta\omega''[X]$, that $\omega''[X''] = \theta'_1 \ldots \theta'_m\omega'[X'']$ and that $X'' = X \cup \mathcal{V}ar(\mathcal{R}ange(\theta))$. By Lemma 2.2, we obtain that $\omega[X] = \theta\theta'_1 \ldots \theta'_m\omega'[X]$. But $\theta_1 = \theta$ and $\theta_i = \theta'_{i-1}$ for $2 \leq i \leq n$ by choice of $\theta_1, \ldots, \theta_n$. Therefore $\omega[X] = \theta_1 \ldots \theta_n\omega'[X]$.

In both cases, we have shown that there exist $n \geq 0, \theta_1, \ldots, \theta_n, t', \mathcal{P}', X'$ such that

$$(t, \mathcal{P}, X) \overset{\theta_1 \ldots \theta_n}{\hookrightarrow} (t', \mathcal{P}', X'),$$

$\omega[X] = \theta_1 \ldots \theta_n\omega'[X]$, $\mathcal{V}ar(t') \subseteq X'$ and $t'\omega'$ is in normal form, which is what we had to prove. □

We are now ready to show the correctness of the algorithm:

Theorem 3.2 (Correctness)**.** *If* R *is a convergent optimally reducing rewrite system, the set*

$$\Sigma = \{\sigma = \theta_1 \ldots \theta_n \mid (t, \mathcal{P}os_{init}(t), \mathcal{V}ar(t)) \overset{\theta_1, \ldots, \theta_n}{\hookrightarrow} (t', \mathcal{P}', X')\}$$

is a finite strongly complete set of variants of t*.*

Proof of Theorem 3.2. Let ω_0 be an arbitrary substitution and let $\omega = \omega_0{\downarrow}$. We have immediately that $t\omega_0{\downarrow} = t\omega{\downarrow}$. We show that there exist a substitution $\sigma \in \Sigma$ and a substitution τ such that $t\omega_0{\downarrow} = t\omega{\downarrow} = (t\sigma){\downarrow}\tau$ and $\omega{\downarrow}[\mathcal{V}ar(t)] = \sigma\tau[\mathcal{V}ar(t)]$.

Let $\mathcal{P} = \mathcal{P}os_{init}(t)$ and $X = \mathcal{V}ar(t)$. We have that $t\omega|_p$ is in normal form for every position $p \in \mathcal{P}os(t\omega) \setminus \mathcal{P}os_{init}(t)$ by the definition of $\mathcal{P}os_{init}$ (every such term $t\omega|_p$ is a subterm of the substitution $\omega = \omega_0{\downarrow}$ and therefore it is obviously in normal form). We have furthermore that $\mathcal{V}ar(t) = X \subseteq X$. Therefore we can apply Lemma 3.4. We have that there exist $n \geq 0$, $\theta_1, \ldots, \theta_n$, $t', \omega', \mathcal{P}', X'$ such that

$$(t, \mathcal{P}, X) \overset{\theta_1 \ldots \theta_n}{\hookrightarrow} (t', \mathcal{P}', X'),$$

such that $\omega[X] = \theta_1 \ldots \theta_n\omega'[X]$ and such that $t'\omega'$ is in normal form.

Let $\sigma = \theta_1 \ldots \theta_n$ and let $\tau = \omega'$. By Lemma 3.2, we have that $t\theta_1 \ldots \theta_n \to^*_{\mathsf{R}} t'$ and therefore $t\theta_1 \ldots \theta_n\omega' \to^*_{\mathsf{R}} t'\omega'$. But $\theta_1 \ldots \theta_n\omega'[X] = \omega[X]$. As $X = \mathcal{V}ar(t)$, it follows by Lemma 2.1 that therefore $t\omega = t\theta_1 \ldots \theta_n\omega'$. Therefore $t\omega \to^*_{\mathsf{R}} t'\omega'$. As $t'\omega'$ is in normal form and R is convergent, it follows that $t\omega{\downarrow} = t'\omega'$. But $t\omega{\downarrow} = t\omega_0{\downarrow}$ and therefore $t\omega_0{\downarrow} = t'\omega'$. Since $t'\omega'$ is in normal form, it follows that t' must also be in normal form. As $t\theta_1 \ldots \theta_n \to^*_{\mathsf{R}} t'$ and $\sigma = \theta_1 \ldots \theta_n$, we have that $t\sigma{\downarrow} = t'$. Therefore $t\omega_0{\downarrow} = (t\sigma){\downarrow}\omega'$. But $\tau = \omega'$ and therefore $t\omega_0{\downarrow} = (t\sigma{\downarrow})\tau$.

By the choice of σ and τ, we have that $\omega[X] = \sigma\tau[X]$. As $\omega = \omega_0{\downarrow}$, it follows that $\omega_0{\downarrow}[X] = \sigma\tau[X]$. We have chosen ω_0 arbitrarily and we have shown that $\omega_0[X] = \sigma\tau[X]$ and $t\omega_0{\downarrow} = (t\sigma{\downarrow})\tau$ for some substitution $\sigma \in \Sigma$ and for some substitution τ. It follows that Σ is a strongly complete set of variants of t.

Each narrowing step $\overset{\theta_i}{\hookrightarrow}$ $(1 \leq i \leq n)$ is finitely branching (there is at most one narrowing choice for each rewrite rule and each position of the term). Furthermore each narrowing step strictly decreases the number of positions at which narrowing can be performed and therefore the number

n of steps $(\overset{\theta_1,\ldots,\theta_n}{\hookrightarrow})$ is finite. We have therefore that Σ is finite and we conclude that Σ is a finite strongly complete set of variants of t.

□

3.6 Equational Unification for Theories Having the SFVP

In this subsection we show that equational theories E implementable by term rewriting systems having the SFVP are finitary, i.e. any two terms s and t have finite complete set of E-unifiers. Our proof is constructive: we give an algorithm that computes a finite complete set of E-unifiers of two terms s and t from the finite strongly complete sets of variants of s and of t.

Indeed, the constructive algorithm follows immediately from the following theorem, which constructs a complete set of equational unifiers of two terms s and t, at most one equational unifier per pair of strong variants of s and of t.

Theorem 3.3. *Let* E *be an equational theory and let* R *be a convergent rewrite system that implements* E.

Let $s, t \in T(\mathcal{F}, \mathcal{A})$ *be two terms with common set of variables* $\mathcal{V}ar(s) \cap \mathcal{V}ar(t) = \{x_1, \ldots, x_n\}$ *and let* $\mathcal{V}ariants(s)$ *and* $\mathcal{V}ariants(t)$ *be finite, strongly complete sets of variants of the terms* s *and* t *with respect to the rewrite system* R. *Then the set*

$$
\begin{array}{lllll}
\Sigma = & \{ & \sigma_s{\downarrow}\pi_s\sigma[\mathcal{V}ar(s) \setminus \mathcal{V}ar(t)] & \uplus & \\
 & & \sigma_s{\downarrow}\pi_s\sigma[\mathcal{V}ar(s) \cap \mathcal{V}ar(t)] & \uplus & \\
 & & \sigma_t{\downarrow}\pi_t\sigma[\mathcal{V}ar(t) \setminus \mathcal{V}ar(s)] & | & \text{for all } \sigma_s \in \mathcal{V}ariants(s) \\
 & & & & \text{for all } \sigma_t \in \mathcal{V}ariants(t) \\
 & & & & \text{where } \pi_s \text{ is a variable renaming from } \mathcal{V}ar(s\sigma_s{\downarrow}) \\
 & & & & \qquad \text{to a fresh set of variables } X_s \\
 & & & & \text{where } \pi_t \text{ is a variable renaming from } \mathcal{V}ar(t\sigma_t) \\
 & & & & \qquad \text{to a fresh set of variables } X_t \\
 & & & & \quad \text{such that } X_s \cap X_t = \emptyset \\
 & & & & \text{where } \sigma = \mathsf{mgu}(\mathsf{tuple}(s\sigma_s{\downarrow}\pi_s, x_1\sigma_s{\downarrow}\pi_s, \ldots, x_n\sigma_s{\downarrow}\pi_s), \\
 & & & & \qquad \mathsf{tuple}(t\sigma_t{\downarrow}\pi_t, x_1\sigma_t{\downarrow}\pi_t, \ldots, x_n\sigma_t{\downarrow}\pi_t)) \\
 & & & & \text{where } \mathsf{tuple} \text{ is a fresh free function symbol} \\
 & & & & \qquad \text{of arity } \mathsf{ar}(\mathsf{tuple}) = n + 1 \qquad \}
\end{array}
$$

is a finite complete set of E*-unifiers of* s *and* t.

Before proving the theorem above, we illustrate the construction in the theorem by an example.

Example 3.6. Let $s = \mathrm{dec}(x, y)$ and $t = \mathrm{dec}(z, y)$. Notice that both terms are decryptions with the same variable y as a key. We are interested in computing a complete set Σ of unifiers of $s \overset{?}{=}_{\mathsf{R}} t$, where $\mathsf{R} = \{\mathrm{dec}(\mathrm{enc}(x, y), y) \to x\}$. We have that $\mathcal{V}ar(s) \cap \mathcal{V}ar(t) = \{y\}$. It is easy to see that

$$\mathcal{V}ariants(s) = \{\sigma_1, \sigma_2\}$$

is a strongly complete set of variants of the term s if $\sigma_1 = \{x \mapsto \mathrm{enc}(z, y)\}$ and $\sigma_2 = \{\}$ (the identity substitution). Similarly,

$$\mathcal{V}ariants(t) = \{\sigma_3, \sigma_4\}$$

is a strongly complete set of variants of the term t if $\sigma_3 = \{z \mapsto \mathrm{enc}(x, y)\}$ and $\sigma_4 = \{\}$ (the identity substitution).

For each pair of strong variants of s and t, we will add (at most) one equational unifier to the set of unifiers, as illustrated by the following table:

Variants $\sigma_s \in \{\sigma_1, \sigma_2\}$ $\sigma_t \in \{\sigma_3, \sigma_4\}$	$s\sigma_s\downarrow$	π_s	$t\sigma_t\downarrow$	π_t	$\sigma = \mathsf{mgu}($ $\mathsf{tuple}(s\sigma_s\downarrow\pi_s, y\pi_s),$ $\mathsf{tuple}(t\sigma_t\downarrow\pi_t, y\pi_t))$	resulting equational unifier
$\sigma_s = \{x \mapsto \mathrm{enc}(z, y)\}$ $\sigma_t = \{z \mapsto \mathrm{enc}(x, y)\}$	z	$\{z \mapsto z_s\}$	x	$\{x \mapsto x_t\}$	$\{z_s \mapsto x_t\}$	$\{x \mapsto \mathrm{enc}(x_t, y),$ $z \mapsto \mathrm{enc}(x_t, y)\}$
$\sigma_s = \{\}$ $\sigma_t = \{z \mapsto \mathrm{enc}(x, y)\}$	$\mathrm{dec}(x, y)$	$\{x \mapsto x_s,$ $y \mapsto y_s\}$	x	$\{x \mapsto x_t\}$	$\{x_t \mapsto \mathrm{dec}(x_s, y_s),$ $y \mapsto y_s\}$	$\{x \mapsto x_s,$ $y \mapsto y_s,$ $z \mapsto \mathrm{enc}($ $\mathrm{dec}(x_s, y_s),$ $y_s)\}$
$\sigma_s = \{z \mapsto \mathrm{enc}(x, y)\}$ $\sigma_t = \{\}$	x	$\{x \mapsto x_s\}$	$\mathrm{dec}(z, y)$	$\{z \mapsto z_t,$ $y \mapsto y_t\}$	$\{x_s \mapsto \mathrm{dec}(z_t, y_t),$ $y \mapsto y_t\}$	$\{x \mapsto \mathrm{enc}($ $\mathrm{dec}(z_t, y_t),$ $y_t)$ $y \mapsto y_t,$ $z \mapsto z_t\}$
$\sigma_s = \{\}$ $\sigma_t = \{\}$	$\mathrm{dec}(x, y)$	$\{x \mapsto x_s,$ $y \mapsto y_s\}$	$\mathrm{dec}(z, y)$	$\{z \mapsto z_t,$ $y \mapsto y_t\}$	$\{x_s \mapsto z_t,$ $y_s \mapsto y_t\}$	$\{x \mapsto z_t,$ $y \mapsto y_t,$ $z \mapsto z_t\}$

By the theorem above, we obtain that the set containing the four substitution on the rightmost column of the above table is a complete set of unifiers of $s = \mathrm{dec}(x, y)$ and $t = \mathrm{dec}(z, y)$.

We are now ready to proceed to the proof of the theorem.

Proof of Theorem 3.3. In order to prove the theorem, we have to show that the set Σ satisfies the following properties:

1. it is finite.
2. it is sound: any $\omega \in \Sigma$ is an E-unifier of s and t.
3. it is complete: any E-unifier τ' of s and t ($s\tau' =_{\mathsf{E}} t\tau'$) is an instance of some substitution $\omega \in \Sigma$.

We prove that Σ has each of the three properties in turn:

1. Σ **is finite**. Indeed, there is at most one substitution in Σ for every pair $(\sigma_s, \sigma_t) \in \mathcal{V}ariants(s) \times \mathcal{V}ariants(t)$ of variants of s and t.
2. Σ **is sound**, i.e. any substitution $\omega \in \Sigma$ is such that $s\omega =_{\mathsf{E}} t\omega$.

   Let $\omega \in \Sigma$ be an arbitrary substitution. As $\omega \in \Sigma$, it follows that there exist substitutions $\sigma_s \in \mathcal{V}ariants(s), \sigma_t \in \mathcal{V}ariants(t)$, a variable renaming π_s from $\mathcal{V}ar(s\sigma_s\downarrow)$ to some set of variables X_s, a variable renaming π_t from $\mathcal{V}ar(t\sigma_t\downarrow)$ to some set of variables X_t such that such that $X_s \cap X_t = \emptyset$ and a substitution

   $$\sigma = \mathsf{mgu}(\mathsf{tuple}(s\sigma_s\downarrow\pi_s, x_1\sigma_s\downarrow\pi_s, \ldots, x_n\sigma_s\downarrow\pi_s), \mathsf{tuple}(t\sigma_t\downarrow\pi_t, x_1\sigma_t\downarrow\pi_t, \ldots, x_n\sigma_t\downarrow\pi_t))$$

   such that $\omega = \sigma_s\downarrow\pi_s\sigma[\mathcal{V}ar(s) \setminus \mathcal{V}ar(t)] \uplus \sigma_s\downarrow\pi_s\sigma[\mathcal{V}ar(s) \cap \mathcal{V}ar(t)] \uplus \sigma_t\downarrow\pi_t\sigma[\mathcal{V}ar(t) \setminus \mathcal{V}ar(s)]$.

   We trivially have that $\omega[\mathcal{V}ar(s)] = \sigma_s\downarrow\pi_s\sigma[\mathcal{V}ar(s)]$.

   By the choice of σ, we have that $x_i\sigma_s\downarrow\pi_s\sigma = x_i\sigma_t\downarrow\pi_t\sigma$ for all variables $x_i \in \mathcal{V}ar(s) \cap \mathcal{V}ar(t) = \{x_1, \ldots, x_n\}$ and therefore

   $$\sigma_s\downarrow\pi_s\sigma[\mathcal{V}ar(s) \cap \mathcal{V}ar(t)] = \sigma_t\downarrow\pi_t\sigma[\mathcal{V}ar(s) \cap \mathcal{V}ar(t)]. \tag{3.12}$$

By using Equation (3.12), we immediately have that $\omega[\mathcal{V}ar(t)] = \sigma_t{\downarrow}\pi_t\sigma[\mathcal{V}ar(t)]$.

We have that $s\omega = s(\omega[\mathcal{V}ar(s)]) = s(\sigma_s{\downarrow}\pi_s\sigma) =_{\mathsf{R}} (s\sigma_s){\downarrow}\pi_s\sigma$. Similarly, $t\omega = t(\omega[\mathcal{V}ar(t)]) = t(\sigma_t{\downarrow}\pi_t\sigma) =_{\mathsf{R}} (t\sigma_t){\downarrow}\pi_t\sigma$. By choice of σ (it unifies $s\sigma_s{\downarrow}\pi_s$ and $t\sigma_t{\downarrow}\pi_t$) we have that $s\sigma_s{\downarrow}\pi_s\sigma = t\sigma_t{\downarrow}\pi_t\sigma$.

As $s\omega =_{\mathsf{R}} s\sigma_s{\downarrow}\pi_s\sigma = t\sigma_t{\downarrow}\pi_t\sigma =_{\mathsf{R}} t\omega$, we have that $s\omega =_{\mathsf{R}} t\omega$. But R implements E and therefore $s\omega =_{\mathsf{E}} t\omega$.

3. Σ **is complete**, i.e. for any substitution τ such that $s\tau =_{\mathsf{E}} t\tau$, there exist $\omega \in \Sigma$ and γ such that $\tau[\mathcal{V}ar(s,t)] =_{\mathsf{E}} \omega\gamma[\mathcal{V}ar(s,t)]$.

   Let τ be an arbitrary substitution such that $s\tau =_{\mathsf{E}} t\tau$. As R implements E, we have that $s\tau{\downarrow} = t\tau{\downarrow}$.

   By the definition of a strongly complete set of variants, we have that there exist $\sigma_s \in \mathcal{V}ariants(s)$ and τ_s such that

$$s\tau{\downarrow} = s\sigma_s{\downarrow}\tau_s \qquad \text{and} \qquad \tau{\downarrow}[\mathcal{V}ar(s)] = \sigma_s{\downarrow}\tau_s[\mathcal{V}ar(s)].$$

   Analogously, there exist $\sigma_t \in \mathcal{V}ariants(t)$ and τ_t such that

$$t\tau{\downarrow} = s\sigma_t{\downarrow}\tau_t \qquad \text{and} \qquad \tau{\downarrow}[\mathcal{V}ar(t)] = \sigma_t{\downarrow}\tau_t[\mathcal{V}ar(t)].$$

   Let π_s and π_t be the substitutions used in the definition of Σ. Let $\theta = \pi_s^{-1}\tau_s[X_s] \uplus \pi_t^{-1}\tau_t[X_t]$. We will show that θ unifies the term $\mathsf{tuple}(s\sigma_s{\downarrow}\pi_s, x_1\sigma_s{\downarrow}\pi_s, \ldots, x_n\sigma_s{\downarrow}\pi_s)$ with the term $\mathsf{tuple}(t\sigma_t{\downarrow}\pi_t, x_1\sigma_t{\downarrow}\pi_t, \ldots, x_n\sigma_t{\downarrow}\pi_t)$.

   Indeed, we have that $s\sigma_s{\downarrow}\pi_s\theta = s\sigma_s{\downarrow}\pi_s\pi_s^{-1}\tau_s = s\sigma_s{\downarrow}\tau_s = s\tau{\downarrow}$. Similarly, $t\sigma_t{\downarrow}\pi_t\theta = t\sigma_t{\downarrow}\pi_t\pi_t^{-1}\tau_t = t\sigma_t{\downarrow}\tau_t = t\tau{\downarrow}$. But $s\tau{\downarrow} = t\tau{\downarrow}$ and therefore $s\sigma_s{\downarrow}\pi_s\theta = t\sigma_t{\downarrow}\pi_t\theta$. This shows that θ unifies the elements on the first position in the two tuples.

   For the other elements, let $x_i \in \mathcal{V}ar(s) \cap \mathcal{V}ar(t) = \{x_1, \ldots, x_n\}$ be an arbitrary variable. We have that $x_i\sigma_s{\downarrow}\pi_s\theta = x_i\sigma_s{\downarrow}\pi_s\pi_s^{-1}\tau_s = x_i\sigma_s{\downarrow}\tau_s = x_i\tau{\downarrow}$. Analogously, $x_i\sigma_t{\downarrow}\pi_t\theta = x_i\tau{\downarrow}$. Therefore θ unifies the elements on the others positions in the two tuples as well.

   As θ unifies $\mathsf{tuple}(s\sigma_s{\downarrow}\pi_s, x_1\sigma_s{\downarrow}\pi_s, \ldots, x_n\sigma_s{\downarrow}\pi_s)$ and $\mathsf{tuple}(t\sigma_t{\downarrow}\pi_t, x_1\sigma_t{\downarrow}\pi_t, \ldots, x_n\sigma_t{\downarrow}\pi_t)$, it follows that the two terms have a most general unifier σ of which θ is an instance: $\theta = \sigma\theta'$.

   Let $\gamma = \theta'$ and let

$$\begin{array}{rcll} \omega & = & \sigma_s{\downarrow}\pi_s\sigma[\mathcal{V}ar(s) \setminus \mathcal{V}ar(t)] & \uplus \\ & & \sigma_s{\downarrow}\pi_s\sigma[\mathcal{V}ar(s) \cap \mathcal{V}ar(t)] & \uplus \\ & & \sigma_t{\downarrow}\pi_t\sigma[\mathcal{V}ar(t) \setminus \mathcal{V}ar(s)]. & \end{array}$$

   By the definition of Σ, we have that $\omega \in \Sigma$. We will show that $\tau{\downarrow}[\mathcal{V}ar(s,t)] = \omega\gamma[\mathcal{V}ar(s,t)]$.

   Indeed, let $x \in \mathcal{V}ar(s)$ be an arbitrary variable. We have that $x\omega\gamma = x(\sigma_s{\downarrow}\pi_s\sigma[\mathcal{V}ar(s)])\theta' = x\sigma_s{\downarrow}\pi_s\sigma\theta' = x\sigma_s{\downarrow}\pi_s\theta = x\sigma_s{\downarrow}\pi_s\pi_s^{-1}\tau_s = x\sigma_s{\downarrow}\tau_s = x\tau{\downarrow}$. Analogously, if $x \in \mathcal{V}ar(t)$, we have that $x\omega\gamma = x\tau{\downarrow}$.

   As $x\omega\gamma = x\tau{\downarrow}$ for any $x \in \mathcal{V}ar(s) \cup \mathcal{V}ar(t)$, we have that $\omega\gamma[\mathcal{V}ar(s,t)] =_{\mathsf{E}} \tau[\mathcal{V}ar(s,t)]$. As τ was chosen arbitrarily, it follows that Σ is a complete set of equational unifiers of s and t.

We have shown that Σ is a finite (first item), complete set (third item) of E-unifiers (second item) of s and t. □

Theorem 3.3 shows that any equational theory that can be implemented by term rewriting system having the strong finite variant property is finitary. Furthermore, it shows how a complete set of unifiers of two terms can be effectively constructed from strongly complete sets of variants of the terms in question. As an immediate corollary, we have that convergent optimally reducing equational theories are finitary and that finite complete sets of unifiers are effectively constructible.

3.7 Conclusion and Tool Support

We have created a prototype implementation of the algorithm in Section 3.5 for computing strongly complete sets of variants in the SubVariant tool [48].

The SubVariant tool is written in OCaml. The goal of the tool is to demonstrate the implementation of the algorithm and it was not designed with efficiency in mind. In particular, the implementation of the algorithm used to compute the most general unifier takes exponential time in the worst case.

The tool receives as input a term rewriting system R and a term t and it computes a finite set of variants of t with respect to the rewrite system R. If the rewrite system is convergent and optimally reducing, the computed set is guaranteed to be a strongly complete set of variants of t. For other convergent rewrite systems (that are not optimally reducing) the set of variants is sound (due to the soundness of the narrowing rule) but it is not in general complete, since the narrowing rule is only guaranteed to be complete when the term rewriting system is optimally reducing.

In addition to computing strongly complete sets of variants, the tool also can also compute sets of unifiers for an equational unification problem. Again, if the underlying rewrite system is optimally reducing, the result is guaranteed to be a complete set of unifiers for the given equation.

In our experiments, the tool SubVariant finished instantaneously on small, hand chosen examples. However, it can be made to take exponential time by using specially crafted examples, since the number of variants of a term can be exponential, as shown in the following example.

> *Example* 3.7. Let $t = \mathsf{tuple}(\mathrm{dec}(x_1, y_1), \ldots, \mathrm{dec}(x_n, y_n))$ be a term and consider the following subterm convergent rewrite system $\mathsf{R} = \{\mathrm{dec}(\mathrm{enc}(x, y), y) \to x\}$ (R is optimally reducing as well). The symbol tuple is a free symbol of arity n.
>
> In any strongly complete set of variants of t, there must be a variant for every possible choice of success or failure of the decryption operations $\mathrm{dec}(x_i, y_i)$ (for $1 \leq i \leq n$). Therefore, any strongly complete set of variants of t will have size at least 2^n.

The equational unification algorithm for two terms s and t will create a complete set of unifiers that is of size at most $|\mathcal{V}ariants(s)| \times |\mathcal{V}ariants(t)|$, since at most an equational unifier is added for each pair of strong variants of s and t. Therefore, the equational unification algorithm will introduce at most a quadratic blow-up compared to the algorithm for finding strongly complete set of variants.

In general, there is no guarantee that the strongly complete set of variants computed by the algorithm in Section 3.5 is *minimal* in the sense that any strict subset of it is not strongly complete. However, it is easy to obtain minimal strongly complete sets of variants by a post-processing step that removes redundant variants.

Similarly, there is no guarantee that the complete set of equational unifiers found by the algorithm in Section 3.6 is minimal but can be rendered minimal by a post-processing step.

As future work, it would be interesting to study the notion of SFVP in the presence of an equational theory such as AC, to obtain algorithms for finding strongly complete sets of variants and complete sets of unifiers modulo such a theory and to apply these algorithms in the verification of security protocols.

Chapter 4

A Decision Procedure for Static Equivalence

4.1 Introduction

We have seen in Chapter 1 that cryptographic protocols are small distributed programs that make use of cryptographic primitives such as encryption and digital signatures to communicate securely over a network. We have also shown that it is essential to gain as much confidence as possible in their security because insecurity can have important negative consequences.

Symbolic methods have been developed to analyze such protocols [12, 107, 118]. In these approaches, one of the most important aspects is to be able to reason about the *knowledge* of the attacker. Traditionally, the knowledge of the attacker is expressed in terms of *deducibility* (e.g. [118, 39]). A message s (intuitively the secret) is said to be deducible from a set of messages φ, if an attacker is able to compute s from φ. To perform this computation, the attacker is allowed, for example, to decrypt deducible messages by deducible keys.

However, deducibility is not always sufficient. Consider for example the case where a protocol participant sends over the network the encryption of one of the constants "yes" or "no" (e.g. the value of a vote). Deducibility is not the right notion of knowledge in this case, since both possible values ("yes" and "no") are indeed "known" to the attacker. In this case, a more adequate form of knowledge is *indistinguishability* (e.g. [2]): is the attacker able to distinguish between two transcripts of the protocol, one running with the value "yes" and the other one running with the value "no"?

4.1.1 Related Work

Many decision procedures have been proposed for deducibility (e.g. [39, 8, 100, 40]) under a variety of equational theories modelling encryption, digital signatures, exclusive or, and homomorphic operators. Several papers are also devoted to the study of static equivalence. Most of these results introduce a new procedure for each particular theory and even in the case of the general decidability criterion given in [2, 58], the algorithm underlying the proof has to be adapted for each particular theory, depending on how the criterion is fulfilled. A combination result was obtained in [13]: if deduction and static equivalence are decidable for two disjoint equational theories, then deduction and static equivalence are also decidable for the union of the two theories.

The first generic algorithm that has been proposed handles subterm convergent equational theories [2] and covers the classical theories for encryption and signatures. This result is encompassed by the recent work of Baudet *et al.* [21, 22] in which the authors propose a generic procedure that works for any convergent equational theory, but which may fail or not terminate. This procedure has been implemented in the YAPA tool [20] and has been shown to terminate without failure in several cases (e.g. subterm convergent theories and blind signatures). However, due to its simple

representation of deducible terms (represented by a finite set of *ground* terms), the procedure fails on several interesting equational theories like the theory of trapdoor commitments. Our representation of deducible terms overcomes this limitation by including terms with variables which can be substituted by any deducible terms. Independently of our work, specific decision procedures for the theory of trapdoor commitment and that of reencryption have been presented in [24], but have not been implemented.

Another tool that can be used to check static equivalence is ProVerif [25, 27], although this tool is overkill in the sense that it was designed to handle a more general problem: observational equivalence of two processes. ProVerif can handle various equational theories and analyze security protocols under active adversaries but termination is not guaranteed in general and the tool performs safe approximations.

4.1.2 Contribution

In this chapter, we present a decision procedure for checking static equivalence under certain equational theories. Static equivalence models indistinguishability of two sequences of messages. Intuitively, this notion does not take into account the dynamic behavior of the protocol. Nevertheless, in order to establish that two dynamic behaviors of a protocol are indistinguishable, an important subproblem is to establish indistinguishability between the sequences of messages generated by the protocol [118, 3]. We will come back to the dynamic behavior of protocols in Chapter 5.

Static equivalence also plays an important role in the study of guessing attacks (e.g. [56, 19]), as well as for anonymity properties in e-voting protocols (e.g. [72, 17]). This was actually the starting point of this work. During the study of e-voting protocols, we came across several equational theories for which we needed to show static equivalence while no decision procedure for deduction or static equivalence existed.

Our procedure is based on modeling the frames into first-order Horn clauses which we describe in Section 4.2. We then present a saturation strategy (Section 4.3) for the set of Horn clauses such that static equivalence of two frames can easily be checked from the saturated sets of Horn. We prove that the saturation process is sound and complete in Section 4.4. We show how to decide static equivalence and the intruder deduction problem from the saturated knowledge base(s) in Section 4.5.

Both static equivalence and the intruder deduction problem are undecidable, even when the equational theory is convergent [2]. Therefore, our saturation procedure does not always terminate. In Section 4.6, we illustrate examples of non-termination and we identify several useful (classes of) equational theories where it does terminate.

As a byproduct we obtain an algorithm for the intruder deduction problem, namely the problem of deciding if a message can be computed by the intruder from a given set of messages.

This chapter is based on work that has been published [50, 51]. The algorithms described here have been implemented in the tool KiSs [45] that we discuss in Section 4.7. We conclude and present possible future work in Section 4.8

Our method extends previous work [2] most notably by providing a decision procedure for an equational theory modeling trap-door commitment. This equational theory is particularly relevant to voting protocols, where it is used to provide forms of coercion-resistance [72]. However, none of the previous work provided a decision procedure for static equivalence in the case of this equational theory.

Example 4.1. As a running example for this section we will consider the rewrite system

$$\mathsf{R}_{\mathrm{mal}} \quad = \quad \{ \quad \begin{array}{l} \mathrm{dec}(\mathrm{enc}(x,y),y) \to x \\ \mathrm{mal}(\mathrm{enc}(x,y),z) \to \mathrm{enc}(z,y)\}, \end{array}$$

modeling a hypothetical *malleable* encryption algorithm that allows one to obtain via the mal function symbol the encryption of any known term with an unknown key, as long as a prior encryption with that key is available. While such an encryption algorithm would

be too far-fetched to be used in practice, this rewrite system allows us to illustrate our procedure. In particular, the theory cannot be handled by current procedures [20], but it is simple to present and it presents the same issue as more practical theories such as trapdoor commitment.

We will consider the frame $\varphi_1 = \{w_1 \mapsto \text{enc}(a, k)\}$, where $a, k \in \mathcal{N}$ are private names and the frame $\varphi_2 = \{w_1 \mapsto \text{enc}(b, k)\}$ where $b \in \mathcal{M}$ is a public name. The two frames represent two runs of a protocol: in the first run, the protocol outputs on the network the encryption of a private name a with a private key k and in the second run the protocol outputs the encryption of a public name b with the private key k. The intruder can tell the difference between the two runs as follows:

1. apply the function symbol mal to the parameter w_1 (which will "evaluate" to $\text{enc}(a, k)$ or to $\text{enc}(b, k)$, depending on the run) and to the term b (in the first run, the resulting term will be $\text{mal}(\text{enc}(a, k), b) =_{\mathsf{R}_{\text{mal}}} \text{enc}(b, k)$ and in the second run $\text{mal}(\text{enc}(b, k), b) =_{\mathsf{R}_{\text{mal}}} \text{enc}(b, k)$),
2. compare the resulting term to w_1: if they are equal then we are in the second run; otherwise in the first.

The fact that the intruder can tell the difference between the two frames is because the two frames are not statically equivalent. This follows because $\text{mal}(w_1, b) = w_1$ is a test that holds in φ_2 ($(\text{mal}(w_1, b) = w_1)\varphi_2$) but not in φ_1 ($(\text{mal}(w_1, b) \neq w_1)\varphi_1$).

Note that the above inequivalence is due to the choice of the rewrite system. Had we chosen to compare the two frames φ_1 and φ_2 with respect to the classical rewrite system $\mathsf{R} = \{\text{dec}(\text{enc}(x, y), y) \to x\}$, we would have found that φ_1 is statically equivalent to φ_2. The fact that the two frames are not statically equivalent with respect to R_{mal} is due to the additional power given to the intruder by the mal function symbol.

4.2 Modeling Frames as Sets of Horn Clauses

As we have already announced, our procedure is based on a modeling of frames into first order Horn clauses. A refinement of resolution described in Section 4.3 is then applied to the set of Horn clauses in order to obtain an equivalent saturated set of Horn clauses that we called *solved*. The set of solved Horn clauses is equivalent in a sense made precise by Theorem 4.1 to the initial set of Horn clauses. However, starting from the solved clauses associated to two frames, it is easy to decide static equivalence as shown in Section 4.5. In order to obtain termination of the saturation procedure (shown in Section 4.6), our refinement of resolution makes heavy use of the form of redundancy elimination described in Section 4.3.

We now show how we model frames as first order Horn clauses.

Predicates. To construct the Horn clauses, we consider the usual first-order logic connectives, with two additional binary predicates, k and i. The $\mathsf{k}(R, t)$ predicate intuitively denotes that R is a recipe for t while the $\mathsf{i}(R, R')$ predicate intuitively denotes that R and R' are recipes for the same term.

We say that a substitution σ is *well-formed* with respect to a formula if $t\sigma$ does not contain any parameter $w_i \in \mathcal{W}$ for any term t appearing as the second argument of a k predicate in the formula and if $R\sigma$ does not contain any private name $n \in \mathcal{N}$ for any term R appearing as the first argument of a k predicate or as an argument of the i predicate. A formula is *well-formed* if the identity substitution is well-formed with respect to the formula.

Well-formed first-order formulas are interpreted over frames φ and well-formed valuations σ. (i.e. well-formed substitutions associating a ground term to each free variable in the formula).

The semantics of the two predicates for ground arguments is given in Figure 4.1. The semantics of the first-order connectives $\wedge, \vee, \neg$ are defined as usual. The semantics of the quantifiers is

defined as usual, but restricting the domain of the variables such that the resulting valuation is well-formed. A well-formed formula is valid in a frame if it is true when interpreted over all well-formed valuations. From hereon, we will assume that all formulas are well-formed.

Predicates:
$\mathsf{k}(R,t)$ (intruder Knowledge predicate)
$\mathsf{i}(R,R')$ (intruder Identity predicate)

Semantics:
$\varphi \models \mathsf{k}(R,t)$ if $\varphi \vdash^R t$
$\varphi \models \mathsf{i}(R,R')$ if $(R = R')\varphi$

Figure 4.1: Predicates (for ground terms t, R)

The "intruder knowledge predicate" $\mathsf{k}(R,t)$ intuitively states that R is a recipe for t in the frame φ and the "intruder identity predicate" $\mathsf{i}(R,R')$ intuitively states that $R = R'$ is an identity that holds in the frame φ (i.e. $(R = R')\varphi$).

Example 4.2. Continuing the previous example, the ground formula $\mathsf{k}(w_1, \mathrm{enc}(a,k))$ is true in the frame φ_1 but not in the frame φ_2. The formula $\mathsf{k}(b,b)$ is true in both frames, since b is a public name and is recipe of itself in any frame. The formula $\mathsf{k}(a,a)$ is not well-formed since a is a private name and it appears as a first argument of the k predicate.

The formula $\forall X, Y.(\mathsf{k}(\mathrm{mal}(X,Y), \mathrm{mal}(x,y)) \Leftarrow \mathsf{k}(X,x) \wedge \mathsf{k}(Y,y))$ is true in all frames (The symbol "$\Leftarrow$" denotes implication) since if X is a recipe for x and Y is a recipe of y then $\mathrm{mal}(X,Y)$ will be a recipe for $\mathrm{mal}(x,y)$.

The formula $\mathsf{i}(w_1, \mathrm{mal}(w_1, b))$ is true in φ_2 but not in φ_1 (this is the reason that the two frames are not statically equivalent). The formula $\mathsf{i}(w_1, \mathrm{mal}(w_1, a))$ is not well-formed since the private name a appears in one of the arguments of the i predicate.

Statements. Our algorithm works with a particular class of Horn clauses that we call *statements*.

Definition 4.1. A *statement* is a Horn clause of the form

$$f = \Big(\mathsf{k}(R,t) \Leftarrow \mathsf{k}(X_1,t_1), \ldots, \mathsf{k}(X_n,t_n)\Big)$$

or

$$f = \Big(\mathsf{i}(R,R') \Leftarrow \mathsf{k}(X_1,t_1), \ldots, \mathsf{k}(X_n,t_n)\Big)$$

such that:

1. $\mathcal{V}ar(R,R',X_1,\ldots,X_n) \cap \mathcal{V}ar(t,t_1,\ldots,t_n) = \emptyset$,
2. $\mathcal{V}ar(t) \subseteq \mathcal{V}ar(t_1,\ldots,t_n)$ and
3. $X_1,\ldots,X_n$ are distinct variables.

In the first case in the above definition, we call f a *deduction statement* and in the second case we call f an *identity statement* or an *equational statement*. We say that a statement is *solved* if $t_i \in \mathcal{X}$ for all $1 \leq i \leq n$.

A statement is *well-formed* if it is not a deduction statement, or if it is a deduction statement that is not solved, or if it is a deduction statement that is solved and $t \notin \mathcal{X}$. In other words, a statement is not well-formed only if it is a solved deduction statement with $t \in \mathcal{X}$ being a variable.

As statements are Horn clauses, we identify them up to bijective renaming of variables and up to permutation of the antecedents. In inferences rules that use statements, we assume as expected that the variables in the statements are renamed to fresh variables.

Example 4.3. The statement $\mathsf{k}(w_1, \mathrm{enc}(a,k)) \Leftarrow$ (a Horn clause with an empty body) is true in φ_1 but not in φ_2. The statement $\mathsf{i}(w_1, X) \Leftarrow \mathsf{k}(X, \mathrm{enc}(b,k))$ is true in φ_2 but not in φ_1. The statement $\mathsf{i}(X,Y) \Leftarrow \mathsf{k}(X,x), \mathsf{k}(Y,x)$ is true of all frames.

4.3 Saturation Procedure

Derivable statements. Let K be a set of statements and t, R be terms.

Definition 4.2. We say that the predicate $\mathsf{k}(R,t)$ is derivable from K (and we write $\mathsf{k}(R,t) \in \mathbf{deriv}(K)$) if:

1. either t and R are the same variable ($t = R$ and $t \in X$)
2. or there exist a statement $f = \Big(\mathsf{k}(R_0,t_0) \Leftarrow \mathsf{k}(X_1,t_1), \ldots, \mathsf{k}(X_n,t_n)\Big) \in K$, some terms $R_1, \ldots, R_n$ and a substitution σ with $\mathcal{D}om(\sigma) \subseteq \mathcal{V}ar(t_0)$ such that $t = t_0\sigma$, $R = R_0\{X_i \mapsto R_i\}_{1\leq i\leq n}$ and $\mathsf{k}(R_i, t_i\sigma) \in \mathbf{deriv}(K)$ for $1 \leq i \leq n$.

If K is a set of well-formed solved statements, there is a simple recursive algorithm that, given K and t, tests if there exists a term R such that $\mathsf{k}(R,t) \in \mathbf{deriv}(K)$ and constructs such an R if it exists. Termination is ensured by the fact that $|t_i\sigma| < |t|$ for all $1 \leq i \leq n$. Note that using memoization the algorithm works in polynomial time.

Example 4.4. Consider the following set K of statements (all of which are true in the frame φ_2 defined in the previous examples):

$$\left(\begin{array}{ccc} \mathsf{k}(w_1, \mathrm{enc}(b,k)) & \Leftarrow & \end{array}\right) \quad (f_1)$$
$$\left(\begin{array}{ccc} \mathsf{k}(b,b) & \Leftarrow & \end{array}\right) \quad (f_2)$$
$$\left(\begin{array}{ccc} \mathsf{k}(\mathrm{enc}(Y_1,Y_2), \mathrm{enc}(y_1,y_2)) & \Leftarrow & \mathsf{k}(Y_1,y_1), \mathsf{k}(Y_2,y_2) \end{array}\right) \quad (f_3)$$

We have that $\mathsf{k}(\mathrm{enc}(w_1,b), \mathrm{enc}(\mathrm{enc}(b,k),b))$ is derivable from K. This follows easily by instantiating the two antecedents of f_3 with f_1 and f_2, respectively.

Equational consequences. Given a finite set K of solved equational statements and terms M, N, we write $K \models \mathsf{i}(M,N)$ if $\mathsf{i}(M,N)$ is a consequence, in the usual first order theory of equality (for the binary symbol i) of

$$\{\mathsf{i}(R\sigma, R'\sigma) \mid \Big(\mathsf{i}(R,R') \Leftarrow \mathsf{k}(X_1,x_1), \ldots, \mathsf{k}(X_k,x_k)\Big) \in K\} \text{ where } \sigma = \{X_i \mapsto x_i\}_{1\leq i\leq k}.$$

Note that it may be the case that $x_i = x_j$ for some $i \neq j$ (whereas $X_i \neq X_j$ whenever $i \neq j$ by the definition of a statement).

Canonical form. We define for each statement f its *canonical form* $f{\Downarrow}$, obtained by first applying Rule Rename as much as possible and then Rule Remove as much as possible. The idea is to ensure that each variable x_i occurs at most once in the body and to get rid of those variables that do not occur in t. This will be particularly useful to characterize the form of solved statements when we prove termination of the saturation procedure. Unsolved statements and equational statements are kept unchanged by canonicalization.

$$\textsc{Rename} \quad \frac{\Big(\mathsf{k}(R,t) \Leftarrow \mathsf{k}(X_1,x_1), \ldots, \mathsf{k}(X_k,x_k)\Big) \;\; \{i,j\} \subseteq \{1,\ldots,n\} \;\; j \neq i \text{ and } x_j = x_i}{\Big(\mathsf{k}(R\{X_i \mapsto X_j\},t) \Leftarrow \mathsf{k}(X_1,x_1), \ldots, \mathsf{k}(X_{i-1},x_{i-1}), \mathsf{k}(X_{i+1},x_{i+1}), \ldots, \mathsf{k}(X_k,x_k)\Big)}$$

$$\textsc{Remove}\quad \frac{\Big(\mathsf{k}(R,t) \Leftarrow \mathsf{k}(X_1,x_1),\ldots,\mathsf{k}(X_k,x_k)\Big) \quad x_i \notin \mathcal{V}ar(t)}{\Big(\mathsf{k}(R,t) \Leftarrow \mathsf{k}(X_1,x_1),\ldots,\mathsf{k}(X_{i-1},x_{i-1}),\mathsf{k}(X_{i+1},x_{i+1}),\ldots,\mathsf{k}(X_k,x_k)\Big)}$$

Example 4.5. Consider the statement

$$f = \Big(\mathsf{k}(\mathrm{dec}(\mathrm{enc}(X_1,X_2),X_3),x_1) \Leftarrow \mathsf{k}(X_1,x_1),\mathsf{k}(X_2,y),\mathsf{k}(X_3,y)\Big).$$

To find the canonical form $f\Downarrow$ of f, we start by applying Rule Rename, after which we obtain the statement:

$$\Big(\mathsf{k}(\mathrm{dec}(\mathrm{enc}(X_1,X_2),X_2),x_1) \Leftarrow \mathsf{k}(X_1,x_1),\mathsf{k}(X_2,y)\Big).$$

We continue with the application of Rule Remove, after which we obtain the canonical form:

$$f\Downarrow = \Big(\mathsf{k}(\mathrm{dec}(\mathrm{enc}(X_1,X_2),X_2),x_1) \Leftarrow \mathsf{k}(X_1,x_1)\Big).$$

Note that it is important that Rule Remove is applied only after all instances of Rule Rename where treated. Otherwise, if a variable $x_i = x_j$ does not appear in the head of the statement, the fact that the recipes X_i and X_j must be recipes for the same term would be lost and therefore we would make an unsound inference.

Update. We group statements into *knowledge base*s.

Definition 4.3. A *knowledge base* is a set K set of well-formed statements in canonical form.

When adding a statement to a knowledge base, we perform some redundancy checks. The redundancy checks are performed through the *update function*.

Definition 4.4 (update)**.** Given a deduction statement $f = \Big(\mathsf{k}(R,t) \Leftarrow \mathsf{k}(X_1,t_1),\ldots,\mathsf{k}(X_n,t_n)\Big)$ and a knowledge base K, *the update of K by f*, written $K \oplus f$, is defined as:

$$\begin{cases} K \cup \{f\Downarrow\} & \text{if } f \text{ is solved and } \mathsf{k}(R,t) \notin \mathbf{deriv}(K) \text{ for any } R \quad \textsf{useful statement} \\ K \cup \Big\{\Big(\mathsf{i}(R',R\sigma) \Leftarrow \emptyset\Big)\Big\} & \text{if } f \text{ is solved and } R' \text{ is such that } \mathsf{k}(R',t) \in \mathbf{deriv}(K) \text{ and } \sigma = \{X_1 \mapsto t_1,\ldots,X_n \mapsto t_n\} \quad \textsf{redundant statement} \\ (K \cup \{f\}) & \text{if } f \text{ is not solved} \quad \textsf{unsolved statement} \end{cases}$$

Given an equational statement f and a knowledge base K, the update $K \oplus f$ of K by f is simply $K \cup \{f\}$.

The choice of the recipe R' in the *redundant statement* case is defined by the implementation. While this choice does not influence the correctness of the saturation procedure, it might influence its termination as we will see later.

Note that the result of updating a knowledge base by a (possibly not well-formed and/or not canonical) statement is again a knowledge base. Statements that are not well-formed will be captured by the *redundant statement* case, which adds an equational statement instead.

The role of the update function is to add statements to the knowledge base, while performing some redundancy elimination. If $\mathsf{k}(R,t) \notin \mathbf{deriv}(K)$ for any R, then the statement to be added clearly provides interesting information and it is added to the knowledge base.

If the statement to be added is unsolved, it is added anyway (because it might prove useful later on).

If the statement to be added is solved and $\mathsf{k}(R,t) \in \mathbf{deriv}(K)$ for some R, then this deduction statement does not provide new information about deducible terms, but it might provide a new recipe for terms we already know deducible. Therefore, an equational statement is added instead, stating that the two recipes are equal provided the antecedents are satisfied.

Example 4.6. We consider the knowledge base K containing the following statements:

$$\left\{\begin{array}{rcll} \mathsf{k}(w_1, \mathrm{enc}(b,k)) & \Leftarrow & & \end{array}\right\} \quad (f_1)$$
$$\left\{\begin{array}{rcll} \mathsf{k}(b,b) & \Leftarrow & & \end{array}\right\} \quad (f_2)$$
$$\left\{\begin{array}{rcll} \mathsf{k}(\mathrm{enc}(Y_1,Y_2), \mathrm{enc}(y_1,y_2)) & \Leftarrow & \mathsf{k}(Y_1,y_1), \mathsf{k}(Y_2,y_2) & \end{array}\right\} \quad (f_3)$$

We have already seen that $\mathsf{k}(\mathrm{enc}(w_1,b), \mathrm{enc}(\mathrm{enc}(b,k),b)) \in \mathbf{deriv}(K)$. Updating the knowledge base by $\left(\mathsf{k}(w_2, \mathrm{enc}(\mathrm{enc}(b,k),b)) \Leftarrow\right)$ would result in no modification of the set of deduction statements, since we already know that $\mathsf{k}(R', \mathrm{enc}(\mathrm{enc}(b,k),b)) \in \mathbf{deriv}(K)$ (with $R' = \mathrm{enc}(w_1,b)$). However, a new equational statement $\left(\mathsf{i}(w_2, \mathrm{enc}(w_1,b)) \Leftarrow\right)$ would be added instead.

Initialisation. Given a frame $\varphi = \{w_1 \mapsto t_1, \ldots, w_n \mapsto t_n\}$, our procedure starts from an *initial knowledge base* associated to φ and defined as follows:

$$\begin{array}{lll} K_i(\varphi) = & & \emptyset \\ & \bigoplus_{1 \le i \le n} & \left(\mathsf{k}(w_i, t_i) \Leftarrow\right) \\ & \bigoplus_{n \in (\mathcal{N}ames(\varphi) \cap \mathcal{M})} & \left(\mathsf{k}(n,n) \Leftarrow\right) \\ & \bigoplus_{f \in \mathcal{F}} & \left(\mathsf{k}(f(X_1,\ldots,X_k), f(x_1,\ldots,x_k)) \Leftarrow \mathsf{k}(X_1,x_1), \ldots, \mathsf{k}(X_k,x_k)\right) \\ & & \text{where } k = \mathsf{ar}(f) \text{ is the arity of } f \end{array}$$

Example 4.7. We will continue with the frames $\varphi_1 = \{w_1 \mapsto \mathrm{enc}(a,k)\}$, where $a \in \mathcal{N}$ is a private name and $\varphi_2 = \{w_1 \mapsto \mathrm{enc}(b,k)\}$ with $k \in \mathcal{N}$ being a private name and $b \in \mathcal{M}$ being a public name defined in the previous examples. The knowledge base $K_i(\varphi_2)$ is the following set of statements:

$$\left\{\begin{array}{rcll} \mathsf{k}(w_1, \mathrm{enc}(b,k)) & \Leftarrow & & \end{array}\right\} \quad (f_1)$$
$$\left\{\begin{array}{rcll} \mathsf{k}(b,b) & \Leftarrow & & \end{array}\right\} \quad (f_2)$$
$$\left\{\begin{array}{rcll} \mathsf{k}(\mathrm{enc}(Y_1,Y_2), \mathrm{enc}(y_1,y_2)) & \Leftarrow & \mathsf{k}(Y_1,y_1), \mathsf{k}(Y_2,y_2) & \end{array}\right\} \quad (f_3)$$
$$\left\{\begin{array}{rcll} \mathsf{k}(\mathrm{dec}(Y_1,Y_2), \mathrm{dec}(y_1,y_2)) & \Leftarrow & \mathsf{k}(Y_1,y_1), \mathsf{k}(Y_2,y_2) & \end{array}\right\} \quad (f_4)$$
$$\left\{\begin{array}{rcll} \mathsf{k}(\mathrm{mal}(Y_1,Y_2), \mathrm{mal}(y_1,y_2)) & \Leftarrow & \mathsf{k}(Y_1,y_1), \mathsf{k}(Y_2,y_2) & \end{array}\right\} \quad (f_5)$$

Saturation. The idea of the saturation procedure is to keep adding valid statements to the knowledge base such as to produce:

1. a set of solved deduction statements which have the same set of syntactic consequences as the initial set of deduction statements modulo the equational theory and

2. a set of solved equational statements whose consequences are exactly the equations holding in the frame.

The main part of this procedure consists in saturating the initial knowledge base $K_i(\varphi)$ by means of the transformation rules described in Figure 4.2.

NARROWING

$$\frac{f = \Big(\mathsf{k}(M, C[t]) \Leftarrow \mathsf{k}(X_1, x_1), \ldots, \mathsf{k}(X_k, x_k)\Big) \in K \quad l \to r \in \mathsf{R},\ t \notin \mathcal{X},\ \sigma = \mathsf{mgu}(l, t) \text{ and } \mathcal{V}ar(f) \cap \mathcal{V}ar(l) = \emptyset.}{K \Longrightarrow K \oplus f_0}$$

where $f_0 = \Big(\mathsf{k}(M, (C[r])\sigma) \Leftarrow \mathsf{k}(X_1, x_1)\sigma, \ldots, \mathsf{k}(X_k, x_k)\sigma\Big)$.

F-SOLVING

$$\frac{f_1 = \Big(\mathsf{k}(M, t) \Leftarrow \mathsf{k}(X, u), \mathsf{k}(X_1, t_1), \ldots, \mathsf{k}(X_k, t_k)\Big) \in K \quad f_2 = \Big(\mathsf{k}(N, s) \Leftarrow \mathsf{k}(Y_1, y_1), \ldots, \mathsf{k}(Y_\ell, y_\ell)\Big) \in K \quad \text{with } u \notin \mathcal{X},\ \sigma = \mathsf{mgu}(s, u) \text{ and } \mathcal{V}ar(f_1) \cap \mathcal{V}ar(f_2) = \emptyset.}{K \Longrightarrow K \oplus f_0}$$

where $f_0 = \Big(\mathsf{k}(M\{X \mapsto N\}, t\sigma) \Leftarrow \{\mathsf{k}(X_i, t_i\sigma)\}_{1 \le i \le k} \cup \{\mathsf{k}(Y_i, y_i\sigma)\}_{1 \le i \le \ell}\Big)$.

UNIFYING

$$\frac{f_1 = \Big(\mathsf{k}(M, t) \Leftarrow \mathsf{k}(X_1, x_1), \ldots, \mathsf{k}(X_k, x_k)\Big) \in K \quad f_2 = \Big(\mathsf{k}(N, s) \Leftarrow \mathsf{k}(Y_1, y_1), \ldots, \mathsf{k}(Y_\ell, y_\ell)\Big) \in K \quad \text{with } \sigma = \mathsf{mgu}(s, t) \text{ and } \mathcal{V}ar(f_1) \cap \mathcal{V}ar(f_2) = \emptyset.}{K \Longrightarrow K \cup \{f_0\}}$$

where $f_0 = \Big(\mathsf{i}(M, N) \Leftarrow \{\mathsf{k}(X_i, x_i\sigma)\}_{1 \le i \le k} \cup \{\mathsf{k}(Y_i, y_i\sigma)\}_{1 \le i \le \ell}\Big)$.

E-SOLVING

$$\frac{f_1 = \Big(\mathsf{i}(U, V) \Leftarrow \mathsf{k}(Y, s), \mathsf{k}(X_1, t_1), \ldots, \mathsf{k}(X_k, t_k)\Big) \in K \quad f_2 = \Big(\mathsf{k}(M, t) \Leftarrow \mathsf{k}(Y_1, y_1), \ldots, \mathsf{k}(Y_\ell, y_\ell)\Big\} \in K \quad \text{with } s \notin \mathcal{X},\ \sigma = \mathsf{mgu}(s, t) \text{ and } \mathcal{V}ar(f_1) \cap \mathcal{V}ar(f_2) = \emptyset.}{K \Longrightarrow K \cup \{f_0\}}$$

where $f_0 = \Big(\mathsf{i}(U\{Y \mapsto M\}, V\{Y \mapsto M\}) \Leftarrow \{\mathsf{k}(X_i, t_i\sigma)\}_{1 \le i \le k} \cup \{\mathsf{k}(Y_i, y_i\sigma)\}_{1 \le i \le \ell}\Big)$.

Figure 4.2: Saturation rules

The rule NARROWING is designed to apply a rewriting step on an existing deduction statement. Intuitively, this rule allows us to get rid of the equational theory and nevertheless ensures that the set of derivable predicates remains complete. This rule might introduce unsolved antecedents in the statement. The rule F-SOLVING is then used to instantiate the unsolved antecedents of an existing deduction statement. UNIFYING and E-SOLVING add equational statements which memorize when different recipes for the same term exist.

Note that this procedure may not terminate and that the fixed point may not be unique (the order in which new statements are added by the $\oplus$ operation matters).

We write $\Longrightarrow^*$ for the reflexive and transitive closure of $\Longrightarrow$.

Example 4.8. Continuing Example 4.7, we illustrate the saturation procedure. We can apply the rule NARROWING on the statement f_4 and the rewrite rule $\mathrm{dec}(\mathrm{enc}(x, y), y) \to x$, as well as on the statement f_5 and the rewrite rule $\mathrm{mal}(\mathrm{enc}(x, y), z) \to \mathrm{enc}(z, y)$, thereby adding the statements:

$$\Big(\mathsf{k}(\mathrm{dec}(Y_1, Y_2), x) \Leftarrow \mathsf{k}(Y_1, \mathrm{enc}(x, y)), \mathsf{k}(Y_2, y)\Big) \qquad (f_6)$$

$$\Big(\mathsf{k}(\mathrm{mal}(Y_1, Y_2), \mathrm{enc}(z, y)) \Leftarrow \mathsf{k}(Y_1, \mathrm{enc}(x, y)), \mathsf{k}(Y_2, z)\Big) \qquad (f_7)$$

The statements f_6 and f_7 are not solved and we can apply the rule F-SOLVING with f_1,

adding the statements:

$$\begin{aligned} \Big(\mathsf{k}(\mathrm{dec}(w_1, Y_2), b) &\Leftarrow \mathsf{k}(Y_2, k)\Big) && (f_8)\\ \Big(\mathsf{k}(\mathrm{mal}(w_1, Y_2), \mathrm{enc}(z, k)) &\Leftarrow \mathsf{k}(Y_2, z)\Big) && (f_9) \end{aligned}$$

Rule UNIFYING can be used on statements f_1/f_3, f_3/f_9 as well as f_1/f_9 to add equational statements. This third case allows one to obtain $f_{10} = \Big(\mathsf{i}(w_1, \mathrm{mal}(w_1, Y_2)) \Leftarrow \mathsf{k}(Y_2, b)\Big)$ which can be solved (using E-SOLVING with f_2) to obtain $f_{11} = \Big(\mathsf{i}(w_1, \mathrm{mal}(w_1, b)) \Leftarrow \Big)$, etc. When reaching a fixed point, f_9, f_{11} and the statements in $\mathrm{Init}(\varphi_2)$ are some of the solved statements contained in the knowledge base.

4.4 Soundness and Completeness

In this section we prove the soundness and completeness of the saturation procedure:

Theorem 4.1. *Let φ be a frame and K be a saturated knowledge base such that $\mathrm{Init}(\varphi) \Longrightarrow^* K$. Let $t \in T(\mathcal{F})$ and let $K^+ = K \cup \{\Big(\mathsf{k}(n, n) \Leftarrow \Big) \mid n \in \mathcal{N}ames(t) \setminus \mathcal{N}\}$. Then*

1. *for all terms M we have that:*

$$\varphi \vdash^M t \text{ iff there exists } N \text{ such that } K|_{\mathsf{solved}} \models \mathsf{i}(M, N) \text{ and } \mathsf{k}(N, t{\downarrow}) \in \mathbf{deriv}(K^+|_{\mathsf{solved}})$$

2. *and for all terms M, N we have that*

$$(M = N)\varphi \text{ iff } K|_{\mathsf{solved}} \models \mathsf{i}(M, N).$$

We begin by showing the soundness of the saturation procedure.

4.4.1 Soundness

We first show that the update operator ($\oplus$) is sound:

Lemma 4.1. *Let φ be a frame and K be a knowledge base such that every statement in K holds in φ. Let f_0 be a statement that holds in φ. Then every statement in $K \oplus f_0$ holds in φ.*

Proof. We immediately have that if f holds in φ, then $f{\Downarrow}$ holds in φ. Therefore, if we are in the useful fact or unsolved fact cases, the lemma obviously holds.

We also have that if $\mathsf{k}(R, t) \in \mathbf{deriv}(K)$, then $\mathsf{k}(R, t)$ is valid in φ. Therefore, in the redundant statement case, we have that R' and $R\sigma$ are recipes for the same term t. Therefore, $\mathsf{i}(R', R\sigma)$ is also valid in φ.

□

Next we show that any statement in a knowledge base obtained by saturating the initial knowledge base holds in the frame.

Lemma 4.2. *Let φ be a frame and K be a knowledge base such that $\mathrm{Init}(\varphi) \Longrightarrow^* K$. Then every statement $f \in K$ holds in φ.*

Proof. By induction on the derivation $\mathrm{Init}(\varphi) \Longrightarrow^* K$.

Base case: We have that $K = \mathrm{Init}(\varphi)$. To conclude, we have to show that the statements that are added to the initial knowledge base hold in φ.

There are three kind of deduction statements that can be added to the knowledge base: the statements that come from φ, those of the form $\Big(\mathsf{k}(n,n) \Leftarrow \Big)$ for $n \in \mathcal{N}ames(\varphi) \setminus \mathcal{N}$ and those of the form:

$$\Big(\mathsf{k}(f(X_1,\ldots,X_k), f(x_1,\ldots,x_k)) \Leftarrow \mathsf{k}(X_1,x_1),\ldots,\mathsf{k}(X_k,x_k)\Big).$$

It is easy to see that all these statements hold in φ and we can conclude by Lemma 4.1.

Induction step:

In such a case, we have $\mathrm{Init}(\varphi) \Longrightarrow^* K' \Longrightarrow K$. We perform a case analysis on the inference rule used in $K' \Longrightarrow K$. For each rule, we show that the resulting statements f_0 holds in φ and we conclude by relying on Lemma 4.1.

Rule NARROWING: Let $f = \Big(\mathsf{k}(M, C[t]) \Leftarrow \mathsf{k}(X_1,x_1),\ldots,\mathsf{k}(X_k,x_k)\Big)$ be the deduction statement, $l \to r \in \mathsf{R}$ be the rewrite rule and $\sigma = \mathsf{mgu}(l,t)$ be the substitution involved in this step. Let $f_0 = \Big(\mathsf{k}(M,(C[r])\sigma) \Leftarrow \mathsf{k}(X_1,x_1\sigma),\ldots,\mathsf{k}(X_k,x_k\sigma)\Big)$ be the resulting deduction statement.

We show that f_0 holds in φ. Let τ be a substitution such that $\varphi \vdash^{M_i} x_i\sigma\tau$for $1 \le i \le k$. Since f holds in φ, we have that $\varphi \vdash^{M'} (C[t])\sigma\tau$ for $M' = M\{X_1 \mapsto M_1,\ldots,X_k \mapsto M_k\}$. It is easy to see that the following equalities are satisfied:

$$(C[t])\sigma\tau = (C[l])\sigma\tau =_{\mathsf{R}} (C[r])\sigma\tau$$

Therefore $\varphi \vdash^{M'} (C[r])\sigma\tau$, and thus f_0 holds in φ.

Rule F-SOLVING: Let $f_1 = \Big(\mathsf{k}(M,t) \Leftarrow \mathsf{k}(X_0,t_0),\ldots,\mathsf{k}(X_k,t_k)\Big)$ with $t_0 \notin \mathcal{X}$ and $f_2 = \Big(\mathsf{k}(N,s) \Leftarrow \mathsf{k}(Y_1,y_1),\ldots,\mathsf{k}(Y_\ell,y_\ell)\Big)$ be the two deduction statements and $\sigma = \mathsf{mgu}(s,t_0)$ be the substitution involved in this step. Let f_0 be the resulting deduction statement:

$$f_0 = \Big(\mathsf{k}(M\{X_0 \mapsto N\}, t\sigma) \Leftarrow \mathsf{k}(X_1,t_1\sigma),\ldots,\mathsf{k}(X_k,t_k\sigma),\mathsf{k}(Y_1,y_1\sigma),\ldots,\mathsf{k}(Y_\ell,y_\ell\sigma)\Big).$$

We show that f_0 holds in φ. Let τ be a substitution such that $\varphi \vdash^{M_i} t_i\sigma\tau$ for $1 \le i \le k$ and $\varphi \vdash^{N_j} y_j\sigma\tau$ for $1 \le j \le \ell$). Since f_2 holds in φ, we have that $\varphi \vdash^{N'} s\sigma\tau$ where $N' = N\{Y_1 \mapsto N_1,\ldots,Y_\ell \mapsto N_\ell\}$. Since f_1 holds in φ and $s\sigma\tau = t_0\sigma\tau$, we deduce that $\varphi \vdash^{M''} t\sigma\tau$ where

$$\begin{aligned} M'' &= M\{X_0 \mapsto N', X_1 \mapsto M_1,\ldots,X_k \mapsto M_k\} \\ &= (M\{X_0 \mapsto N\})\{X_1 \mapsto M_1,\ldots,X_k \mapsto M_k, Y_1 \mapsto N_1,\ldots,Y_\ell \mapsto N_\ell\}. \end{aligned}$$

This allows us to conclude that f_0 holds in φ.

Rule UNIFYING: Let the statement $f_1 = \Big(\mathsf{k}(M,t) \Leftarrow \mathsf{k}(X_1,x_1),\ldots,\mathsf{k}(X_k,x_k)\Big)$ and the statement $f_2 = \Big(\mathsf{k}(N,s) \Leftarrow \mathsf{k}(Y_1,y_1),\ldots,\mathsf{k}(Y_\ell,y_\ell)\Big)$ be the two solved deduction statements and $\sigma = \mathsf{mgu}(s,t)$ be the substitution involved in this step Let f_0 be the resulting equational statement:

$$f_0 = \Big(\mathsf{i}(M,N) \Leftarrow \mathsf{k}(X_1,x_1\sigma),\ldots,\mathsf{k}(X_k,x_k\sigma),\mathsf{k}(Y_1,y_1\sigma),\ldots,\mathsf{k}(Y_\ell,y_\ell\sigma)].$$

We show that f_0 holds in φ. Let τ be a substitution such that $\varphi \vdash^{M_i} x_i\sigma\tau$ for all $1 \le i \le k$ and $\varphi \vdash^{N_j} y_j\sigma\tau$ for all $1 \le j \le \ell$. Since f_1 and f_2 hold in φ and $s\sigma\tau = t\sigma\tau$, we deduce that $\varphi \vdash^{M'} t\sigma\tau$ for $M' \in \{M\{X_1 \mapsto M_1,\ldots,X_k \mapsto M_k\}, N\{Y_1 \mapsto N_1,\ldots,Y_k \mapsto N_\ell\}$. This allows us to conclude that f_0 holds in φ.

Rule E-SOLVING: Let $f_1 = \Big(\mathsf{i}(U,V) \Leftarrow \mathsf{k}(Y,s),\mathsf{k}(X_1,t_1),\ldots,\mathsf{k}(X_k,t_k)\Big)$ be the equational statement and $f_2 = \Big(\mathsf{i}(N,t) \Leftarrow \mathsf{k}(Y_1,y_1),\ldots,\mathsf{k}(Y_\ell,y_\ell)\Big)$ be the solved deduction statement, and $\sigma = \mathsf{mgu}(s,t)$ be the substitution involved in this step. Let f_0 be the resulting equational statement:

$$f_0 = \Big(\mathsf{i}(U\{Y \mapsto N\}, V\{Y \mapsto N\}) \Leftarrow \mathsf{k}(X_1,t_1\sigma),\ldots,\mathsf{k}(X_k,t_k\sigma),\mathsf{k}(Y_1,y_1\sigma),\ldots,\mathsf{k}(Y_\ell,y_\ell\sigma)\Big).$$

We show that f_0 holds in φ. Let τ be a substitution such that $\varphi \vdash^{M_i} t_i\sigma\tau$ for all $1 \leq i \leq k$ and $\varphi \vdash^{N_j} y_j\sigma\tau$ for all $1 \leq j \leq \ell$. Since f_2 holds in φ, we deduce that $\varphi \vdash^{N'} t\sigma\tau$ for $N' = N\{Y_1 \mapsto N_1, \ldots, Y_\ell \mapsto N_\ell\}$. Since $s\sigma\tau = t\sigma\tau$, we deduce that $\varphi \vdash^{N'} s\sigma\tau$, and by using the fact that f_1 holds in φ we deduce that

$$(U\{Y \mapsto N', X_1 \mapsto M_1, \ldots, X_k \mapsto M_k\} = V\{Y \mapsto N', X_1 \mapsto M_1, \ldots, X_k \mapsto M_k\})\varphi.$$

Thus, f_0 holds in φ. □

Finally, we are ready to present the main soundness lemma:

Lemma 4.3 (soundness)**.** *Let φ be a frame and K be a knowledge base such that $\mathrm{Init}(\varphi) \Longrightarrow^* K$. Let t, M, N be terms and let $K^+ = K \cup \{\Big(\mathsf{k}(n,n) \Leftarrow\Big) \mid n \in \mathcal{N}ames(t) \setminus \mathcal{N}\}$. We have that:*

1. *$\mathsf{k}(M,t) \in \mathbf{deriv}(K^+)$ implies $\varphi \vdash^M t$ and*
2. *$K \models \mathsf{i}(M,N)$ implies $(M = N)\varphi$.*

Proof. By Lemma 4.2 and because every $f \in \{\Big(\mathsf{k}(n,n) \Leftarrow\Big) \mid n \in \mathcal{N}ames(t) \setminus \mathcal{N}\}$ holds in φ, we have that all facts in K^+ hold in φ. To conclude, we show Points 1 and 2 stated in the Lemma.

1. Let M and t be such that $\mathsf{k}(M,t) \in \mathbf{deriv}(K^+)$. By definition of **deriv**, as t is ground, there exists a solved deduction statement $f_0 = \Big(\mathsf{k}(M_0,t_0) \Leftarrow \mathsf{k}(X_1,x_1), \ldots, \mathsf{k}(X_k,x_k)\Big) \in K^+$ such that $t = t_0\sigma$ for some substitution σ and $\mathsf{k}(M_i, x_i\sigma) \in \mathbf{deriv}(K^+)$ for some M_i for all $1 \leq i \leq k$ and $M = M_0\{X_1 \mapsto M_1, \ldots, X_k \mapsto M_k\}$. We show the result by induction on $|t|$.

   Base case: $|t| = 1$. In such a case t is either a name or a constant. We have that $k = 0$, $t_0 = t$ and $M = M_0$. Since f_0 holds in φ, we deduce that $\varphi \vdash^{M_0} t$. This allows us to conclude.

   Induction step. Note that $|x_i\sigma| < |t|$ and $\mathsf{k}(M_i, x_i\sigma) \in \mathbf{deriv}(K^+)$, thus we can apply our induction hypothesis on $x_i\sigma$. We deduce that $\varphi \vdash^{M_i} x_i\sigma$ and thus $\varphi \vdash^M t_0\sigma = t$ since f_0 holds in φ.

2. Let M and N be such that $K \models \mathsf{i}(M,N)$. To show that $(M = N)\varphi$, it is sufficient to establish that

   $$(M'\sigma = N'\sigma)\varphi \quad \text{where } \sigma = \{X_1 \mapsto x_1, \ldots, X_k \mapsto x_k\}$$

   for every solved equational fact $\Big(\mathsf{i}(M',N') \Leftarrow \mathsf{k}(X_1,x_1), \ldots, \mathsf{k}(X_k,x_k)\Big) \in K$. This follows easily from Lemma 4.2.

□

4.4.2 Completeness

Let us now prove the completeness of the saturation procedure. First we show completeness with respect to equational statements.

Lemma 4.4. *Let K be a saturated knowledge base and $f = \Big(\mathsf{i}(U,V) \Leftarrow \mathsf{k}(X_1,t_1), \ldots, \mathsf{k}(X_k,t_k)\Big)$ be an equational statement in K. For any substitution σ grounding for $\{t_1, \ldots, t_k\}$ such that $\mathsf{k}(R_i', t_i\sigma) \in \mathbf{deriv}(K|_{\mathsf{solved}})$ for some R_i' for all $1 \leq i \leq k$, we have that $\mathsf{k}(R_i, t_i\sigma) \in \mathbf{deriv}(K|_{\mathsf{solved}})$ for some R_i for all $1 \leq i \leq k$ and $K|_{\mathsf{solved}} \models \mathsf{i}(U\tau, V\tau)$ where $\tau = \{X_1 \mapsto R_1, \ldots, X_k \mapsto R_k\}$.*

Proof. We show this result by induction on $\sum_{i=1}^k |t_i\sigma|$. We distinguish two cases:

1. *f is a solved equational statement*, i.e. $t_1, \ldots, t_k$ are variables (not necessarily distinct), say $x_1, \ldots, x_k$. In such a case, we have that

$$K|_{\mathsf{solved}} \models \mathsf{i}(U\{X_1 \mapsto x_1, \ldots, X_k \mapsto x_k\}, V\{X_1 \mapsto x_1, \ldots, X_k \mapsto x_k\}).$$

   We choose each R_i arbitrarily such that $x_i = x_j$ implies $R_i = R_j$. Then, it is easy to conclude.

2. *f is an unsolved equational statement.* In such a case, there exists t_j such that $t_j \notin \mathcal{X}$. Let us assume w.l.o.g. that $j = 1$. As $\mathsf{k}(R'_1, t_1\sigma) \in \mathbf{deriv}(K|_{\mathsf{solved}})$, we know that there exist a solved deduction statement $f^1 = \Big(\mathsf{k}(R^1, t^1) \Leftarrow \mathsf{k}(X^1_1, x^1_1), \ldots, \mathsf{k}(X^1_\ell, x^1_\ell)\Big)$ in K and a substitution τ such that $t^1\tau = t_1\sigma$ and $\mathsf{k}(R'_i, x^1_i\tau) \in K|_{\mathsf{solved}}$ for all $1 \leq i \leq \ell$.

   Let $\rho = \mathsf{mgu}(t_1, t^1)$. Since K is saturated, by rule E-Solving , we have that the following statement f_2 is in K:

$$\Big(\mathsf{i}(U\{X_1 \mapsto R^1\}, V\{X_1 \mapsto R^1\}) \Leftarrow \mathsf{k}(X^1_1, x^1_1\rho), \ldots, \mathsf{k}(X^1_\ell, x^1_\ell\rho), \mathsf{k}(X_2, t_2\rho), \ldots, \mathsf{k}(X_k, t_k\rho)\Big).$$

   Let σ' be the substitution such that $\sigma \cup \tau = \rho\sigma'$. As the deduction statement f^1 is solved, $x^1_1\rho\sigma', \ldots, x^1_\ell\rho\sigma'$ are strict subterms of $t^1\rho\sigma' = t^1\tau$ and $\sum_{i=1}^{\ell} |x^1_i\rho\sigma'| < |t^1\tau| = |t_1\sigma|$. Thus we can apply our induction hypothesis on the equational statement f_2 with the substitution σ'. This allows us to obtain that there exist $M^1_1, \ldots, M^1_\ell, M_2, \ldots, M_k$ such that $\mathsf{k}(M_i, t_i\rho\sigma' = t_i\sigma) \in \mathbf{deriv}(K|_{\mathsf{solved}})$ for all $2 \leq i \leq k$ and $\mathsf{k}(M^1_i, x^1_i\rho\sigma = x^1\sigma)$ for all $1 \leq i \leq \ell$ and the following equation ($\star$)

$$\begin{array}{rcl} K|_{\mathsf{solved}} & \models & (\mathsf{i}(U\{X_1 \mapsto R^1\})\{X^1_1 \mapsto M^1_1, \ldots, X^1_\ell \mapsto M^1_\ell, X_2 \mapsto M_2, \ldots, X_k \mapsto M_k\}, \\ & & (V\{X_1 \mapsto R^1\})\{X^1_1 \mapsto M^1_1, \ldots, X^1_\ell \mapsto M^1_\ell, X_2 \mapsto M_2, \ldots, X_k \mapsto M_k\}) \end{array}$$

   We choose $R_1 = R^1\{X^1_1 \mapsto M^1_1, \ldots, X^1_\ell \mapsto M^1_\ell\}$ and $R_2 = M_2, \ldots, R_k = M_k$. Thus, the equation ($\star$) can be rewritten as follows:

$$K|_{\mathsf{solved}} \models \mathsf{i}(U\{X_1 \mapsto R_1, \ldots, X_k \mapsto R_k\}, V\{X_1 \mapsto R_1, \ldots, X_k \mapsto R_k\}).$$

   This allows us to conclude.

□

Now we show that if the same term can be derived with two recipes then the equality between the two statements is a consequence of the solved knowledge base.

Proposition 4.1 (completeness, equation)**.** *Let K be a saturated knowledge base, and M, N be two terms such that $\mathsf{k}(M, t) \in \mathbf{deriv}(K|_{\mathsf{solved}})$ and $\mathsf{k}(N, t) \in \mathbf{deriv}(K|_{\mathsf{solved}})$ for some ground term t. Then, we have that $K|_{\mathsf{solved}} \models \mathsf{i}(M, N)$.*

Proof. By definition of $\mathsf{k}(M, t) \in \mathbf{deriv}(K|_{\mathsf{solved}})$ we know that there exist a substitution σ_1 and a deduction statement $f_1 = \Big(\mathsf{k}(M_0, u_0) \Leftarrow \mathsf{k}(X_1, x_1), \ldots, \mathsf{k}(X_k, x_k)\Big)$ in K such that $u_0\sigma_1 = t$, $\mathsf{k}(M_i, x_i\sigma_1) \in \mathbf{deriv}(K|_{\mathsf{solved}})$ for all $1 \leq i \leq k$ and $M_0\{X_i \mapsto M_i\}_{1 \leq i \leq k} = M$.

Similarly, by definition of $\mathsf{k}(N, t) \in \mathbf{deriv}(K|_{\mathsf{solved}})$ we know that there exist a substitution σ_2 and a deduction statement $f_2 = \Big(\mathsf{k}(N_0, v_0) \Leftarrow \mathsf{k}(Y_1, y_1), \ldots, \mathsf{k}(Y_\ell, y_\ell)\Big)$ in K such that $v_0\sigma_2 = t$, $\mathsf{k}(N_j, y_j\sigma_2) \in \mathbf{deriv}(K|_{\mathsf{solved}})$ for all $1 \leq j \leq \ell$ and $N_0\{Y_j \mapsto N_j\}_{1 \leq i \leq \ell} = N$.

We prove the result by induction on $|t|$. As our knowledge base K is saturated, rule Unifying must have been applied to the statements f_1 and f_2. Therefore, we have that there exists an equational statement $f_3 \in K$ such that:

$$f_3 = \Big(\mathsf{i}(M_0, N_0) \Leftarrow \mathsf{k}(X_1, x_1\sigma), \ldots, \mathsf{k}(X_k, x_k\sigma), \mathsf{k}(Y_1, y_1\sigma), \ldots, \mathsf{k}(Y_\ell, y_\ell\sigma)\Big).$$

where $\sigma = mgu(u_0, v_0)$.

Let σ' be a substitution such that $\sigma_1 \cup \sigma_2 = \sigma\sigma'$. We can now apply Lemma 4.4 on f_3 with substitution σ'. We obtain that there exist $R_1, \ldots, R_k$ and $W_1, \ldots, W_\ell$ such that $\mathsf{k}(R_i, x_i\sigma\sigma') \in \mathbf{deriv}(K|_{\mathsf{solved}})$ for all $1 \leq i \leq k$ and $\mathsf{k}(W_j, y_j\sigma\sigma') \in \mathbf{deriv}(K|_{\mathsf{solved}})$ for all $1 \leq j \leq \ell$ and such that

$$K|_{\mathsf{solved}} \models \mathsf{i}(M_0\delta, N_0\delta) \tag{4.1}$$

where $\delta = \{X_1 \mapsto R_1, \ldots, X_k \mapsto R_k, Y_1 \mapsto W_1, \ldots, Y_\ell \mapsto W_\ell\}$.

As M_i and R_i $(1 \leq i \leq k)$ are such that $\mathsf{k}(M_i, x_i\sigma_1) \in \mathbf{deriv}(K|_{\mathsf{solved}})$ and $\mathsf{k}(R_i, x_i\sigma\sigma') \in \mathbf{deriv}(K|_{\mathsf{solved}})$, and as $x_1\sigma\sigma' = x_1\sigma_1$ is a strict subterm of $u_0\sigma_1 = t$, we can apply the induction hypothesis to obtain that $K|_{\mathsf{solved}} \models \mathsf{i}(M_i, R_i)$. In a similar way, we also deduce that $K|_{\mathsf{solved}} \models \mathsf{i}(N_j, W_j)$ for all $1 \leq j \leq \ell$. By replacing W_j by M_j and R_i by N_i in equation (4.1), we obtain our conclusion. □

We now need a helper lemma that shows that if a derivable term is instantiated with derivable terms, the result is still derivable.

Lemma 4.5. *Let K be a knowledge base and t be a term. Let σ be a grounding substitution for t. If $\mathsf{k}(W, t) \in \mathbf{deriv}(K)$ and $\mathsf{k}(R_x, x\sigma) \in \mathbf{deriv}(K)$ for every $x \in \mathcal{V}ar(t)$, then $\mathsf{k}(W', t\sigma) \in \mathbf{deriv}(K)$ where $W' = W\{x \mapsto R_x\}_{x \in \mathcal{V}ar(t)}$.*

Proof. We show this result by induction on $|t|$.
Base case: if $|t| = 1$.

- If $t = x$ is a variable, as $\mathsf{k}(W, t) \in \mathbf{deriv}(K)$, it follows that $W = t = x$. By hypothesis, there exists R_x such that $\mathsf{k}(R_x, x\sigma) \in \mathbf{deriv}(K)$. This allows us to conclude.
- If t is a name or a constant, then $W' = W$ and $t\sigma = t$ and we immediately conclude.

Induction case: $|t| > 1$.

As $\mathsf{k}(W, t) \in \mathbf{deriv}(K)$, it follows that there exist a statement $f \in K$ and a substitution τ such that:

- $f = \Big(\mathsf{k}(R, u) \Leftarrow \mathsf{k}(X_1, x_1), \ldots, \mathsf{k}(X_k, x_k)\Big)$;
- $t = u\tau$;
- $\mathsf{k}(R_i, x_i\tau) \in \mathbf{deriv}(K)$ for every $1 \leq i \leq k$ and $W = R\{X_1 \mapsto R_1, \ldots, X_k \mapsto R_k\}$.

We have that $\mathcal{V}ar(u) = \{x_1, \ldots, x_k\}$ and thus, $x_i\tau$ is a strict subterm of $u\tau$ $(1 \leq i \leq k)$. Therefore, we can apply our induction hypothesis on each term $x_i\tau$ with the substitution σ. For each i such that $1 \leq i \leq k$, we obtain that:

$$\mathsf{k}(W_i, x_i\tau\sigma) \in \mathbf{deriv}(K) \text{ where } W_i = R_i\{x \mapsto R_x\}_{x \in \mathcal{V}ar(x_i\tau)}.$$

Note that since $t = u\tau$ and $\mathcal{V}ar(u) = \{x_1, \ldots, x_k\}$, we have that $\mathcal{V}ar(t) = \mathcal{V}ar(\{x_1\tau, \ldots, x_k\tau\})$. By using the statement f, we get that $\mathsf{k}(W'', u\tau\sigma) \in \mathbf{deriv}(K)$ where

$$\begin{array}{rcl} W'' & = & R\{X_1 \mapsto R_1\{x \mapsto R_x\}_{x \in \mathcal{V}ar(t)}, \ldots, X_k \mapsto R_k\{x \mapsto R_x\}_{x \in \mathcal{V}ar(t)}\} \\ & = & (R\{X_1 \mapsto R_1, \ldots, X_k \mapsto R_k\})\{x \mapsto R_x\}_{x \in \mathcal{V}ar(t)} \\ & = & W\{x \mapsto R_x\}_{x \in \mathcal{V}ar(t)} \end{array}$$

Let $W' = W\{x \mapsto R_x\}_{x \in \mathcal{V}ar(t)}$. We have that $\mathsf{k}(W', u\tau\sigma) \in \mathbf{deriv}(K)$ and since $u\tau\sigma = t\sigma$ we easily conclude. □

We now show the completeness of the $\oplus$ operator with respect to solved statements.

Lemma 4.6. *Let* $f = \Big(\mathsf{k}(R,t) \Leftarrow \mathsf{k}(X_1,x_1),\ldots,\mathsf{k}(X_k,x_k)\Big)$ *be a solved statement and* K *be a knowledge base such that* $K \oplus f = K$*. Let* σ *be a substitution grounding for* $\{x_1,\ldots,x_k\}$ *such that* $\mathsf{k}(T_i',x_i\sigma) \in \mathbf{deriv}(K|_{\mathsf{solved}})$ *for all* $1 \leq i \leq k$*. Then there exist* W *and* R_i *for all* $1 \leq i \leq k$ *such that:*

- $\mathsf{k}(W,t\sigma) \in \mathbf{deriv}(K|_{\mathsf{solved}})$*, and* $\mathsf{k}(R_i,x_i\sigma) \in \mathbf{deriv}(K|_{\mathsf{solved}})$ *for every* $1 \leq i \leq k$*;*
- $K|_{\mathsf{solved}} \models \mathsf{i}(W, R\{X_1 \mapsto R_1,\ldots,X_k \mapsto R_k\})$.

Proof. Let $f' = f{\Downarrow}$ be the canonical form of f. We first show that $K \cup \{f'\} = K$ implies that there exists T' such that $\mathsf{k}(T',t) \in \mathbf{deriv}(K|_{\mathsf{solved}})$. This is easily shown by induction on the number of steps to compute the canonical form.

Base case: If f is already in canonical form we have that $f = f'$ and hence there exists T' such that $\mathsf{k}(T',t) \in \mathbf{deriv}(K|_{\mathsf{solved}})$.
Inductive case: The two rules are of the form

$$\frac{\Big(\mathsf{k}(R,t) \Leftarrow \mathsf{k}(X_1,x_1),\ldots,\mathsf{k}(X_k,x_k)\Big)}{f_0 = \Big(\mathsf{k}(R',t) \Leftarrow \mathsf{k}(X_1,x_1),\ldots,\mathsf{k}(X_{i-1},x_{i-1}),\mathsf{k}(X_{i+1},x_{i+1}),\ldots,\mathsf{k}(X_k,x_k)\Big)}$$

Let f_0' be the canonical form of f_0. By the induction hypothesis we have $K \cup \{f_0'\} = K$ implies that there exist T' such that $\mathsf{k}(T',t) \in \mathbf{deriv}(K|_{\mathsf{solved}})$. As $f' = f_0'$ we conclude.

To prove the lemma we consider both cases where f is either useful or redundant.

Useful statement: If f is useful we have that $\mathsf{k}(T',t) \in \mathbf{deriv}(K|_{\mathsf{solved}})$ for some T'. By usefulness, $K \cup \{f{\Downarrow}\} \neq K$ which contradicts that $K \oplus f = K$. Hence, this case is impossible.

Redundant statement: Since $K \oplus f = K$, it follows that there exists W' such that $\mathsf{k}(W',t) \in \mathbf{deriv}(K|_{\mathsf{solved}})$ and $K \models \mathsf{i}(W', R\{X_1 \mapsto x_1,\ldots,X_k \mapsto x_k\})$.We choose R_i arbitrarily such that $\mathsf{k}(R_i,x_i\sigma) \in \mathbf{deriv}(K|_{\mathsf{solved}})$. Let $W'' = W'\{x_1 \mapsto R_1,\ldots,x_k \mapsto R_k\}$. By Lemma 4.5, we deduce that $\mathsf{k}(W'',t\sigma) \in \mathbf{deriv}(K|_{\mathsf{solved}})$ and we also have that

$$K|_{\mathsf{solved}} \models \mathsf{i}(W', (R\{X_1 \mapsto x_1,\ldots,X_k \mapsto x_k\})\{x_1 \mapsto R_1,\ldots,x_k \mapsto R_k\})$$

i.e. $K|_{\mathsf{solved}} \models \mathsf{i}(W'', R\{X_1 \mapsto R_1,\ldots,X_k \mapsto R_k\})$.

Let $W = W''$. We have that $\mathsf{k}(W,t\sigma) \in \mathbf{deriv}(K|_{\mathsf{solved}})$, and $\mathsf{k}(R_i,x_i\sigma) \in \mathbf{deriv}(K|_{\mathsf{solved}})$ for every $1 \leq i \leq k$. Lastly, we have that $K \models \mathsf{i}(W, R\{X_1 \mapsto R_1,\ldots,X_k \mapsto R_k\})$.

□

We now show the completeness of the $\oplus$ operator with respect to any type of deduction statements.

Lemma 4.7. *Let* K *be a saturated knowledge base. Let* $f = \Big(\mathsf{k}(R,t) \Leftarrow \mathsf{k}(X_1,t_1),\ldots,\mathsf{k}(X_k,t_k)\Big)$ *be a deduction statement such that* $K \oplus f = K$*. For any substitution* σ *grounding for* $\{t_1,\ldots,t_k\}$ *such that* $\mathsf{k}(T_i',t_i\sigma) \in \mathbf{deriv}(K|_{\mathsf{solved}})$ *for all* $1 \leq i \leq k$*, we have that there exist* $R_1,\ldots,R_k$ *and* W *such that*

- $\mathsf{k}(W,t\sigma) \in \mathbf{deriv}(K|_{\mathsf{solved}})$*, and* $\mathsf{k}(R_i,t_i\sigma) \in \mathbf{deriv}(K|_{\mathsf{solved}})$ *for all* $1 \leq i \leq k$*;*
- $K|_{\mathsf{solved}} \models \mathsf{i}(W, R\{X_1 \mapsto R_1,\ldots,X_k \mapsto R_k\})$.

Proof. We show the result by induction on $\sum_{i=1}^{k} |t_i\sigma|$. We distinguish two cases. If f is solved then we easily conclude by applying Lemma 4.6.

If f is not solved, there exists j such that $t_j \notin \mathcal{X}$. We assume w.l.o.g. that $j = 1$. Since $\mathsf{k}(T_1',t_1\sigma) \in \mathbf{deriv}(K|_{\mathsf{solved}})$, there exist a solved deduction statement $f' \in K$, some terms $R_i'(1 \leq i \leq \ell)$ and a substitution τ such that:

- $f' = \Big(\mathsf{k}(R', t') \Leftarrow \mathsf{k}(Y_1, y_1), \ldots, \mathsf{k}(Y_\ell, y_\ell)\Big)$;
- $t'\tau = t_1\sigma$;
- $\mathsf{k}(R'_i, y_i\tau) \in \mathbf{deriv}(K|_{\mathsf{solved}})$ for every $1 \leq i \leq \ell$.

By application of the F-SOLVING rule to the deduction statements f and f', we obtain the following statement

$$f_0 = \Big(\mathsf{k}(R\{X_1 \mapsto R'\}, t\rho) \Leftarrow \mathsf{k}(X_2, t_2\rho), \ldots, \mathsf{k}(X_k, t_k\rho), \mathsf{k}(Y_1, y_1\rho), \ldots, \mathsf{k}(Y_\ell, y_\ell\rho)\Big)$$

where $\rho = mgu(t', t_1)$.

As K is saturated, $K \oplus f_0 = K$. Let σ' be the substitution such that $\sigma \cup \tau = \rho\sigma'$. As $y_i\rho\sigma' = y_i(\sigma \cup \tau) = y_i\tau$ are strict disjoint subterms of $t'\tau = t_1\sigma$, it follows that we can apply our induction hypothesis on f_0 and the substitution σ'. Therefore, there exist $R'_2, \ldots, R'_k, R^y_1, \ldots, R^y_\ell$ and W' such that:

- $\mathsf{k}(W', t\rho\sigma') \in \mathbf{deriv}(K|_{\mathsf{solved}})$,
- $\mathsf{k}(R'_i, t_i\rho\sigma') \in \mathbf{deriv}(K|_{\mathsf{solved}})$ for every $2 \leq i \leq k$,
- $\mathsf{k}(R^y_j, y_j\rho\sigma') \in \mathbf{deriv}(K|_{\mathsf{solved}})$ for every $1 \leq j \leq \ell$,
- $K|_{\mathsf{solved}} \models \mathsf{i}(W', (R\{X_1 \mapsto R'\})\{X_2 \mapsto R'_2, \ldots, X_k \mapsto R'_k, Y_1 \mapsto R^y_1, \ldots, Y_\ell \mapsto R^y_\ell\})$.

Let $W = W'$, $R_1 = R'\{Y_1 \mapsto R^y_1, \ldots, Y_\ell \mapsto R^y_\ell\}$, $R_j = R'_j$ for every $2 \leq j \leq k$. It immediately follows that $K \models \mathsf{i}(W, R\{X_1 \mapsto R_1, \ldots, X_k \mapsto R_k\})$, $\mathsf{k}(W, t\sigma) \in \mathbf{deriv}(K|_{\mathsf{solved}})$ and $\mathsf{k}(R_i, t_i\sigma) \in \mathbf{deriv}(K|_{\mathsf{solved}})$ for $1 \leq i \leq k$. This allows us to conclude. □

We now show that the saturation process is complete with respect to the rewrite system.

Proposition 4.2 (completeness, reduction)**.** *Let K be a saturated knowledge base, M a term and t a ground term such that $\mathsf{k}(M, t) \in \mathbf{deriv}(K|_{\mathsf{solved}})$ and $t{\downarrow} \neq t$. Then there exist M' and t' such that $\mathsf{k}(M', t') \in \mathbf{deriv}(K|_{\mathsf{solved}})$ with $t \to^+ t'$ and $K|_{\mathsf{solved}} \models \mathsf{i}(M, M')$.*

Proof. We show this result by induction on $|t|$. By definition of $\mathsf{k}(M, t) \in \mathbf{deriv}(K|_{\mathsf{solved}})$, we know that there exist $f_0 = \Big(\mathsf{k}(M_0, u_0) \Leftarrow \mathsf{k}(X_1, x_1), \ldots, \mathsf{k}(X_k, x_k)\Big)$ in K and a substitution σ such that $u_0\sigma = t$ and $\mathsf{k}(M_i, x_i\sigma) \in \mathbf{deriv}(K|_{\mathsf{solved}})$ for all $1 \leq i \leq k$ and $M_0\{X_i \mapsto M_i\}_{1 \leq i \leq k} = M$ for some M_i $(1 \leq i \leq k)$. We distinguish two cases:

Case 1: there exists $1 \leq j \leq k$ such that $x_j\sigma{\downarrow} \neq x_j\sigma$. Let us assume w.l.o.g. that $j = 1$. Since $x_1\sigma$ is a strict subterm of t, we can apply our induction hypothesis on $x_1\sigma$. We obtain that there exist M'_1 and u'_1 such that $\mathsf{k}(M'_1, u'_1) \in \mathbf{deriv}(K|_{\mathsf{solved}})$ with $x_1\sigma \to^+ u'_1$ and $K|_{\mathsf{solved}} \models \mathsf{i}(M_1, M'_1)$. Now, let σ' be the substitution defined as follows:

$$x\sigma' = \begin{cases} x\sigma \text{ for } x \neq x_1 \\ u'_1 \text{ otherwise} \end{cases}$$

Let $t' = u_0\sigma'$ and $M' = M_0\{X_1 \mapsto M'_1, X_2 \mapsto M_2, \ldots, X_k \mapsto M_k\}$. Since $x_1 \in \mathcal{V}ar(u_0)$, it is easy to see that $t = u_0\sigma \to^+ u_0\sigma' = t'$. Furthermore, it is also easy to see that $\mathsf{k}(M', t') \in \mathbf{deriv}(K|_{\mathsf{solved}})$. Lastly, since $K|_{\mathsf{solved}} \models \mathsf{i}(M_1, M'_1)$, we have that $K|_{\mathsf{solved}} \models \mathsf{i}(M, M')$.

Case 2: $x_j\sigma{\downarrow} = x_j\sigma$ for every $1 \leq j \leq k$. In such a case, we have that $u_0 = C[u'_0]$ for some context C and some term $u'_0 \notin \mathcal{X}$ such that $u'_0\sigma = l\tau$ where $l \to r \in \mathsf{R}$ and τ is a substitution. As the knowledge base K is saturated, the rule NARROWING must have been applied. Therefore there exists f_1 such that:

- $K \oplus f_1 = K$, and

- $f_1 = \Big(\mathsf{k}(M_0, (C[r])\rho) \Leftarrow \mathsf{k}(X_1, x_1\rho), \ldots, \mathsf{k}(X_k, x_k\rho)\Big)$

where $\rho = \mathsf{mgu}(u_0', l)$. Let ρ' be the substitution with $\mathcal{D}om(\rho') = \mathcal{V}ar(\{x_1\rho, \ldots, x_k\rho\})$ and $\sigma \cup \tau = \rho\rho'$. Now, we apply Lemma 4.7 on the statement f_1 and the substitution ρ'. We deduce that there exist $R_1, \ldots, R_k$ and W such that

- $\mathsf{k}(W, (C[r])\rho\rho') \in \mathbf{deriv}(K|_{\mathsf{solved}})$, and $\mathsf{k}(R_i, x_i\rho\rho') \in \mathbf{deriv}(K|_{\mathsf{solved}})$ for $1 \leq i \leq k$; and
- $K|_{\mathsf{solved}} \models \mathsf{i}(W, M_0\{X_1 \mapsto R_1, \ldots, X_k \mapsto R_k\})$.

Let $t' = (C[r])\rho\rho'$ and $M' = W$. We have that $\mathsf{k}(M', t') \in \mathbf{deriv}(K|_{\mathsf{solved}})$. Moreover, since $\mathsf{k}(R_i, x_i\rho\rho') \in \mathbf{deriv}(K|_{\mathsf{solved}})$, $\mathsf{k}(M_i, x_i\sigma) \in \mathbf{deriv}(K|_{\mathsf{solved}})$ and $x_i\rho\rho' = x_i\sigma$, we can apply Lemma 4.1 in order to deduce that $K|_{\mathsf{solved}} \models \mathsf{i}(R_1, M_i)$ for $1 \leq i \leq k$. Thus, we have that $K|_{\mathsf{solved}} \models \mathsf{i}(M, M')$. In order to conclude, it remains to show that $t \to_{\mathsf{R_E}}^+ t'$. Indeed, we have that $t = u_0\sigma = (C[u_0'])\sigma \to_{\mathsf{R_E}}^+ (C[r])\rho\rho' = t'$. □

The soundness and completeness of the saturation procedure is given by the following theorem:

Theorem 4.1. *Let φ be a frame and K be a saturated knowledge base such that $\mathrm{Init}(\varphi) \Longrightarrow^* K$. Let $t \in T(\mathcal{F})$ and let $K^+ = K \cup \{\Big(\mathsf{k}(n, n) \Leftarrow\Big) \mid n \in \mathcal{N}ames(t) \setminus \mathcal{N}\}$. Then*

1. *for all terms M we have that:*

$$\varphi \vdash^M t \text{ iff there exists } N \text{ such that } K|_{\mathsf{solved}} \models \mathsf{i}(M, N) \text{ and } \mathsf{k}(N, t{\downarrow}) \in \mathbf{deriv}(K^+|_{\mathsf{solved}})$$

2. *and for all terms M, N we have that*

$$(M = N)\varphi \text{ iff } K|_{\mathsf{solved}} \models \mathsf{i}(M, N).$$

Proof. Let φ be a frame and K be a saturated knowledge base such that $\mathrm{Init}(\varphi) \Longrightarrow^* K$.

1.($\Leftarrow$) Let M, N and t be such that $K|_{\mathsf{solved}} \models \mathsf{i}(M, N)$ and $\mathsf{k}(N, t{\downarrow}) \in \mathbf{deriv}(K^+|_{\mathsf{solved}})$. By Lemma 4.3, we have that $M\varphi =_{\mathsf{R}} N\varphi =_{\mathsf{R}} t$.

($\Rightarrow$) Let M and t be such that $\varphi \vdash^M t$.

Let $K^{++} = K \cup \{\Big(\mathsf{k}(n, n) \Leftarrow\Big) \mid n \in \mathcal{N}ames(M)\}$. We have that $\mathsf{k}(M, t_0) \in \mathbf{deriv}(K^{++}|_{\mathsf{solved}})$ and $t_0 \to^* t{\downarrow}$ with $t_0 = M\varphi$.

Let $\{n_1, \ldots, n_\ell\} = \mathcal{N}ames(M) \setminus \mathcal{N}ames(\varphi, t)$. Let $y_1, \ldots, y_\ell$ be fresh variables and $\delta = \{n_1 \mapsto y_1, \ldots, n_\ell \mapsto y_\ell\}$. Let $M' = M\delta$. We have that $\mathsf{k}(M', t_0') \in \mathbf{deriv}(K^{++}|_{\mathsf{solved}})$ and $t_0' \to^* t{\downarrow}$ with $t_0' = M'\varphi$.

Now, let $K^{+++} = K^{++} \cup \{\Big(\mathsf{i}(n, n) \Leftarrow\Big) \mid n \in \mathcal{N}ames(M)\}$. As K is a saturated knowledge base, we have that K^{+++} is a saturated knowledge base as well. By Proposition 4.1, we deduce that $K|_{\mathsf{solved}}^{+++} \models \mathsf{i}(M, M')$, thus $K|_{\mathsf{solved}} \models \mathsf{i}(M, M')$ as well.

We show the result by induction on t_0 equipped with the order $<$ induced by the rewriting relation ($t < t'$ if and only if $t' \to^+ t$).

Base case: $\mathsf{k}(M', t_0 = t{\downarrow}) \in \mathbf{deriv}(K^+|_{\mathsf{solved}})$ Let $N = M'$, we have $K|_{\mathsf{solved}} \models \mathsf{i}(M, N)$ and $\mathsf{k}(N, t{\downarrow}) \in \mathbf{deriv}(K|_{\mathsf{solved}})$.

Induction case: $\mathsf{k}(M', t_0) \in \mathbf{deriv}(K^+|_{\mathsf{solved}})$ *with* $t_0 \neq t{\downarrow}$. Let $K^+ = K \cup \{\Big(\mathsf{i}(n, n) \Leftarrow\Big) \mid n \in \mathcal{N}ames(t) \setminus \mathcal{N}\}$. We easily see that as K is a saturated knowledge base we have that K^+ is a saturated knowledge base as well. Hence we can apply Proposition 4.2 and deduce that there exist N' and t' such that $\mathsf{k}(N', t') \in \mathbf{deriv}(K^+|_{\mathsf{solved}})$, $t \to^+ t'$, and $K|_{\mathsf{solved}}^+ \models \mathsf{i}(M', N')$. It is easy to see that $K|_{\mathsf{solved}} \models \mathsf{i}(M', N')$ as well. We have that $\mathsf{k}(N', t') \in \mathbf{deriv}(K^+|_{\mathsf{solved}})$, $t' \to^* t{\downarrow}$ and $t' < t_0$. Thus, we can apply our induction hypothesis and we obtain that there exists N such that $K|_{\mathsf{solved}} \models \mathsf{i}(N', N)$ and $\mathsf{k}(N, t{\downarrow}) \in \mathbf{deriv}(K^+|_{\mathsf{solved}})$.

2.($\Leftarrow$) By Lemma 4.3, $K|_{\mathsf{solved}} \models \mathsf{i}(M, N)$ implies $(M = N)\varphi$.

($\Rightarrow$) Let M and N be terms such that $(M = N)\varphi$. This means that there exists t such that $\varphi \vdash^M t$ and $\varphi \vdash^R t$. Let $K^+ = K \cup \{\Big(\mathsf{k}(n, n) \Leftarrow\Big) \mid n \in \mathcal{N}ames(t) \setminus \mathcal{N}\}$ and $K^{++} = K^+ \cup \{\Big(\mathsf{i}(n, n) \Leftarrow\Big) \mid n \in \mathcal{N}ames(t) \setminus \mathcal{N}\}$. By applying Item 1, we deduce that there exist M', N' such that $K|_{\mathsf{solved}} \models \mathsf{i}(M, M')$, $\mathsf{k}(M', t\downarrow) \in \mathbf{deriv}(K^+|_{\mathsf{solved}})$, $K|_{\mathsf{solved}} \models \mathsf{i}(N, N')$ and $\mathsf{k}(N', t\downarrow) \in \mathbf{deriv}(K^+|_{\mathsf{solved}})$. It is easy to see that $K^{++}|_{\mathsf{solved}} \models \mathsf{i}(M, M')$ and $K^{++}|_{\mathsf{solved}} \models \mathsf{i}(N, N')$ as well. Because K^{++} is a saturated knowledge base we apply Proposition 4.1 and deduce that $K^{++}|_{\mathsf{solved}} \models \mathsf{i}(M', N')$, and thus $K^{++}|_{\mathsf{solved}} \models \mathsf{i}(M, N)$, which easily implies $K|_{\mathsf{solved}} \models \mathsf{i}(M, N)$. □

4.5 Application to Deduction and Static Equivalence

We now show how to use the saturation procedure above in order to check if a term is deducible from a frame (and provide an appropriate recipe) and respectively if two frames are statically equivalent.

Procedure for deduction.

Let φ be a frame and t be a ground term. The procedure for checking if $\varphi \vdash^R t$ for some R runs as follows:

1. Apply the saturation rules to obtain (if any) a saturated knowledge base K such that $\text{Init}(\varphi) \Longrightarrow^* K$. Let $K^+ = K \cup \{\Big(\mathsf{k}(n, n) \Leftarrow\Big) \mid n \in \mathcal{N}ames(t) \setminus \mathcal{N}\}$.

2. Return *yes* if there exists R such that $\mathsf{k}(R, t\downarrow) \in \mathbf{deriv}(K^+|_{\mathsf{solved}})$ (in this case R is a recipe for t in the frame φ); otherwise return *no* (in this case t is not deducible from φ).

Proof. If the algorithm returns *yes*, there exists R such that $\mathsf{k}(R, t\downarrow) \in \mathbf{deriv}(K^+|_{\mathsf{solved}})$. As $K \models \mathsf{i}(R, R)$, by Theorem 4.1 we have that $\varphi \vdash^R t$.

Conversely, if t is deducible from φ, then there exists R such that $\varphi \vdash^R t$. By Theorem 4.1, there exists N such that $\mathsf{k}(N, t\downarrow) \in \mathbf{deriv}(K^+|_{\mathsf{solved}})$. Hence, the algorithm returns *yes*. □

Example 4.9. ex:algodeduc We continue our running example. Let K be the knowledge base obtained from $\text{Init}(\varphi_2)$ described in Example 4.8. We show that $\varphi_2 \vdash \text{enc}(c, k)$ and $\varphi_2 \vdash b$. Indeed we have that

$$\mathsf{k}(\text{mal}(w_1, c), \text{enc}(c, k)) \in \mathbf{deriv}(K|_{\mathsf{solved}} \cup \{\Big(\mathsf{k}(c, c) \Leftarrow\Big)\})$$

using statements f_9 and $\Big(\mathsf{k}(c, c) \Leftarrow\Big)$, and $\mathsf{k}(b, b) \in K|_{\mathsf{solved}}$ using statement f_2.

Procedure for static equivalence. Let φ_1 and φ_2 be two frames. The procedure for checking $\varphi_1 \approx \varphi_2$ runs as follows:

1. Apply the transformation rules to obtain (if possible) two saturated knowledge bases K_i ($i \in \{1, 2\}$) such that $\text{Init}(\varphi_i) \Longrightarrow^* K_i$ for $1 \leq i \leq 2$. Let L_i denote the solved statements of K_i ($1 \leq i \leq 2$).

2. For $\{i, j\} = \{1, 2\}$, for every solved statement $\Big(\mathsf{i}(M, N) \Leftarrow \mathsf{k}(X_1, x_1), \ldots, \mathsf{k}(X_k, x_k)\Big)$ in L_i, check if $(M\sigma = N\sigma)\varphi_j$ where $\sigma = \{X_1 \mapsto x_1, \ldots, X_k \mapsto x_k\}$.

3. If so return *yes*; otherwise return *no*.

Proof. If the algorithm returns *yes*, this means that ($\star$): for every solved equational fact $\Big(\mathsf{i}(M,N) \Leftarrow \mathsf{k}(X_1,x_1),\ldots,\mathsf{k}(X_k,x_k)\Big)$ in L_1, we have that:

$$(M\sigma = N\sigma)\varphi_2$$

where $\sigma = \{X_1 \mapsto x_1,\ldots,X_k \mapsto x_k\}$. Let M,N be terms such that $(M=N)\varphi_1$. By Theorem 4.1, we have that $L_1 \models \mathsf{i}(M,N)$. By ($\star$), we deduce that $(M=N)\varphi_2$. The other direction is analogous.

Conversely, assume now that $\varphi_1 \approx \varphi_2$. Let $\Big(\mathsf{i}(M,N) \Leftarrow \mathsf{k}(X_1,x_1),\ldots,\mathsf{k}(X_k,x_k)\Big)$ be a solved equational fact in L_1 and let us show that $(\tilde{M}=\tilde{N})\varphi_2$ where

- $\tilde{M} = M\{X_1 \mapsto x_1,\ldots,X_k \mapsto x_k\}$, and
- $\tilde{N} = N\{X_1 \mapsto x_1,\ldots,X_k \mapsto x_k\}$.

(The other case is analogous, and we will conclude that the algorithm returns *yes*.)

Let $\{y_1,\ldots,y_\ell\} = \mathcal{V}ar(M,N)$ and $n_1,\ldots,n_\ell$ be ℓ fresh public names that do not occur in $\mathcal{N}ames(\varphi_1)\cup\mathcal{N}ames(\varphi_2)\cup\mathcal{N}ames(M,N)$. Let $\delta = \{y_1 \mapsto n_1,\ldots,y_\ell \mapsto n_\ell\}$. Since $L_1 \models \mathsf{i}(\tilde{M},\tilde{N})$, we have also that $L_1 \models \mathsf{i}(\tilde{M}\delta,\tilde{N}\delta)$. By Theorem 4.1, we have that $(\tilde{M}\delta = \tilde{N}\delta)\varphi_1$. As $\varphi_1 \approx \varphi_2$, we have also that $(\tilde{M}\delta = \tilde{N}\delta)\varphi_2$, and thus $(\tilde{M}=\tilde{N})\varphi_2$. This allows us to conclude. □

Example 4.10. Consider again the frames φ_1 and φ_2 from Example 4.7. Our procedure returns that the frames are not statically equivalent since

$$\Big(\mathsf{i}(\mathrm{mal}(w_1,b),w_1) \Leftarrow\Big) \in K_2$$

whereas $(\mathrm{mal}(w_1,b) \neq w_1)\varphi_1$.

4.6 Termination

The saturation process does not always terminate.

Example 4.11. Consider the convergent rewriting system consisting of the single rule $f(g(x)) \to g(h(x))$ and the frame $\varphi = \{w_1 \mapsto g(a)\}$ where $a \in \mathcal{N}$ is a private name. We have that

$$\mathrm{Init}(\varphi) \supseteq \{\Big(\mathsf{k}(w_1,g(a)) \Leftarrow\Big),\ \Big(\mathsf{k}(f(X),f(x)) \Leftarrow \mathsf{k}(X,x)\Big)\}.$$

By Narrowing we can add the statement $f_1 = \Big(\mathsf{k}(f(X),g(h(x))) \Leftarrow \mathsf{k}(X,g(x))\Big)$. Then we can apply F-Solving to solve its antecedent $\mathsf{k}(X,g(x))$ with the statement $\Big(\mathsf{k}(w_1,g(a)) \Leftarrow\Big)$ yielding the solved statement $\Big(\mathsf{k}(f(w_1),g(h(a))) \Leftarrow\Big)$. Now, applying iteratively F-Solving on f_1 and the newly generated statement, we generate an infinity of solved statements of the form $\Big(\mathsf{k}(f(\ldots f(w_1)\ldots),g(h(\ldots h(a)\ldots))) \Leftarrow\Big)$. Intuitively, this happens because our symbolic representation is unable to express that the function h can be nested an unbounded number of times when it occurs under an application of g.

The same kind of limitation already exists in the procedure implemented in the tool YAPA [22]. However, our symbolic representation which manipulates terms that are not necessarily ground and statements with antecedents allows us to go beyond YAPA. We are able for instance to treat equational theories such as malleable encryption and trapdoor commitment.

4.6.1 Generic Method for Proving Termination

We provide a generic method for proving termination, which we instantiate on several examples. In order to prove that the saturation algorithm terminates, we require the update function $\oplus$ to be *uniform*: i.e., the same recipe R' be used for all redundant solved deduction statements that have the same canonical form. Note that the soundness and completeness of the algorithm does not depend on the choice of the recipe R' when updating the knowledge base with a redundant statement (*cf.* Definition 4.4).

Definition 4.5 (projection)**.** We define the *projection* $\hat{f}$ of a deduction statement $f = \Big(\mathsf{k}(R,t) \Leftarrow \mathsf{k}(X_1,t_1),\ldots,\mathsf{k}(X_n,t_n)\Big)$ to be the Horn clause

$$\hat{f} = \Big(t \Leftarrow t_1,\ldots,t_n\Big).$$

We extend the projection to sets of statements K and define $\hat{K} = \{\hat{f} \mid f \in K\}$.

As usual, we identify projections which are equal up to bijective renaming of variables and we sometimes omit braces for the antecedents.

Proposition 4.3 (generic termination)**.** *The saturation algorithm terminates if $\oplus$ is uniform and there exist some functions $\mathcal{Q}$, m_f, m_e and some well-founded orders $<_f$ and $<_e$ such that for all frames φ, and for all knowledge bases K such that* $\mathrm{Init}(\varphi) \Longrightarrow^* K$*, we have that:*

1. $\{\hat{f} \mid f \in K|_{\mathsf{solved}}$ *and f is a deduction statement*$\} \subseteq \mathcal{Q}(\varphi)$ *and $\mathcal{Q}(\varphi)$ is finite;*
2. $\mathrm{m}_f(f_0) <_f \mathrm{m}_f(f_1)$ *where f_0, f_1 are defined as in rule* F-Solving*;*
3. $\mathrm{m}_e(f_0) <_e \mathrm{m}_e(f_1)$ *where f_0, f_1 are defined as in rule* E-Solving*.*

Proof. A *solved deduction statement* f is only added to K if there is no $f' \in K$ such that $\hat{f} = \hat{f}'$. Indeed, if $\hat{f} = \hat{f}'$ then $\hat{f}$ is redundant and an equational statement will be added instead. As $\{\hat{f} \mid f \in K|_{\mathsf{solved}}$ and f is a deduction statement$\} \subseteq \mathcal{Q}(\varphi)$ and $\mathcal{Q}(\varphi)$ is finite we conclude that only a finite number of solved deduction statements are in K.

An *unsolved deduction statement* f can be added in two ways.

- f can be added by the rule Narrowing. Since the number of solved deduction statements and the number of rewriting rules are finite the number of statements added by the rule Narrowing is also finite.
- f can be added by the rule F-Solving. The number of statements added by the rule F-Solving is bounded by the measure m_f which is strictly decreasing for a well-founded order.

An *equational statement* f can be added in three ways.

- f can be added when the knowledge base is updated with a redundant deduction statement. However, since $\oplus$ is uniform only a finite number of such statements is added.
- f can be added by the rule Unifying. Since the number of solved deduction statements is finite, the number of statements added by Unifying is bounded.
- f can be added by the rule E-Solving. The number of statements added by rule E-Solving is bounded by the measure m_e which is strictly decreasing for a well-founded order.

Altogether, this allows us to conclude. □

4.6.2 Applications

We now give several examples for which the saturation procedure indeed terminates. For each of these theories the definition of the function $\mathcal{Q}$ relies on the following notion of *extended subterm*.

Definition 4.6 (extended subterm)**.** Let t be a term, its set of *extended subterms* $\mathsf{st}_{ext}(t)$ (w.r.t. R), is the smallest set such that:

1. $t \in \mathsf{st}_{ext}(t)$,
2. $f(t_1, \ldots, t_k) \in \mathsf{st}_{ext}(t)$ implies $t_1, \ldots, t_k \in \mathsf{st}_{ext}(t)$,
3. $t' \in \mathsf{st}_{ext}(t)$ and $t' \to t''$ implies $t'' \in \mathsf{st}_{ext}(t)$.

This notation is extended to frames in the usual way.

All examples in this section rely on the same m_f and m_e. Let $\{\mathsf{k}(X_1, t_1), \ldots, \mathsf{k}(X_n, t_n)\}$ be the set of antecedents of a statement f. We define

$$\mathrm{m}_f(f) = (|\mathcal{V}ar(t_1, \ldots, t_n)|, \sum_{1 \leq i \leq n} |t_i|)$$

and $<_f$ is the lexicographical order on ordered pairs of integers. The measure m_e and the order $<_e$ are defined in the same way.

Subterm convergent equational theories. Abadi and Cortier [2] have shown that deduction and static equivalence are decidable for *subterm convergent* equational theories in polynomial time. We retrieve the same decidability results with our algorithm.

The termination proof for this class relies on the function $\mathcal{Q}$ where $\mathcal{Q}(\varphi)$ is defined as the smallest set that contains

1. $\Big(t \Leftarrow \emptyset\Big)$, where $t \in \mathsf{st}_{ext}(\varphi)$;
2. $\Big(f(x_1, \ldots, x_k) \Leftarrow x_1, \ldots, x_k\Big)$, where $ar(f) = k$.

Lemma 4.8. *For any frame φ, and any knowledge base K such that* $\mathrm{Init}(\varphi) \Longrightarrow K$*, we have that:*

1. $\{\hat{f} \mid f \in K$ *and f is a solved deduction statement*$\} \subseteq \mathcal{Q}(\varphi)$ *and $\mathcal{Q}(\varphi)$ is finite;*
2. $\mathrm{m}_f(f_0) <_f \mathrm{m}_f(f_1)$ *where f_0, f_1 are defined as in rule* F-Solving*;*
3. $\mathrm{m}_e(f_0) <_e \mathrm{m}_e(f_1)$ *where f_0, f_1 are defined as in rule* E-Solving

where $\mathcal{Q}$, m_f, m_e, $<_f$, and $<_e$ are defined w.r.t. the rewrite system as described above.

Proof. The proof of item 1 is done by induction on the number of saturation steps needed to reach K. To ease the induction we strengthen the induction hypothesis and prove a slightly stronger statement. We define $\mathcal{Q}'(\varphi, K)$ as the smallest set such that

1. $\Big(t \Leftarrow \emptyset\Big) \in \mathcal{Q}'(\varphi, K)$, where $t \in \mathsf{st}_{ext}(\varphi)$
2. $\Big(f(x_1, \ldots, x_k) \Leftarrow x_1, \ldots, x_k\Big) \in \mathcal{Q}'(\varphi, K)$, where $ar(f) = k$
3. $\Big(r\sigma \Leftarrow t_1, \ldots, t_k\Big) \in \mathcal{Q}'(\varphi, K)$, where:
   - $l \to r \in \mathsf{R}$
   - $\sigma : \mathcal{V}ar(l) \to \mathsf{st}_{ext}(\varphi)$ is a partial function

- $l\sigma = C[t_1, \ldots, t_k]$ for some context C
- $r\sigma \in \mathsf{st}(D[t_1, \ldots, t_k, u_1, \ldots, u_n])$ for some public context D and some terms u_i such that $\left(u_i \Leftarrow\right) \in \hat{K}$
- $\exists i : t_i \notin \mathcal{X}$

In the following when a projection $\hat{f}$ corresponds to one of the above 3 cases, we say that f is of type i ($1 \leq i \leq 3$). Note that a solved deduction statement is either of type 1 or 2. We prove that for any K such that $\text{Init}(\varphi) \Longrightarrow^* K$ we have that $\{\hat{f} \mid f \in K \text{ and } f \text{ is a deduction statement}\} \subseteq \mathcal{Q}'(\varphi, K)$. We have that $\{\hat{f} \mid \hat{f} \in \mathcal{Q}'(\varphi, K) \text{ and } \hat{f} \text{ is solved}\} \subseteq \mathcal{Q}(\varphi)$ and this allows us to conclude.

We prove the result by induction on the number of saturation steps of $\text{Init}(\varphi) \Longrightarrow^* K$.

Base case. It is clear that for all deduction statements $f \in \text{Init}(\varphi)$ we have that $\hat{f}$ is either of type 1 or type 2.

Inductive case. We assume that the result holds for K, i.e. $\hat{K} \subseteq \mathcal{Q}'(\varphi, K)$, and show that any possible application of a saturation rule preserves the result.

1. Consider a statement $f \in K$ of type 1, i.e. $\hat{f} = \left(t \Leftarrow\right)$. By applying rule Narrowing to it, we obtain a statement f' such that $\hat{f}' = \left(t' \Leftarrow\right)$ with $t \to t'$. As $t \in \mathsf{st}_{ext}(\varphi)$, we have that $t' \in \mathsf{st}_{ext}(\varphi)$ and therefore f' is of type 1.

2. Consider a statement $f \in K$ of type 2, i.e. $\hat{f} = \left(f(x_1, \ldots, x_k) \Leftarrow x_1, \ldots, x_k\right)$. As all positions of the term $f(x_1, \ldots, x_k)$, except the head are variables, rule Narrowing can only be applied at this position. Let $l \to r \in \mathsf{R}$ be the rewrite rule involved in this step. We obtain a statement f' such that $\hat{f}' = \left(r\tau \Leftarrow x_1\tau, \ldots, x_k\tau\right)$ where $\tau = \mathsf{mgu}(f(x_1, \ldots, x_k), l)$. By the definition of a rewrite rule, l is not a variable. Therefore, we have that $l = f(l_1, \ldots, l_k)$ for some terms $l_1, \ldots, l_k$ and $\hat{f}' = \left(r \Leftarrow l_1, \ldots, l_k\right)$. Let σ be such that $\mathcal{D}om(\sigma) = \emptyset$, $C = f(\_, \ldots, \_)$. It is clear that $\hat{f}'$ satisfies the three first conditions of a statement of type 3. Now, either $r \in T(\mathcal{F}, \emptyset)$ (i.e. r is a public ground term) and in such a case it is clear that the statement is redundant. Otherwise, we have that r is a strict subterm of l, i.e $r \in \mathsf{st}(l_j)$ for some $1 \leq j \leq k$. Therefore the fourth condition also holds. Now, assume that all the l_i are variables (i.e. f' is solved), we show it is redundant and it is not added to the knowledge base. Indeed, in such a situation, we necessarily have that r is a variable (remember that $r \in \mathsf{st}(l_j)$) and therefore the statement f' is redundant. Otherwise, the resulting fact is of type 3.

3. Consider a statement $f \in K$ of type 3. Let $\hat{f} = \left(r\sigma \Leftarrow t_1, \ldots, t_k\right)$. In such a case, there exist a rewrite rule $l \to r$, a partial function $\sigma : \mathcal{V}ar(l) \to \mathsf{st}_{ext}(\varphi)$, a context C such that $l\sigma = C[t_1, \ldots, t_k]$ and we have that $r\sigma \in \mathsf{st}(D[t_1, \ldots, t_k, u_1, \ldots, u_n])$ for some public context D and some terms u_i such that $\left(u_i \Leftarrow\right) \in \hat{K}$. Assume that one of the antecedents of f is being solved by rule F-Solving with a solved statement $f' \in K$. We assume w.l.o.g. that t_1 is being solved. We distinguish two cases depending on the type of f'.

   - *Case 1:* $\hat{f}' = \left(u_0 \Leftarrow\right)$. Let $\tau = \mathsf{mgu}(u_0, t_1)$. The projection of the statement resulting from the F-Solving rule is $\hat{f}'' = \left(r\sigma\tau \Leftarrow t_2\tau, \ldots, t_k\tau\right)$. We consider $\sigma' = \tau \cup \sigma$, $C' = C[u_0, \ldots, \_]$ and $D' = D$. We can show that the first four conditions hold. If the last condition does not hold, and because the fourth holds, the resulting statement must be either of type 1 or redundant and therefore not added to the knowledge base.
   - *Case 2:* $\hat{f}' = \left(f(x_1, \ldots, x_k) \Leftarrow x_1, \ldots, x_k\right)$. Let $\tau = \mathsf{mgu}(f(x_1, \ldots, x_k), t_1)$. As t_1 is not a variable, we have that $t_1 = f(s_1, \ldots, s_\ell)$. The projection statement resulting

from the application of the rule F-Solving is $\hat{f''} = \Big(r\sigma \Leftarrow s_1, \ldots, s_\ell, t_2, \ldots, t_k\Big)$. We can show that the first four conditions hold. If the last condition does not hold, and because the fourth holds, the resulting statement must be either of type 1 or redundant and therefore not added to the knowledge base.

To show items 2 and 3 it remains to be proven that m_f and m_e strictly decrease after an antecedent of an unsolved statement is solved. As an antecedent can only be solved by statements of type 1 or 2 this is easily shown by a case analysis. We detail the proof for m_f. The case of m_e can be done in a similar way.
Let $f_1 = \Big(\mathsf{k}(R,t) \Leftarrow \mathsf{k}(X_1,t_1), \ldots \mathsf{k}(X_n,t_n)\Big)$.

- Suppose f_1 is solved by a solved statement f_2 of type 1. Let $\hat{f}_2 = \Big(u \Leftarrow\Big)$ where $u \in \mathsf{st}_{ext}(\varphi)$ and $\sigma = \mathsf{mgu}(u, t_1)$. There are two possible cases. Either $u = t_1$. As $u \in \mathsf{st}_{ext}(\varphi)$ we have that u is ground and $\mathcal{D}om(\sigma) = \emptyset$. In this case $|\mathcal{V}ar(t_2, \ldots, t_n)| = |\mathcal{V}ar(t_1, \ldots, t_n)|$ but $\sum_{2 \leq i \leq n} |t_i| < \sum_{1 \leq i \leq n} |t_i|$. Or $u \neq t_1$ and $|\mathcal{V}ar(t_2, \ldots t_n)| < |\mathcal{V}ar(t_1, \ldots t_n)|$.

- Suppose f_1 is solved by a solved statement f_2 of type 2. Let $\hat{f}_2 = \Big(f(x_1, \ldots, x_k) \Leftarrow x_1, \ldots, x_k\Big)$ and $\sigma = \mathsf{mgu}(u, t_1)$. As $t_1 \notin \mathcal{X}$ we have that $t_1 = f(s_1, \ldots, s_k)$. We have that $\sigma = \{x_1 \mapsto s_1, \ldots, x_k \mapsto s_k\}$ and the resulting statement f_0 is such that

$$\hat{f}_0 = \Big(t\sigma \Leftarrow s_1, \ldots, s_k, t_2, \ldots, t_n\Big).$$

  Thus, we have that $|\mathcal{V}ar(s_1, \ldots, s_k, t_2, \ldots, t_n)| = |\mathcal{V}ar(t_1, \ldots, t_n)|$ and $\sum_{u \in \{s_1, \ldots, s_k, t_2, \ldots, t_n\}} |u| < \sum_{1 \leq i \leq n} |t_i|$.

This allows us to conclude the proof.

□

Malleable encryption. We also obtain termination for the equational theory E_{mal} described in Example 4.1. This is a toy example that does not fall in the class studied in [2]. Indeed, this theory is not *locally stable*: the set of terms in normal form deducible from a frame φ cannot always be obtained by applying public contexts over a finite set (called $sat(\varphi)$ in [2]) of ground terms.

As a witness, let $k \in \mathcal{N}$ be a private name and consider the frame $\varphi_2 = \{w_1 \mapsto \mathrm{enc}(b, k)\}$ introduced in Example 4.7. Among the terms that are deducible from φ_2, we have those of the form $\mathrm{enc}(t, k)$ where t represents any term deducible from φ_2. From this observation, it is easy to see that $\mathsf{E}_{\mathrm{mal}}$ is not locally stable.

Our procedure does not have this limitation. A prerequisite for termination is that the set of terms in normal form deducible from a frame is exactly the set of terms obtained by nesting in all possible ways a finite set of contexts. The theory $\mathsf{E}_{\mathrm{mal}}$ falls in this class. In particular, for the frame φ_2, our procedure produces the statement $f_9 = \Big(\mathsf{k}(\mathrm{mal}(w_1, Y_2), \mathrm{enc}(z, k)) \Leftarrow \mathsf{k}(Y_2, z)\Big)$ allowing us to capture all the terms of the form $\mathrm{enc}(t, k)$ by the means of a single deduction statement.

The termination proof relies on the function $\mathcal{Q}$ where $\mathcal{Q}(\varphi)$ is defined as the smallest set that contains:

1. $\Big(t \Leftarrow\Big)$, for every $t \in \mathsf{st}_{ext}(\varphi)$;
2. $\Big(f(x_1, x_2) \Leftarrow x_1, x_2\Big)$, where $f \in \{\mathrm{enc}, \mathrm{dec}, \mathrm{mal}\}$;
3. $\Big(\mathrm{enc}(x, t) \Leftarrow x\Big)$, if there exists t' such that $\mathrm{enc}(t', t) \in \mathsf{st}_{ext}(\varphi)$.

Lemma 4.9. *For any frame φ, and any knowledge base K such that* $\mathrm{Init}(\varphi) \Longrightarrow^* K$*, we have that:*

1. $\{\hat{f} \mid f \in K$ *and* f *is a solved deduction statement*$\} \subseteq \mathcal{Q}(\varphi)$ *and* $\mathcal{Q}(\varphi)$ *is finite;*
2. $\mathrm{m}_f(f_0) <_f \mathrm{m}_f(f_1)$ *where* f_0, f_1 *are defined as in rule* F-Solving*;*
3. $\mathrm{m}_e(f_0) <_e \mathrm{m}_e(f_1)$ *where* f_0, f_1 *are defined as in rule* E-Solving

where $\mathcal{Q}$, m_f, m_e, $<_f$, *and* $<_e$ *are defined as described above.*

Proof. The proof of item 1 is done by induction on the number of saturation steps of $\mathrm{Init}(\varphi) \Longrightarrow^* K$. To ease the induction we strengthen the induction hypothesis and prove a slightly stronger statement. We define $\mathcal{Q}'(\varphi)$ as the smallest set such that:

1. $\Big(t \Leftarrow\Big) \in \mathcal{Q}'(\varphi)$, for every $t \in \mathsf{st}_{ext}(\varphi)$
2. $\Big(f(x_1, x_2) \Leftarrow x_1, x_2\Big) \in \mathcal{Q}'(\varphi)$, where $f \in \{\mathrm{enc}, \mathrm{dec}, \mathrm{mal}\}$
3. $\Big(\mathrm{enc}(x, t) \Leftarrow x\Big) \in \mathcal{Q}'(\varphi)$, if there exists t' such that $\mathrm{enc}(t', t) \in \mathsf{st}_{ext}(\varphi)$
4. $\Big(x \Leftarrow \mathrm{enc}(x, y), y\Big) \in \mathcal{Q}'(\varphi)$
5. $\Big(\mathrm{enc}(z, y) \Leftarrow \mathrm{enc}(x, y), z\Big) \in \mathcal{Q}'(\varphi)$
6. $\Big(t \Leftarrow t_1, \ldots, t_k\Big) \in \mathcal{Q}'(\varphi)$, if $t \in \mathsf{st}_{ext}(\varphi)$ and $C[t_1, \ldots, t_k] \in \mathsf{st}_{ext}(\varphi)$ for some context C
7. $\Big(x \Leftarrow x, t_1, \ldots, t_k\Big)$, where $C[t_1, \ldots, t_k] \in \mathsf{st}_{ext}(\varphi)$ for some context C

In the following when a projection $\hat{f}$ corresponds to one of the above 7 cases, we say that f is of type i ($1 \leq i \leq 7$). We prove that for any knowledge base K such that $\mathrm{Init}(\varphi) \Longrightarrow^* K$ we have that $\{\hat{f} \mid f \in K$ and f is a deduction statement$\} \subseteq \mathcal{Q}'(\varphi)$. It is easy to see that $\{\hat{f} \mid \hat{f} \in \mathcal{Q}'(\varphi)$ and $\hat{f}$ is solved$\} \subseteq \mathcal{Q}(\varphi)$ and this will indeed allows us to conclude.

We prove the result by induction on the number of saturation steps of $\mathrm{Init}(\varphi) \Longrightarrow^* K$.

Base case. It is clear that for all deduction statements $f \in \mathrm{Init}(\varphi)$ we have that $\hat{f}$ is either of type 1 or type 2.

Inductive case. We assume that the result holds for K and show that any possible application of a saturation rule preserves the result.

- Consider a statement $f \in K$ of type 1, i.e. $\hat{f} = \Big(t \Leftarrow\Big)$ with $t \in \mathsf{st}_{ext}(\varphi)$. By applying rule Narrowing, we obtain a statement f' such that $\hat{f}' = \Big(t' \Leftarrow\Big)$, and $t \to t'$. As $t \in \mathsf{st}_{ext}(\varphi)$, it follows that $t' \in \mathsf{st}_{ext}(\varphi)$ and therefore f' is a statement of type 1.
- Consider a statement $f \in K$ of type 2 such that $\hat{f} = \Big(f(x_1, x_2) \Leftarrow x_1, x_2\Big)$. By applying the rule Narrowing we obtain a statement of type 4, or 5.
- Consider a statement $f \in K$ of type 3, then $\hat{f} = \Big(\mathrm{enc}(x, t) \mid x\Big)$ and the rule Narrowing can only be applied on a position in t. Therefore, Narrowing will produce another statement $\hat{f}' = \Big(\mathrm{enc}(x, u) \Leftarrow x\Big)$, where $t \to u$. As there exists t' such that $\mathrm{enc}(t', t) \in \mathsf{st}_{ext}(\varphi)$ by definition of st_{ext}, $\mathrm{enc}(t', u) \in \mathsf{st}_{ext}(\varphi)$ yielding again a statement of type 3.

- Consider a statement $f \in K$ of type 4, then its unsolved antecedents can be solved using a statement of type 1, 2 or 3. In the first case, we obtain a statement of type 6. In the second case, we obtain a redundant statement. In the third case, we obtain a statement of type 7.
- Consider a statement $f \in K$ of type 5, its unsolved antecedent can be solved using a statement of type 1, 2 or 3. In the first case, we obtain a statement of type 3. In the second and third case, we obtain a redundant statement.
- Consider a statement $f \in K$ of type 6 or 7, its unsolved antecedents can be solved using a statement of type 1, 2 or 3. Let f' be the new statement obtained by applying the F-SOLVING rule. If f' is unsolved, it has the same type as f. If f' is solved, it is either of type 1 if f is of type 6 or it is redundant if f is of type 7.

To show items 2 and 3 it remains to be proven that m_f and m_e strictly decrease after an antecedent of an unsolved statement is solved. As antecedents can only be solved by statements of type $1-3$ this is easily shown by a case analysis. We detail the proof for m_f. The case of m_e can be done in a similar way.
Let $f_1 = \Big(\mathsf{k}(R,t) \Leftarrow \mathsf{k}(X_1,t_1), \ldots \mathsf{k}(X_n,t_n)\Big)$. The case where f_1 is solved by a statement f_2 of type 1 (resp. type 2) is similar to the proof done in Lemma 4.8. It remains the case where f_2 is of type 3.

Let $\hat{f}_2 = \Big(\mathrm{enc}(x,u) \Leftarrow x\Big)$ and $\sigma = \mathsf{mgu}(\mathrm{enc}(x,u), t_1)$. As there exists u' such that $\mathrm{enc}(u',u) \in \mathsf{st}_{ext}(\varphi)$ we have that u is ground. As $t_1 \notin \mathcal{X}$ we have that $t_1 = \mathrm{enc}(t_1', t_1'')$.

The projection of the resulting statement f_0 is $\hat{f}_0 = \Big(t\sigma \Leftarrow x\sigma, t_2\sigma, \ldots, t_n\sigma\Big)$. We distinguish two cases. Either $\sigma = \{x \mapsto t_1'\}$ and $\hat{f}_0 = \Big(t \Leftarrow t_1', t_2, \ldots, t_n\Big)$. In such a case $|\mathcal{V}ar(t_2,\ldots,t_n)| \leq |\mathcal{V}ar(t_1,\ldots,t_n)|$ and $\sum_{2\leq i\leq n} |t_i| < \sum_{1\leq i\leq n} |t_i|$. Otherwise, we have that $|\mathcal{V}ar(t_2,\ldots,t_n)| < |\mathcal{V}ar(t_1,\ldots,t_n)|$.

□

Trap-door commitment. The following convergent equational theory $\mathsf{E}_{\mathrm{tdcommit}}$ is a model for trap-door commitment

1. $\mathrm{open}(\mathrm{tdcommit}(x,y,z),y) = x$
2. $\mathrm{tdcommit}(x_2, \mathrm{f}(x_1,y,z,x_2), z) = \mathrm{tdcommit}(x_1,y,z)$
3. $\mathrm{open}(\mathrm{tdcommit}(x_1,y,z), \mathrm{f}(x_1,y,z,x_2)) = x_2$
4. $\mathrm{f}(x_2, \mathrm{f}(x_1,y,z,x_2), z, x_3) = \mathrm{f}(x_1,y,z,x_3)$

We will refer below to the four corresponding rewrite rules as R1, R2, R3 and R4.

As said in the introduction, we encountered this equational theory when studying electronic voting protocols. The term $\mathrm{tdcommit}(m,r,td)$ models the commitment of the message m under the key r using an additional trap-door td. Such a commitment scheme allows a voter who has performed a commitment to open it in different ways using its trap-door. Hence, trap-door bit commitment $\mathrm{tdcommit}(v,r,td)$ does not bind the voter to the vote v. This is useful to ensure privacy-type properties in e-voting and in particular receipt-freeness [114]. With such a scheme, even if a coercer requires the voter to reveal his commitment, this does not give any useful information to the coercer as the commitment can be viewed as the commitment of any vote (depending on the key that will be used to open it).

For the same reason as $\mathsf{E}_{\mathrm{mal}}$, the theory of trap-door commitment described below cannot be handled by the algorithms described in [2, 22]. Our termination proof relies on the function $\mathcal{Q}$ where $\mathcal{Q}(\varphi)$ is the smallest set that contains:

1. $\Big(t \Leftarrow\Big)$, for every $t \in \mathsf{st}_{ext}(\varphi)$;

2. $\Big(\text{tdcommit}(t_1, r, tp) \Leftarrow \Big)$ such that $\text{f}(t_1, r, tp, t_2) \in \mathsf{st}_{ext}(\varphi)$ for some t_2;
3. $\Big(g(x_1, \ldots, x_k) \Leftarrow x_1, \ldots, x_k\Big)$, where $g \in \{\text{open}, \text{tdcommit}, \text{f}\}$ and $ar(g) = k$;
4. $\Big(\text{f}(t_1, r, tp, x) \Leftarrow x\Big)$, such that $\text{f}(t_1, r, tp, t_2) \in \mathsf{st}_{ext}(\varphi)$ for some t_2.

Lemma 4.10. *For any frame φ, and any knowledge base K such that $\text{Init}(\varphi) \Longrightarrow^* K$, we have that:*

1. *$\{\hat{f} \mid f \in K$ and f is a solved deduction statement$\} \subseteq \mathcal{Q}(\varphi)$ and $\mathcal{Q}(\varphi)$ is finite;*
2. *$\text{m}_f(f_0) <_f \text{m}_f(f_1)$ where f_0, f_1 are defined as in rule* F-Solving*;*
3. *$\text{m}_e(f_0) <_e \text{m}_e(f_1)$ where f_0, f_1 are defined as in rule* E-Solving

where $\mathcal{Q}(\varphi)$ is defined as the smallest set that contains:

1. *$\Big(t \Leftarrow \Big)$, for every $t \in \mathsf{st}_{ext}(\varphi)$*
2. *$\Big(\text{tdcommit}(t_1, r, tp) \Leftarrow \Big)$ such that $\text{f}(t_1, r, tp, t_2) \in \mathsf{st}_{ext}(\varphi)$ for some term t_2*
3. *$\Big(g(x_1, \ldots, x_k) \Leftarrow x_1, \ldots, x_k\Big)$, where $g \in \{\text{open}, \text{tdcommit}, \text{f}\}$ and $ar(g) = k$*
4. *$\Big(\text{f}(t_1, r, tp, x) \Leftarrow x\Big)$, such that $\text{f}(t_1, r, tp, t_2) \in \mathsf{st}_{ext}(\varphi)$ for some t_2*

and m_f, m_e, $<_f$, and $<_e$ are defined as above.

Proof. The proof of item 1 is done by induction on the number of saturation steps of $\text{Init}(\varphi) \Longrightarrow^* K$. To ease the induction we strengthen the induction hypothesis and prove a slightly stronger statement. We define $\mathcal{Q}'(\varphi)$ as the smallest set that contains:

1. $\Big(t \Leftarrow \Big)$, for every $t \in \mathsf{st}_{ext}(\varphi)$
2. $\Big(\text{tdcommit}(t_1, r, tp) \Leftarrow \Big)$ such that $\text{f}(t_1, r, tp, t_2) \in \mathsf{st}_{ext}(\varphi)$ for some t_2
3. $\Big(g(x_1, \ldots, x_k) \Leftarrow x_1, \ldots, x_k\Big)$, where $g \in \{\text{open}, \text{tdcommit}, \text{f}\}$ and $ar(g) = k$
4. $\Big(\text{f}(t_1, r, tp, x) \Leftarrow x\Big)$, such that $\text{f}(t_1, r, tp, t_2) \in \mathsf{st}_{ext}(\varphi)$ for some t_2
5. $\Big(x \Leftarrow \text{tdcommit}(x, y, z), y\Big)$
6. $\Big(\text{tdcommit}(x_1, y, z) \Leftarrow x_2, \text{f}(x_1, y, z, x_2), z\Big)$
7. $\Big(x_2 \Leftarrow \text{tdcommit}(x_1, y, z), \text{f}(x_1, y, z, x_2)\Big)$
8. $\Big(\text{f}(x_1, y, z, x_3) \Leftarrow x_2, \text{f}(x_1, y, z, x_2), z, x_3]$
9. $\Big(x_2 \Leftarrow x_1, y, z, f(x_1, y, z, x_2)\Big)$
10. $\Big(x_2 \Leftarrow \text{tdcommit}(x, y, z), x, y, z, x_2\Big)$
11. $\Big(x \Leftarrow f(t_1, r, tp, x)\Big)$ for every $t_1, r, tp \in \mathsf{st}_{ext}(\varphi)$

12. $\Big(x \Leftarrow \text{tdcommit}(t, r, tp), x\Big)$ for every $t, r, tp \in \mathsf{st}_{ext}(\varphi)$
13. $\Big(x \Leftarrow x, t_1, \ldots, t_k\Big)$ for every $t_1, \ldots, t_k \in \mathsf{st}_{ext}(\varphi)$
14. $\Big(t \Leftarrow \text{tdcommit}(t_1, r, tp)\Big)$ for every $t, t_1, r, tp \in \mathsf{st}_{ext}(\varphi)$
15. $\Big(t \Leftarrow t_1, \ldots, t_k\Big)$ for every $t, t_1, \ldots, t_k \in \mathsf{st}_{ext}(\varphi)$, $k \geq 1$
16. $\Big(\text{tdcommit}(t, r, tp) \Leftarrow t_1, \ldots, t_k\Big)$, $\exists t'\ \text{f}(t, r, tp, t') \in \mathsf{st}_{ext}(\varphi)$, $t_1, \ldots, t_k \in \mathsf{st}_{ext}(\varphi), k \geq 1$
17. $\Big(\text{tdcommit}(t, r, tp) \Leftarrow x, t_1, \ldots, t_k\Big)$, $\exists t'\ \text{f}(t, r, tp, t') \in \mathsf{st}_{ext}(\varphi)$, $t_1, \ldots, t_k \in \mathsf{st}_{ext}(\varphi), k \geq 1$
18. $\Big(\text{f}(t, r, tp, x) \Leftarrow x, t_1, \ldots, t_k\Big)$, $\exists t'\ \text{f}(t, r, tp, t') \in \mathsf{st}_{ext}(\varphi)$, $t_1, \ldots, t_k \in \mathsf{st}_{ext}(\varphi)$
19. $\Big(\text{f}(t, r, tp, x) \Leftarrow x, x', t_1, \ldots, t_k\Big)$, $\exists t'\ f(t, r, tp, t') \in \mathsf{st}_{ext}(\varphi)$, $t_1, \ldots, t_k \in \mathsf{st}_{ext}(\varphi)$

In the following when a projection $\hat{f}$ corresponds to one of the above 19 cases, we say that f is of type i ($1 \leq i \leq 19$). We prove that for any K such that $\text{Init}(\varphi) \Longrightarrow^* K$ we have that $\{\hat{f} \mid f \in K \text{ and } f \text{ is a deduction statement}\} \subseteq \mathcal{Q}'(\varphi)$. It is easy to see that $\{\hat{f} \mid \hat{f} \in \mathcal{Q}'(\varphi) \text{ and } \hat{f} \text{ is solved}\} \subseteq \mathcal{Q}(\varphi)$, which will indeed allows us to conclude. We prove the result by induction on the number of saturation steps of $\text{Init}(\varphi) \Longrightarrow^* K$.

Base case. It is clear that all deduction statements $f \in \text{Init}(\varphi)$ are either of type 1 or type 3.

Inductive case. We assume that the result holds for K and show that any possible application of a saturation rule preserves the result. We summarize case analysis in the following two matrices.

NARROWING	R1	R2	R3	R4
type 1	1	1	1	1
type 2	2	2	2	2
type 3	5	6	7	8
type 4	4	4	4	4

F-SOLVING	type 1	type 2	type 3	type 4
type 5	15	15	redundant	impossible
type 6	16	impossible	redundant	17
type 7	11 or 14	11	9 or 10	12
type 8	18	impossible	redundant	19
type 9	15	impossible	redundant	13
type 10	13	13	redundant	impossible
type 11	1	impossible	13	redundant
type 12	redundant	redundant	13	impossible
type 13	13 or redundant	13 or redundant	13	13
type 14	1	1	15	impossible
type 15	15 or 1	15 or 1	15	15
type 16	16 or 2	16 or 2	16	16
type 17	17 or 2	17 or 2	17	17
type 18	18 or 4	18 or 4	18	18
type 19	19 or 4	19 or 4	19	19

Items 2 and 3 are shown as in Lemma 4.9.

□

Blind signatures. The following convergent equational theory E_{blind} has been introduced in [98] for modeling blind signatures in e-voting protocols. Abadi and Cortier have shown that deduction and static equivalence are decidable for this theory [2].

1. $\mathrm{unblind}(\mathrm{blind}(x,y),y) = x$
2. $\mathrm{unblind}(\mathrm{sign}(\mathrm{blind}(x,y),z),y) = \mathrm{sign}(x,z)$
3. $\mathrm{checksign}(\mathrm{sign}(x,y),\mathrm{pk}(y)) = x$

We will refer below to the three corresponding rewrite rules as R1, R2 and R3.

Lemma 4.11. *For any frame φ, and any knowledge base K such that $\mathrm{Init}(\varphi) \Longrightarrow^* K$, we have that:*

1. $\{\hat{f} \mid f \in K$ *and* f *is a solved deduction statement*$\} \subseteq \mathcal{Q}(\varphi)$ *and* $\mathcal{Q}(\varphi)$ *is finite;*
2. $\mathrm{m}_f(f_0) <_f \mathrm{m}_f(f_1)$ *where* f_0, f_1 *are defined as in rule* F-Solving*;*
3. $\mathrm{m}_e(f_0) <_e \mathrm{m}_e(f_1)$ *where* f_0, f_1 *are defined as in rule* E-Solving

where $\mathcal{Q}(\varphi)$ is defined as the smallest set that contains:

1. $\Big(t \Leftarrow\Big)$, *for every* $t \in \mathsf{st}_{ext}(\varphi)$
2. $\Big(f(x_1,\ldots,x_k) \Leftarrow x_1,\ldots,x_k\Big)$, *where* $f \in \mathcal{F}$ *and* $ar(f) = k$
3. $\Big(\mathrm{sign}(t,x) \Leftarrow x\Big)$, *for every* $t \in \mathsf{st}_{ext}(\varphi)$
4. $\Big(\mathrm{sign}(t,t') \Leftarrow\Big)$, *for every* $t,t' \in \mathsf{st}_{ext}(\varphi)$

and m_f, m_e, $<_f$, and $<_e$ are defined as described above.

Proof. The proof of item 1 is done by induction on the number of saturation steps of $\mathrm{Init}(\varphi) \Longrightarrow^* K$. To ease the induction we strengthen the induction hypothesis and prove a slightly stronger statement. We define $\mathcal{Q}'(\varphi)$ as the smallest set that contains:

1. $\Big(t \Leftarrow\Big)$, for every $t \in \mathsf{st}_{ext}(\varphi)$
2. $\Big(f(x_1,\ldots,x_k) \Leftarrow x_1,\ldots,x_k\Big)$, where $f \in \mathcal{F}$ and $ar(f) = k$
3. $\Big(\mathrm{sign}(t,x) \Leftarrow x\Big)$, for every $t \in \mathsf{st}_{ext}(\varphi)$
4. $\Big(\mathrm{sign}(t,t') \Leftarrow\Big)$, for every $t,t' \in \mathsf{st}_{ext}(\varphi)$
5. $\Big(x \Leftarrow \mathrm{blind}(x,y),y\Big)$
6. $\Big(\mathrm{sign}(x,z) \Leftarrow \mathrm{sign}(\mathrm{blind}(x,y),z),y\Big)$
7. $\Big(x \Leftarrow \mathrm{sign}(x,y),\mathrm{pk}(y)\Big)$
8. $\Big(\mathrm{sign}(x,z) \Leftarrow \mathrm{blind}(x,y),z,y\Big)$
9. $\Big(x \Leftarrow \mathrm{sign}(x,y),y\Big)$

10. $\Big(x \Leftarrow x, y, \mathrm{pk}(y)\Big)$
11. $\Big(t \Leftarrow t_1, \ldots, t_k\Big)$ if $C\Big(t_1, \ldots, t_k\Big) \in \mathsf{st}_{ext}(\varphi)$ for some context C and $t \in \mathsf{st}_{ext}(\varphi)$
12. $\Big(\mathrm{sign}(t, t') \Leftarrow t_1, \ldots, t_k\Big)$ if $C\Big(t_1, \ldots, t_k\Big) \in \mathsf{st}_{ext}(\varphi)$ for some context C, $k \geq 1$, and $t, t' \in \mathsf{st}_{ext}(\varphi)$
13. $\Big(t \Leftarrow \mathrm{pk}(t')\Big)$, for every $t, t' \in \mathsf{st}_{ext}(\varphi)$
14. $\Big(x \Leftarrow \mathrm{sign}(x, t)\Big)$, for every $t \in \mathsf{st}_{ext}(\varphi)$
15. $\Big(t \Leftarrow y, \mathrm{pk}(y)\Big)$, for every $t \in \mathsf{st}_{ext}(\varphi)$
16. $\Big(\mathrm{sign}(t, z) \Leftarrow z, t_1, \ldots, t_k\Big)$ if $C\Big(t_1, \ldots, t_k\Big) \in \mathsf{st}_{ext}(\varphi)$ for some context C, $k \geq 1$, and $t \in \mathsf{st}_{ext}(\varphi)$
17. $\Big(x \Leftarrow x, t_1, \ldots, t_k\Big)$ if $C\Big(t_1, \ldots, t_k\Big) \in \mathsf{st}_{ext}(\varphi)$ for some context C

In the following when a projection $\hat{f}$ corresponds to one of the above 17 cases, we say that f is of type i ($1 \leq i \leq 17$). We prove that for any K such that $\mathrm{Init}(\varphi) \Longrightarrow^* K$ we have that $\{\hat{f} \mid f \in K \text{ and } f \text{ is a deduction statement}\} \subseteq \mathcal{Q}'(\varphi)$. It is easy to see that $\{\hat{f} \mid \hat{f} \in \mathcal{Q}'(\varphi) \text{ and } \hat{f} \text{ is solved}\} \subseteq \mathcal{Q}(\varphi)$, which will indeed allows us to conclude. We prove the result by induction on the number of saturation steps of $\mathrm{Init}(\varphi) \Longrightarrow^* K$.

Base case. It is clear that all deduction statements $f \in \mathrm{Init}(\varphi)$ are either of type 1 or type 2.

Inductive case. We assume that the result holds for K and show that any possible application of a saturation rule preserves the result. We summarize the case analysis in the following two matrices.

Narrowing	R1	R2	R3
type 1	1	1	1
type 2	5	6	7
type 3	3	3	3
type 4	4	4	4

F-Solving	type 1	type 2	type 3	type 4
type 5	11	redundant	impossible	impossible
type 6	12	8	16	12
type 7	13 or 14	9 or 10	15	13
type 8	16	redundant	impossible	impossible
type 9	11	redundant	1	11
type 10	17	redundant	impossible	impossible
type 11	11 or 1	11	11	11 or 1
type 12	12 or 4	12	12	12 or 4
type 13	1	11	impossible	impossible
type 14	1	17	11	1
type 15	11	1	impossible	impossible
type 16	16 or 3	16	16	16 or 3
type 17	17 or redundant	17	17	17 or redundant

Items 2 and 3 are shown as in Lemma 4.9. □

Addition. The following convergent equational theory E_{add} is a simple model of addition introduced and proved decidable in [2]:

1. $\mathrm{plus}(x, \mathrm{suc}(y)) = \mathrm{plus}(\mathrm{suc}(x), y)$
2. $\mathrm{plus}(x, \mathrm{zero}) = x$
3. $\mathrm{pred}(\mathrm{suc}(x)) = x$

We will refer below to the three corresponding rewrite rules as $\mathsf{R1}$, $\mathsf{R2}$ and $\mathsf{R3}$. The saturation procedure presented terminates on this equational theory as well.

Lemma 4.12. *For any frame φ, and any knowledge base K such that $\mathrm{Init}(\varphi) \Longrightarrow^* K$, we have that:*

1. $\{\hat{f} \mid f \in K$ *and* f *is a solved deduction statement*$\} \subseteq \mathcal{Q}(\varphi)$ *and* $\mathcal{Q}(\varphi)$ *is finite;*
2. $\mathrm{m}_f(f_0) <_f \mathrm{m}_f(f_1)$ *where* f_0, f_1 *are defined as in rule* F-Solving*;*
3. $\mathrm{m}_e(f_0) <_e \mathrm{m}_e(f_1)$ *where* f_0, f_1 *are defined as in rule* E-Solving

where $\mathcal{Q}(\varphi)$ is defined as the smallest set that contains:

1. $\Big(t \Leftarrow\Big)$, *for every* $t \in \mathsf{st}_{ext}(\varphi)$
2. $\Big(f(x_1, \ldots, x_k) \Leftarrow x_1, \ldots, x_k\Big)$, *where* $f \in \{\mathrm{suc}, \mathrm{plus}, \mathrm{pred}, \mathrm{zero}\}$ *and* $ar(f) = k$
3. $\Big(\mathrm{plus}(\mathrm{suc}^n(x), t) \Leftarrow x\Big)$, *if* $\mathrm{suc}^n(t) \in \mathsf{st}_{ext}(\varphi)$ *for* $n \geq 0$

and m_f, m_e, $<_f$, and $<_e$ are defined as above.

Proof. The proof of item 1 is done by induction on the number of saturation steps of $\mathrm{Init}(\varphi) \Longrightarrow^* K$. To ease the induction we strengthen the induction hypothesis and prove a slightly stronger statement. We define $\mathcal{Q}'(\varphi)$ as the smallest set that contains:

1. $\Big(t \Leftarrow \emptyset\Big)$, for every $t \in \mathsf{st}_{ext}(\varphi)$
2. $\Big(f(x_1, \ldots, x_k) \Leftarrow x_1, \ldots, x_k\Big)$, where $f \in \mathcal{F}$ and $ar(f) = k$
3. $\Big(\mathrm{plus}(\mathrm{suc}^n(x), t) \Leftarrow x\Big)$, if $\mathrm{suc}^n(t) \in \mathsf{st}_{ext}(\varphi)$ for $n \geq 0$
4. $\Big(x \Leftarrow x, \mathrm{zero}\Big)$
5. $\Big(\mathrm{plus}(\mathrm{suc}(x), y) \Leftarrow x, \mathrm{suc}(y)\Big)$
6. $\Big(x \Leftarrow \mathrm{suc}(x)\Big)$

In the following when a projection $\hat{f}$ corresponds to one of the above 6 cases, we say that f is of type i $(1 \leq i \leq 6)$. We prove that for any K such that $\mathrm{Init}(\varphi) \Longrightarrow^* K$ we have that $\{\hat{f} \mid f \in K$ and f is a solved deduction statement$\} \subseteq \mathcal{Q}'(\varphi)$. It is easy to see that $\{\hat{f} \mid \hat{f} \in \mathcal{Q}'(\varphi)$ and $\hat{f}$ is solved$\} \subseteq \mathcal{Q}(\varphi)$, which will indeed allows us to conclude. We prove the result by induction on the number of saturation steps of $\mathrm{Init}(\varphi) \Longrightarrow^* K$.

Base case. It is clear that all deduction statements $f \in \mathrm{Init}(\varphi)$ are either of type 1 or type 2.

Inductive case. We assume that the result holds for K and we show that any possible application of a saturation rule preserves the result. We summarize the case analysis in the following two matrices.

NARROWING	R1	R2	R3
type 1	1	1	1
type 2	5	4	6
type 3	3	redundant or 3	3

F-SOLVING	type 1	type 2	type 3
type 4	redundant	redundant	impossible
type 5	3	redundant	impossible
type 6	1	redundant	impossible

To show item 2 and 3, it remains to be proven that m_f and m_e strictly decrease after an antecedent of an unsolved statement is solved. An antecedent can only be solved by statements of type 1, 2 or 3. We show the result by a case analysis.
Let $f_1 = \Big(\mathsf{k}(R,t) \Leftarrow \mathsf{k}(X_1,t_1),\ldots,\mathsf{k}(X_n,t_n)\Big)$.

- If the solved statement is of type 1 or 2, the proof is similar to the reasoning done in Lemma 4.8.
- It is easy to see that a solved statement of type 3 cannot be used to solved an antecedent of an unsolved statement (types 4-6). Indeed, the antecedents which are not variables, are either zero or a term of the form $s(x)$ and hence unification is impossible.

Let $f = \Big(\mathsf{i}(U,V) \Leftarrow \mathsf{k}(X_1,t_1),\ldots,\mathsf{k}(X_n,t_n)\Big)$

- If the solved statement is of type 1 or 2, the proof is similar to the reasoning done in Lemma 4.8.
- A solved statement of type 3 can be used to solve an antecedent of the form $\mathsf{k}(X,t)$ when t is headed with the symbol plus. It is easy to see (since we already know the form of the deduction statements) that the only terms t occurring in an antecedent of an equational statement and headed with plus are ground. This allows us to conclude that the measure m_e decreases also in this case.

□

4.6.3 Going Beyond with Fair Strategies

In [2] decidability is also shown for a theory modeling homomorphic encryption. For our procedure to terminate on this theory we use a particular saturation strategy.

Homomorphic encryption. The theory $\mathsf{E}_{\mathrm{homo}}$ of homomorphic encryption that has been studied in [2, 22] is as follows:

$$\begin{gathered}\mathrm{fst}(\mathrm{pair}(x,y)) = x \quad \mathrm{snd}(\mathrm{pair}(x,y)) = y \quad \mathrm{dec}(\mathrm{enc}(x,y),y) = x\\ \mathrm{enc}(\mathrm{pair}(x,y),z) = \mathrm{pair}(\mathrm{enc}(x,z),\mathrm{enc}(y,z))\\ \mathrm{dec}(\mathrm{pair}(x,y),z) = \mathrm{pair}(\mathrm{dec}(x,z),\mathrm{dec}(y,z))\end{gathered}$$

In general, our algorithm does not terminate under this equational theory. Consider for instance the frame $\varphi = \{w_1 \mapsto \mathrm{pair}(a,b)\}$ where $a, b \in \mathcal{N}$ are private names. We have that:

$$\mathrm{Init}(\varphi) \supseteq \{\Big(\mathsf{k}(w_1,\mathrm{pair}(a,b))\Big\},\ \Big(\mathsf{k}(\mathrm{enc}(X,Y),\mathrm{enc}(x,y)) \Leftarrow \mathsf{k}(X,x),\mathsf{k}(Y,y)\Big)\}.$$

As in Example 4.11 we can obtain an unbounded number of solved statements whose projections are of the form:

$$\Big(\mathrm{pair}(\mathrm{enc}(\ldots\mathrm{enc}(a,z_1)\ldots,z_n),\mathrm{enc}(\ldots\mathrm{enc}(b,z_1)\ldots,z_n)) \Leftarrow z_1,\ldots,z_n\Big).$$

However, we can guarantee termination by using a *fair* saturation strategy. We say that a saturation strategy is fair if whenever a rule instance is enabled it will eventually be taken.

Indeed in the above example using a fair strategy we will eventually add statements whose projections are $\Big(\mathsf{k}(\mathrm{fst}(w_1),a) \Leftarrow\Big)$ and $\Big(\mathsf{k}(\mathrm{snd}(w_1),b) \Leftarrow\Big)$. Now the "problematic" statements described above become redundant and are not added to the knowledge base anymore. One may note that a fair strategy does not guarantee termination in Example 4.11 (intuitively, because the function g is one-way and a is not deducible in that example).

Lemma 4.13. *If the saturation strategy is fair the saturation process terminates for the equational theory* $\mathsf{E}_{\mathrm{homo}}$.

Proof. Orienting the five equations in $\mathsf{E}_{\mathrm{homo}}$ we obtain the following convergent term rewriting system $\mathsf{R}_{\mathrm{homo}}$:

R1 $\mathrm{fst}(\mathrm{pair}(x,y)) \to x$

R2 $\mathrm{snd}(\mathrm{pair}(x,y)) \to y$

R3 $\mathrm{dec}(\mathrm{enc}(x,y),y) \to x$

R4 $\mathrm{enc}(\mathrm{pair}(x,y),z) \to \mathrm{pair}(\mathrm{enc}(x,z),\mathrm{enc}(y,z))$

R5 $\mathrm{dec}(\mathrm{pair}(x,y),z) \to \mathrm{pair}(\mathrm{dec}(x,z),\mathrm{dec}(y,z))$

For the purpose of this proof we extend the notion of extended subterm and define ${\mathsf{st}_{ext}}^+(t)$ to be the smallest set such that:

1. $t \in {\mathsf{st}_{ext}}^+(t)$,
2. $f(t_1,\ldots,t_k) \in {\mathsf{st}_{ext}}^+(t)$ implies $t_1,\ldots,t_k \in {\mathsf{st}_{ext}}^+(t)$,
3. $t' \in {\mathsf{st}_{ext}}^+(t)$ and $t' \to t''$ implies $t'' \in {\mathsf{st}_{ext}}^+(t)$.
4. $f(t_1,\ldots,t_k) \in {\mathsf{st}_{ext}}^+(t)$ implies $f(s_1,\ldots,s_k) \in {\mathsf{st}_{ext}}^+(t)$ for every $s_i \in {\mathsf{st}_{ext}}^+(t_i)$ and for every $f \in \mathcal{F}$ of arity k.

We extend as usual ${\mathsf{st}_{ext}}^+$ to frames. We will show that the set ${\mathsf{st}_{ext}}^+(t)$ is finite for every term t. Note that $s \in {\mathsf{st}_{ext}}^+(t)$ iff $t \to^*_{\mathsf{R}_{\mathrm{homo}}\cup\{f(x_1,\ldots,x_n)\to x_i\}_{1\le i\le n}} s$ where $f \in \mathcal{F}$ ranges over all function symbols and where n is the arity of f. We will show that $\mathsf{R}_{\mathrm{homo}} \cup \{f(x_1,\ldots,x_n) \to x_i\}_{1\le i\le n}$ is terminating. Indeed, let $h(t) = \{h_1,\ldots,h_m\}$ be the multiset containing the heights of all occurrences of the function symbols enc and dec in the term t. We write $h(s) \prec h(t)$ if $h(s)$ is smaller, in the multiset extension of the usual ordering on natural numbers, than $h(t)$. We have that $t \to_{\mathsf{R}_{\mathrm{homo}}\cup\{f(x_1,\ldots,x_n)\to x_i\}_{1\le i\le n}} s$ implies $(h(s),|s|) < (h(t),|t|)$ where $<$ is the well-founded lexicographic order with the multiset ordering $\prec$ on the first component and the usual order on natural numbers on the second component. As $\mathsf{R}_{\mathrm{homo}} \cup \{f(x_1,\ldots,x_n) \to x_i\}_{1\le i\le n}$ is terminating and (obviously) finitely branching, it follows immediately that the set

$${\mathsf{st}_{ext}}^+(t) = \{s \mid t \to^*_{\mathsf{R}_{\mathrm{homo}}\cup\{f(x_1,\ldots,x_n)\to x_i\}_{1\le i\le n}} s\}$$

is finite.

Let φ be the frame being saturated. We first show that for all knowledge bases K such that $\mathrm{Init}(\varphi) \Longrightarrow^* K$ we have that each $\hat{f} \in \{\hat{f} \mid f \in K \text{ and } f \text{ is a deduction statement}\}$ has one of the following forms:

1. $\Big(t \Leftarrow \emptyset\Big)$, for some $t \in {\mathsf{st}_{ext}}^+(\varphi)$
2. $\Big(\mathrm{fst}(x) \Leftarrow x\Big)$

3. $\Big(\mathrm{snd}(x) \Leftarrow x\Big)$
4. $\Big(\mathrm{enc}(x,y) \Leftarrow x, y\Big)$
5. $\Big(\mathrm{dec}(x,y) \Leftarrow x, y\Big)$
6. $\Big(\mathrm{pair}(x,y) \Leftarrow x, y\Big)$
7. $\Big(C[t_1,\ldots,t_k] \Leftarrow \mathcal{V}ar(C)\Big)$ where:
   - C is obtained by arbitrarily nesting the following (classes of) contexts: $C_1 = \mathrm{enc}(\_, z_i)$, $C_2 = \mathrm{dec}(\_, z_i)$ and $C_3 = \mathrm{pair}(\_, \_)$, where z_i are variables.
   - C contains at least one variable.
   - $C'[t_1,\ldots,t_k] \in \mathsf{st}_{ext}{}^{+}(\phi)$, where C' is obtained from C by replacing $\mathrm{enc}(\_, z_i)$ and $\mathrm{dec}(\_, z_i)$ with $\_$.
8. $\Big(x \Leftarrow \mathrm{pair}(x,y)\Big)$
9. $\Big(y \Leftarrow \mathrm{pair}(x,y)\Big)$
10. $\Big(x \Leftarrow \mathrm{enc}(x,y), y\Big)$
11. $\Big(\mathrm{pair}(\mathrm{enc}(x,z),\mathrm{enc}(y,z)) \Leftarrow \mathrm{pair}(x,y), z\Big)$
12. $\Big(\mathrm{pair}(\mathrm{dec}(x,z),\mathrm{dec}(y,z)) \Leftarrow \mathrm{pair}(x,y), z\Big)$
13. $\Big(t \Leftarrow t_1,\ldots,t_k\Big)$, for some $t, t_1,\ldots,t_k \in \mathsf{st}_{ext}{}^{+}(\varphi)$
14. $\Big(C[t_1,\ldots,t_k] \Leftarrow s_1,\ldots,s_l, \mathcal{V}ar(C)\Big)$ where:
   - C is obtained by arbitrarily nesting the following (classes of) contexts: $C_1 = \mathrm{enc}(\_, z_i)$, $C_2 = \mathrm{dec}(\_, z_i)$, and $C_3 = \mathrm{pair}(\_, \_)$, where z_i are variables.
   - $C'[t_1,\ldots,t_k] \in \mathsf{st}_{ext}{}^{+}(\phi)$, where C' is obtain from C by replacing $\mathrm{enc}(\_, z_i)$ and $\mathrm{dec}(\_, z_i)$ with $\_$.
   - s_i are ground terms

We show this by induction on the number of saturation steps of $\mathrm{Init}(\varphi) \Longrightarrow^{*} K$. In the following when a projection $\hat{f}$ corresponds to one of the above 14 cases, we say that f is of type i $(1 \leq i \leq 14)$.
Base case. It is easy to see that all $f \in \mathrm{Init}(\varphi)$ are indeed of types $1-6$.
Inductive case. We assume that the result holds for K and we show that any possible application of a saturation rule preserves the result. We summarize the case analysis in the following two matrices.

Narrowing	R1	R2	R3	R4	R5
type 1	1	1	1	1	1
type 2	8	impossible	impossible	impossible	impossible
type 3	impossible	9	impossible	impossible	impossible
type 4	impossible	impossible	impossible	11	impossible
type 5	impossible	impossible	10	impossible	12
type 6	impossible	impossible	impossible	impossible	impossible
type 7	7	7	1, 7, 13, 14	7	7

F-Solving	type 1	type 2	type 3	type 4	type 5	type 6	type 7
type 8	1	imp.	imp.	imp.	imp.	redundant	7, 1
type 9	1	imp.	imp.	imp.	imp.	redundant	7, 1
type 10	13	imp.	imp.	imp.	redundant	imp.	7, 1
type 11	7	imp.	imp.	imp.	imp.	redundant	7
type 12	7	imp.	imp.	imp.	imp.	redundant	7
type 13	1, 13	13	13	13	13	13	13
type 14	7, 14	14	14	14	14	14	14

We next show that because the strategy is fair at a given saturation step, no more statements of type 7 are added.

Lemma 4.14. *Suppose that the saturation strategy is fair and let*

$$\text{Init}(\varphi) \Longrightarrow^* K_0 \Longrightarrow \ldots \Longrightarrow K_i \Longrightarrow \ldots$$

be a sequence of saturation steps. If $\hat{f} = \Big(C[t_1, \ldots, t_k] \Leftarrow s_1, \ldots, s_l, \mathcal{V}ar(C)\Big) \in \{\hat{f} \mid f \in K_0$ *is a deduction statement}* $\mathit{is\ of\ type\ 7\ or\ type\ 14\ and}\ \mathsf{k}(R_j, s_j) \in \mathbf{deriv}(K_0|_{\mathsf{solved}})$ *for some* R_j *for all* j*, then there exists* n *such that* $\mathsf{k}(T_i, t_i) \in \mathbf{deriv}(K_n|_{\mathsf{solved}})$ *for some* T_i *for all* i*.*

Proof. The proof is done by induction on the number of saturation steps of $\text{Init}(\varphi) \Rightarrow^* K_0$.
Base case. As $\text{Init}(\varphi)$ does not contain any statements of type 7 or 14 we conclude.
Inductive case. We suppose that the result holds for K_0 and verify that it is maintained by any possible rules that add a statement of type 7 or 14.

- Suppose we add a statement of type 7 by using rule Narrowing on a statement of type 7 in K_0 and $\mathsf{R1}$ or $\mathsf{R2}$. The rewriting must occur at a position in one of the t_i which is rewritten to t_i'. By induction hypothesis we have that there exists n, such that $\mathsf{k}(T_i, t_i) \in \mathbf{deriv}(K_n|_{\mathsf{solved}})$. We can adapt the proof of Proposition 4.2 to show that because of fairness (rather than saturation) narrowing must be applied such that there exists n' such that $\mathsf{k}(T_i', t_i') \in \mathbf{deriv}(K_{n'}|_{\mathsf{solved}})$ for some T_i'.

- Suppose we add a statement of type 7 by using rule Narrowing on a statement of type 7 in K_0 and $\mathsf{R3}$. If narrowing is applied on one of the t_i the case is similar to the previous one. If narrowing is applied inside the context such that the t_i do not change we conclude by induction hypothesis.

- Suppose we add a statement of type 14 by using rule Narrowing on a statement of type 7 in K_0 and $\mathsf{R3}$. Narrowing must have changed both the context and one of the t_i. Suppose w.l.o.g. $i = 1$. It must be that be that $t_1 = \text{enc}(t_1', t_1'')$. We have to show that there exists n such that if $\mathsf{k}(T_1'', t_1'') \in \mathbf{deriv}(K_n|_{\mathsf{solved}})$ for some T_i'' then $\mathsf{k}(T_1', t_1') \in \mathbf{deriv}(K_n|_{\mathsf{solved}})$ for some T_1' and $\mathsf{k}(T_i, t_i) \in \mathbf{deriv}(K_n|_{\mathsf{solved}})$ for some T_i for all $2 \leq i \leq k$. $\mathsf{k}(T_i, t_i) \in \mathbf{deriv}(K_n|_{\mathsf{solved}})$ is obtained by induction hypothesis. If $\mathsf{k}(T_i'', t_i'') \in \mathbf{deriv}(K_n|_{\mathsf{solved}})$ and because $\mathsf{k}(S_1, \text{enc}(t_1', t_1'')) \in \mathbf{deriv}(K_n|_{\mathsf{solved}})$ we can apply Narrowing such that $\mathsf{k}(S_1', t_1') \in \mathit{deriv}(K_{n'}|_{\mathsf{solved}})$ for some n'.

- Suppose we add a statement of type 7 by using rule Narrowing on a statement of type 7 in K_0 and $\mathsf{R4}$ (or analogously $\mathsf{R5}$). If narrowing is applied on one of the t_i the case is similar to previous cases. If narrowing is applied inside the context such that the t_i do not change we conclude by induction hypothesis. Suppose both the context and one of the t_i change. We suppose w.l.o.g. that $i = 1$. It must be that $t_1 = \text{pair}(t_1', t_1'')$. By induction hypothesis we have that there exists n such that $\mathsf{k}(T_i, t_i) \in \mathbf{deriv}(K_n|_{\mathsf{solved}})$ for some T_i for all $2 \leq i \leq k$. As $\mathsf{k}(S_1, \text{pair}(t_1', t_1'')) \in \mathbf{deriv}(K_n|_{\mathsf{solved}})$ for some S_1 we also have that $\mathsf{k}(\text{fst}(S_1), \text{fst}(\text{pair}(t_1', t_1''))) \in \mathbf{deriv}(K_n|_{\mathsf{solved}})$ and $\mathsf{k}(\text{snd}(S_1), \text{snd}(\text{pair}(t_1', t_1''))) \in \mathbf{deriv}(K_n|_{\mathsf{solved}})$. Because of fairness Narrowing can be applied such that $\mathsf{k}(\text{fst}(S_1), t_1') \in \mathbf{deriv}(K_{n'}|_{\mathsf{solved}})$ and $\mathsf{k}(\text{snd}(S_1), t_1'') \in \mathbf{deriv}(K_{n''}|_{\mathsf{solved}})$ for some n' and n''.

- Suppose we add a statement of type 7 by using rule F-Solving on statements of type 11 and 1 in K_0. Let $\mathrm{pair}(t_1, t_2)$ be the statement of type 1. As the strategy is fair we will add statements $\Big(x \Leftarrow \mathrm{pair}(x, y)\Big)$ and $\Big(y \Leftarrow \mathrm{pair}(x, y)\Big)$ by applying rule Narrowing on type 2/R1 and type 3/R2. Again by fairness we will apply solving on $\mathrm{pair}(t_1, t_2)$ and $\Big(x \Leftarrow \mathrm{pair}(x, y)\Big)$ as well as $\Big(y \Leftarrow \mathrm{pair}(x, y)\Big)$. Therefore t_1 and t_2 will be generated.

- Suppose we add a statement of type 7 by using rule F-Solving on statements of type 12 and 1 in K_0. This case is similar to the previous one.

- Suppose we add a statement of type 7 by applying rule F-Solving on statements of type 8-12 with a statement of type 7 in K_0. The resulting statement is a context on the same (or a subset of the) terms t_i $(1 \leq i \leq k)$ as the initial type 7 statement. We conclude by induction hypothesis.

- Suppose we add a statement of type 7 by applying rule F-Solving on a statement of type 14 with a statement of type 1 in K_0. The type 14 statement has only one ground antecedent s_1 which is solved by the type 1 statement. Hence $\Big(s_1\Big) \in \{\hat{f} \mid f \in K_0,\ f \text{ deduction statement}\}$ and $\mathsf{k}(S_1, s_1) \in \mathbf{deriv}(K_0|_{\mathsf{solved}})$ for some S_1. We can apply the induction hypothesis and conclude.

- Suppose we add a statement of type 14 by applying rule F-Solving on a statement of type 14 with a statement of type i $(1 \leq i \leq 14)$ in K_0. We directly conclude by induction hypothesis.

□

There are a finite number of solved statements other than of type 7. There exist only a finite number of t_i which can occur in statements of type 7 as they are in ${\mathsf{st}_{ext}}^+(\varphi)$.

Hence it follows from Lemma 4.14 that for any fair saturation sequence, at some moment all new statements of type 7 become redundant and therefore are not added to the knowledge base. Therefore any fair saturation sequence only contains a finite number of solved statements.

We know that after some number n of saturation steps, no more solved deduction statements are added to the knowledge base. We now show that a finite number of unsolved statements are added after this stage. Indeed, after n iterations, as no more solved statements are added to the knowledge base, the only types of statements potentially added are 13 and 14. The antecedents of these statements contain only ground terms or variables. By solving one of the ground antecedents the cardinality of the antecedents decreases ensuring termination.

We show that all equational statements are of the form $\Big(\mathsf{i}(M, N) \Leftarrow \mathsf{k}(X_1, t_1), \ldots, \mathsf{k}(X_k, t_k)\Big)$, for some M, N where either $t_i \in \mathcal{X}$ or $t_i = C[s_1, \ldots, s_l]$ for some ground terms s_j $(1 \leq j \leq l)$ and for some context C obtained by arbitrary nesting of contexts $C_1 = \mathrm{enc}(\_, z_n)$, $C_2 = \mathrm{dec}(\_, z_n)$, $C_3 = \mathrm{pair}(\_, \_)$ and $C_4 = \_$, where z_n are variables.

This is true for the equational statements obtained by rule Unifying. When applying rule E-Solving on an antecedent of the above type we consider the following cases:

- if we solve $\mathsf{k}(X_i, t_i)$ with a type 1 statement, we easily conclude;

- if we solve $\mathsf{k}(X_i, t_i)$ with a statement of type 2, 3, 4, 5, 6, the result is immediate;

- if we solve $\mathsf{k}(X_i, t_i)$ (where $t_i = C[s_1, \ldots, s_l]$) with a type 7 statement $\Big(C'[u_1, \ldots, u_m] \Leftarrow \mathcal{V}ar(C')\Big)$, we note that $\mathsf{mgu}(t_i, C'[u_1, \ldots, u_m])$ is such that variables are mapped to either variables or ground terms. Therefore the property holds.

Using again the measure

$$\mathrm{m}_e(\Big(\mathsf{i}(M,N) \Leftarrow \mathsf{k}(X_1,t_1),\ldots,\mathsf{k}(X_k,t_k)\Big)) = (|\mathcal{V}ar(t_1,\ldots,t_k)|, |t_1|+\ldots+|t_k|)$$

and the lexicographic order $<_e$ on pairs, we obtain that $f_0 <_e f_1$ for all f_0 and f_1 as in rule E-Solving.

□

4.7 Tool Support

With certain optimizations described below, our saturation algorithm runs in polynomial time for subterm convergent equational theories, $\mathsf{E}_{\mathrm{mal}}$, $\mathsf{E}_{\mathrm{blind}}$, and $\mathsf{E}_{\mathrm{tdcommit}}$.

4.7.1 Optimizations

Optimization 4.1 (Deciding derivations in polynomial time)). To decide if $\mathsf{k}(R,t) \in \mathbf{deriv}(K|_{\mathsf{solved}})$, the recursive algorithm obtained immediately from the **deriv** rules is not polynomial. However, by using memoization, its complexity becomes polynomial. Using the same trick, we can compute a recipe R such that $\mathsf{k}(R,t) \in \mathbf{deriv}(K|_{\mathsf{solved}})$ in polynomial time, if we store R in DAG form.

Optimization 4.2 (Recipes in DAG form). Indeed, recipes might grow to an exponential size if they are not stored in DAG form. Therefore, we require that the term R in $\Big(\mathsf{k}(R,u) \Leftarrow \Delta\Big)$ and the terms U and V in $\Big(\mathsf{k}(U,V) \Leftarrow \Delta\Big)$ are stored in DAG form.

Optimization 4.3 (Optimization to solve ground antecedents). Using different combinations of solved statements to solve ground antecedents is unnecessary work. Therefore we consider that the standard F-Solving and E-Solving rules are applied only when the antecedent being solved contains at least one variable. To solve an antecedent of the form $\mathsf{k}(X,t)$ when t is ground, we use the two rules described below. Again, as for $\oplus$, we suppose that the choice of recipes N and M is uniform.

F-Solving'

$$\frac{f_1 = \Big(\mathsf{k}(M,t) \Leftarrow \mathsf{k}(X_0,t_0),\ldots,\mathsf{k}(X_k,t_k)\Big),\ \mathcal{V}ar(t_0)=\emptyset \quad \mathsf{k}(N,t_0) \in \mathbf{deriv}(K|_{\mathsf{solved}}),\ \mathcal{V}ar(N)\cap\mathcal{V}ar(f_1)=\emptyset}{(K, K|_{\mathsf{solved}}) \Longrightarrow K \oplus f_0}$$

where $f_0 = \Big(\mathsf{k}(M\{X_0 \mapsto N\},t) \Leftarrow \mathsf{k}(X_1,t_1),\ldots,\mathsf{k}(X_k,t_k)\Big)$.

E-Solving'

$$\frac{f_1 = \Big(\mathsf{i}(U,V) \Leftarrow \mathsf{k}(Y,s),\mathsf{k}(X_1,t_1),\ldots,\mathsf{k}(X_k,t_k)\Big) \in K|_{\mathsf{solved}},\ \mathcal{V}ar(s)=\emptyset \quad \mathsf{k}(M,s) \in \mathbf{deriv}(K|_{\mathsf{solved}}),\ \mathcal{V}ar(M)\cap\mathcal{V}ar(f_1)=\emptyset}{(K, K|_{\mathsf{solved}}) \Longrightarrow K \cup \{f_0\}}$$

where $f_0 = \Big(\mathsf{i}(U\{Y \mapsto M\}, V\{Y \mapsto M\}) \Leftarrow \{\mathsf{k}(X_i,t_i)\}_{1\le i\le k}\Big)$.

Figure 4.3: Optimized saturation rules for solving ground antecedents

The soundness of this optimization is assured by Lemma 4.15 (whose proof is immediate) whereas completeness is shown by proving Lemma 4.4 and Lemma 4.7 in the context of the new saturation rules.

Lemma 4.15 (soundness of the two additional rules)**.** *Let φ be a frame and let K be a saturated knowledge base such that every statement in K holds in φ. Let f_1 and f_0 be two statements as in rules* F-Solving' *(resp.* E-Solving'*). If f_1 holds in φ then f_0 holds in φ.*

Lemma 4.4. *Let K be a saturated knowledge base and $f = \Big(\mathsf{i}(U,V) \Leftarrow \mathsf{k}(X_1,t_1),\ldots,\mathsf{k}(X_k,t_k)\Big)$ be an equational statement in K. For any substitution σ grounding for $\{t_1,\ldots,t_k\}$ such that $\mathsf{k}(R_i',t_i\sigma) \in \mathbf{deriv}(K|_{\mathsf{solved}})$ for some R_i' for all $1 \leq i \leq k$, we have that $\mathsf{k}(R_i,t_i\sigma) \in \mathbf{deriv}(K|_{\mathsf{solved}})$ for some R_i for all $1 \leq i \leq k$ and $K|_{\mathsf{solved}} \models \mathsf{i}(U\tau,V\tau)$ where $\tau = \{X_1 \mapsto R_1,\ldots,X_k \mapsto R_k\}$.*

Proof. By induction on $\sum_{i=1}^{k} |t_i\sigma|$. We distinguish two cases:

1. *f is a solved equational statement.* The proof is as before.
2. *f is an unsolved equational statement.* In such a case, there exists t_j such that $t_j \notin \mathcal{X}$. Let us assume w.l.o.g. that $j = 1$. If t_1 is not ground, then the proof is as before.

   If t_1 is ground and because K is saturated,

   $$f_2 = \Big(\mathsf{i}(U\{X_1 \mapsto M\}, V\{X_1 \mapsto M\}) \Leftarrow \mathsf{k}(X_2,t_2),\ldots,\mathsf{k}(X_k,t_k)\Big)$$

   must be in $K|_{\mathsf{solved}}$ by rule E-Solving', where M is such that $\mathsf{k}(M,t_1) \in \mathbf{deriv}(K|_{\mathsf{solved}})$.

   We can apply the induction hypothesis on the statement f_2 and the same substitution σ to obtain that there exist R_i $(i \geq 2)$ such that $\mathsf{k}(R_i,t_i\sigma) \in \mathbf{deriv}(K|_{\mathsf{solved}})$ and:

   $$K|_{\mathsf{solved}} \models \mathsf{i}(U,V)\{X_1 \mapsto M\}\{X_2 \mapsto R_2,\ldots,X_k \mapsto R_k\}.$$

   We chose $R_1 = M$ and we immediately obtain the conclusion.

□

Lemma 4.7. *Let K be a saturated knowledge base. Let $f = \Big(\mathsf{k}(R,t) \Leftarrow \mathsf{k}(X_1,t_1),\ldots,\mathsf{k}(X_k,t_k)\Big)$ be a deduction statement such that $K \oplus f = K$. For any substitution σ grounding for $\{t_1,\ldots,t_k\}$ such that $\mathsf{k}(T_i',t_i\sigma) \in \mathbf{deriv}(K|_{\mathsf{solved}})$ for all $1 \leq i \leq k$, we have that there exist $R_1,\ldots,R_k$ and W such that*

- *$\mathsf{k}(W,t\sigma) \in \mathbf{deriv}(K|_{\mathsf{solved}})$, and $\mathsf{k}(R_i,t_i\sigma) \in \mathbf{deriv}(K|_{\mathsf{solved}})$ for all $1 \leq i \leq k$;*
- *$K|_{\mathsf{solved}} \models \mathsf{i}(W, R\{X_1 \mapsto R_1,\ldots,X_k \mapsto R_k\})$.*

Proof. By induction on $\sum_{i=1}^{k} |t_i\sigma|$. We distinguish two cases. If f is solved, the proof is as before.

If f is not solved, there exists j such that $t_j \notin \mathcal{X}$. We assume w.l.o.g. that $j = 1$. If t_1 contains at least one variable, the proof is as before.

Otherwise, if t_1 is ground and because K is saturated, rule F-Solving' must have been applied and therefore we can apply the induction hypothesis on

$$f_2 = \Big(\mathsf{k}(R\{X_1 \mapsto N\},t) \Leftarrow \mathsf{k}(X_2,t_2),\ldots,\mathsf{k}(X_k,t_k)\}\Big)$$

(where N is such that $\mathsf{k}(N,t_1) \in \mathbf{deriv}(K|_{\mathsf{solved}})$) and on the same substitution σ to obtain that there exist R_i $(i \geq 2)$ and W such that

- $\mathsf{k}(W,t\sigma) \in \mathbf{deriv}(K|_{\mathsf{solved}})$ and $\mathsf{k}(R_i,t_i\sigma) \in \mathbf{deriv}(K|_{\mathsf{solved}})$, for $2 \leq i \leq k$
- $K|_{\mathsf{solved}} \models \mathsf{i}(R\{X_1 \mapsto N\}\{X_2 \mapsto R_2,\ldots,X_k \mapsto R_k\}, W)$

We choose $R_1 = N$ and we immediately obtain our conclusion.

□

4.7.2 Complexity

Theorem 4.2. *Using the optimizations 4.3, 4.2 and 4.1 described above, and if φ is in normal form, the saturation algorithm terminates in polynomial time for any subterm convergent equational theory, for $\mathsf{E}_{\text{tdcommit}}$, for E_{mal} and for $\mathsf{E}_{\text{blind}}$.*

In the remaining, we consider an equational theory E that is either subterm convergent, or $\mathsf{E} \in \{\mathsf{E}_{\text{mal}}, \mathsf{E}_{\text{blind}}, \mathsf{E}_{\text{tdcommit}}\}$. We define the following set:

$$\mathcal{Q}(\varphi) = \{\Big(r\sigma \Leftarrow t_1, \ldots, t_k\Big)\}$$

for every rewrite rule $l \to r$, for every partial substitution $\sigma : \mathcal{V}ar(l) \to \mathsf{st}_{ext}(\varphi)$ and for every set of incomparable positions $p_1, \ldots, p_k \in \mathcal{P}os(l)$ such that for every i $(1 \leq i \leq k)$ we have that $t_i = (l|_{p_i})\sigma$.

In order to prove Theorem 4.2, we need an additional lemma.

Lemma 4.16. *Let φ be a frame and let K be such that $\text{Init}(\varphi) \Longrightarrow^* K$. For any unsolved deduction statement $f \in K$ we have that $\hat{f} \in \mathcal{Q}(\varphi)$.*

Proof. First, note that an unsolved deduction statement obtained by applying NARROWING on a solved statement satisfies this property. Now assume we have an unsolved deduction statement $\hat{f} = \Big(r\sigma \Leftarrow (l|_{p_1})\sigma, \ldots, (l|_{p_k})\sigma\Big) \in \mathcal{Q}(\varphi)$ and assume one of its antecedents $(l|_{p_i})\sigma$ is being solved. Assume w.l.o.g. that $i = 1$.

- If $(l|_{p_1})\sigma$ is ground, rule F-SOLVING' must be applied. We therefore obtain a statement $\hat{f'} = \Big(r\sigma \Leftarrow (l|_{p_2})\sigma, \ldots, (l|_{p_k})\sigma\Big)$.
- If $(l|_{p_1})\sigma$ is not ground, rule F-SOLVING is applied and $l|_{p_1}$ is necessarily not a variable (by the definition of σ, it maps variables only to ground terms). Therefore $l|_{p_1}$ is of the form $g(s_1, \ldots, s_l)$ for some function symbol $g \in \mathcal{F}$. We distinguish three cases:
  - If the antecedent is solved using a deduction statement whose projection is of the form $\Big(t \Leftarrow\Big)$ for some $t \in \mathsf{st}_{ext}(\varphi)$, let $\sigma' = \mathsf{mgu}((l|_{p_1})\sigma, t)$ and consider $\tau = \sigma\sigma'$. By rule F-SOLVING, the antecedent $(l|_{p_1})\sigma$ will simply be removed.
  - If the antecedent is solved using a statement whose projection is $\Big(g(x_1, \ldots, x_l) \Leftarrow x_1, \ldots, x_l\Big)$, then the antecedent $(l|_{p_1})\sigma$ will be replaced by antecedents $(l|_{p_1 \cdot j})\sigma$, for $1 \leq j \leq l$.
  - If the antecedent is solved using a "special" statement $\Big(\text{sign}(t, x) \Leftarrow x\Big)$ (with $t \in \mathsf{st}_{ext}(\varphi)$), $\Big(\text{enc}(x, t) \Leftarrow x\Big)$ (with $t \in \mathsf{st}_{ext}(\varphi)$), $\Big(\text{tdcommit}(t_1, t_2, t_3) \Leftarrow\Big)$ (with $t_1, t_2, t_3 \in \mathsf{st}_{ext}(\varphi)$) or $\Big(\text{f}(t_1, t_2, t_3, x) \Leftarrow x\Big)$ (with $t_1, t_2, t_3 \in \mathsf{st}_{ext}(\varphi)$), we obtain by a case-by-case analysis that the property is satisfied by the resulting statement.

□

Now, we are able to prove Theorem 4.2

Proof. (of Theorem 4.2)

We first show that any knowledge base contains a polynomial number of deduction statements. Indeed, there are a polynomial number of solved deduction statements. Furthermore, from Lemma 4.11, Lemma 4.10, Lemma 4.9 and Lemma 4.8, we can deduce that the term in the head of each such solved deduction statement has polynomial size. As each NARROWING rule is applied at most once for each rewrite rule and each position of the term in the head of a solved deduction

statement, applying rule NARROWING yields a polynomial number of unsolved deduction statements. We also know, thanks to Lemma 4.16, that for any frame φ (in normal form), for any knowledge base K reachable from $\text{Init}(\varphi)$, and for any unsolved statement $f \in K$, we have that $\hat{f} \in \mathcal{Q}(\varphi)$.

We consider the two following orders:

- the order $<_p$ defined on sets of positions as follows:

  $\{p_0, \ldots, p_\ell\} <_p \{q_1, \ldots, q_k, p_1, \ldots, p_\ell\}$ iff $q_1, \ldots, q_k$ are incomparable positions and p_0 is a prefix of q_i $(1 \leq i \leq k)$.

- the order $<_f$ defined on deduction statements whose projection are in $\mathcal{Q}(\varphi)$:

  $$f_0 <_f f_1 \text{ iff either } \ell < k \text{ or } \ell = k \text{ and } \{p_1, \ldots, p_k\} <_p \{p'_1, \ldots, p'_\ell\}.$$

  where $f_0 = \Big(\mathsf{k}(R, r\sigma) \Leftarrow \mathsf{k}(X_1, l|_{p_1}\sigma), \ldots, \mathsf{k}(X_k, l|_{p_k})\Big)$, and where $f_1 = \Big(\mathsf{k}(R', r\sigma') \Leftarrow \mathsf{k}(X_1, l|_{p'_1}\sigma'), \ldots, \mathsf{k}(X_l, l|_{p'_\ell}\sigma')\Big)$.

As $<_f$ does not depend on the frame, all strictly decreasing sequences of deduction statements have at most a constant size. Also note that if f_1 and f_0 are as in rule F-SOLVING or F-SOLVING', we have that $f_0 <_f f_1$.

There is at most a polynomial number of choices to be made when solving each deduction statement (which antecedent, which solved deduction statement). As the resulting statements will be smaller (according to $<_f$) than the initial statement, and as any such sequence has at most a constant length, an unsolved statement will generated at most a polynomial number of statements.

We now show that each deduction statement has at most a polynomial size if the recipes are stored in DAG form. This is obviously true of the initial statements. The other recipes are obtained from the initial recipes by applying a polynomial number of substitutions whose size is polynomially bounded. Therefore all recipes have polynomial size.

It remains to show that there are a polynomial number of equational statements. This is true of the (necessarily solved) equational statements added during application of NARROWING and F-SOLVING (via the $\oplus$ operation). The other possibility to generate equational statements is UNIFYING, which generates a polynomial number of (possible unsolved) equational statements. All such unsolved equational statements have antecedents which are either ground or variables. Therefore, each such unsolved equational statement will lead to at most a polynomial number of other equational statements by applying rule E-SOLVING'.

□

4.8 Conclusion and Further Work

In this chapter, we have presented a procedure for checking static equivalence under convergent term rewriting systems. The procedure terminates and is therefore an algorithm when the term rewriting system implements one of the following (classes) of equational theories: subterm convergent, blind signatures, trapdoor commitment schemes, homomorphic encryption.

A C++ implementation of the procedures described in this paper is provided in the KISS (Knowledge IN Security protocols) tool [45].

The tool implements a partially fair saturation strategy and a uniform $\oplus$. The fairness employed by the tool is sufficient to decide the theory E_{homo}. Moreover the tool implements the optimizations described in Section 4.7.1. This makes the procedure terminate in polynomial time for subterm convergent equational theories, and the theories $\mathsf{E}_{\text{blind}}$, E_{mal} and $\mathsf{E}_{\text{tdcommit}}$.

The performances of the tool are comparable to the YAPA tool [20, 22] and on most examples the tool terminates in less than a second. In [22] a family of contrived examples is presented

to diminish the performance of YAPA, exploiting the statement that YAPA does not implement DAG representations of terms and recipes, as opposed to KiSs. As expected, KiSs indeed performs better on these examples.

Regarding termination, our procedure terminates on all examples of equational theories presented in [22]. In addition, our tool terminates on the theories $\mathsf{E}_{\mathrm{mal}}$ and $\mathsf{E}_{\mathrm{tdcommit}}$ whereas YAPA does not. In [22] a class of equational theories for which YAPA terminates is identified and it is not known whether our procedure terminates. YAPA may also terminate on examples outside this class. Hence the question whether termination of our procedures encompasses termination of YAPA is still open.

As further work, it would be interesting to extend the range of equational theories handled by the tool. In particular, associative-commutative function symbols should be handled in order to model exclusive-or. Another direction for future work is to extend the technique in order to handle the active case. This direction is taken in Chapter 5.

Chapter 5

Automated Verification of Trace Equivalence

5.1 Introduction

When security protocols are verified in a formal setting, the most usual security property that is checked is *secrecy* of some data. To check secrecy of some value s, the security protocol is modeled as a mathematical object P in some formalism such as the process algebra. Then, by using automated methods, hand-made proofs, or a combination of the two, it is proven that either the protocol preserves the secrecy of s in the chosen formalism, or we find an *attack*, i.e. a series of intruder actions that lead to the discovery of s.

Secrecy is a *trace property*, in the sense that it is sufficient to analyze each trace of the protocol in turn and check if the trace leaks s or not. Another important trace property is authentication, which is usually modeled as an implication of the form:

> Any time an event E takes place (e.g. a request for a money transfer has been sent to the bank), it must be the case that a previous event E' also took place (i.e. the credentials for the bank account were sent to the bank).

There are by now a large number of theoretical results [64, 118, 43, 42, 41, 54, 8, 32] regarding the automated verification of secrecy and authentication properties. Furthermore, a number of automated tools [63, 12, 25, 82] are available to automatically check such properties.

However, while secrecy and authentication are certainly important, they are not sufficient, in the sense that there exist security protocols of which we expect more. Many crucial security properties can only be expressed in terms of *indistinguishability* (or equivalence).

In particular, it is usually important for electronic voting protocols to preserve *vote privacy*, in the sense that an attacker (who might be one of the election organizers) should not be able to link voters to their vote. Informally, we would like to say voting protocol satisfies vote privacy if an attacker cannot find out how any given person voted. It turns out however that this requirement is too strong: for example, in an election where there is unanimity, everyone knows how everyone voted as soon as the result is announced.

Kremer, Delaune and Ryan propose another definition [72]. Let $P\{v_A \mapsto X, v_B \mapsto Y\}$ be an instance of the voting protocol where voter A voted for some candidate X (modeled by the fact that $v_A = X$) and where voter B voted for candidate Y. We can then say that the voting protocol satisfies vote privacy if the attacker cannot distinguish this situation from the situation where A and B swap their votes:

$$P\{v_A \mapsto X, v_B \mapsto Y\} \approx_t P\{v_A \mapsto Y, v_B \mapsto X\}.$$

Here the symbol $\approx_t$ denotes an *indistinguishability* relation between processes, i.e. $\approx_t$ identifies processes which *look the same* to the intruder. In this thesis, we will assume that $\approx_t$ is *trace*

equivalence, as it best models the capabilities of the standard Dolev-Yao attacker, which has complete control of the network and no control whatsoever of the machines belonging to the honest participants.

Other uses of indistinguishability include strong flavors of confidentiality [26]; resistance to guessing attacks in password based protocols [19]; and anonymity properties in private authentication protocols [4], electronic voting [72, 17], vehicular networks [65, 66] and RFID protools [9, 31]. More generally, indistinguishability allows to model security by the means of ideal systems, which are correct by construction [5, 70]. Indistinguishablity properties of cryptographic protocols are naturally modeled by the means of *observational* and *testing equivalences* in cryptographic extensions of process calculi, e.g., the spi-calculus [5] and the applied-pi calculus [3].

While we have a good number of tools [63, 12, 25, 82] for automated verification of trace properties, the situation is different for indistinguishability properties. This chapter addresses this situation by introducing a procedure for checking trace equivalence.

5.1.1 Related Work

In the case of a passive adversary, i.e. an adversary that only listens to messages on the network but cannot modify them, deciding trace equivalence reduces to checking static equivalence. As we have seen in the previous chapter, many decidability results have been obtained [2, 58, 13]. Exact [21, 51] and approximate [27] tools exist for a variety of cryptographic primitives.

The field of automated verification for equivalence properties is less well-developed. There are a few theoretical results [97, 37, 59, 27, 121] and very few tools [25, 37] that can handle equivalence properties. Undecidability of observational equivalence in the spi calculus, even for the finite control fragment, was shown by Hüttel [97]. Current results [27] allow to approximate observational equivalence for an unbounded number of sessions. However, this approximation does not suffice to conclude for many applications, such as electronic voting protocols.

For the finite, replication-free, fragment of the spi calculus a decision result [97] and an automated tool [79] exist for checking observational equivalence, but only for a fixed, limited set of cryptographic primitives. Symbolic bisimulations have also been devised for the spi [30, 29] and applied pi calculus [73, 102] to avoid unbounded branching due to adversary inputs. However, only [73] and [30] yield a decision procedure, but again only approximating observational equivalence. The results of [73] have been further refined to show a decision procedure on a restricted class of *simple* processes [59]. In particular they rely on a procedure deciding the equivalence of constraint systems, introduced by Baudet [19], for the special case of verifying the existence of guessing attacks. Baudet's procedure allows arbitrary cryptographic primitives that can be modeled as a subterm convergent rewrite systems. An alternate procedure achieving the same goal was proposed by Chevalier and Rusinowitch [44]. However, both procedures are highly nondeterministic and do not yield a reasonable algorithm which could be implemented. Therefore, Cheval *et al.* [37, 38] have designed a new procedure and a prototype tool to decide the equivalence of constraint systems, but only for a fixed set of primitives. In particular, [37] decides equivalence of two symbolic traces and an implementation is available. In [38], the same authors extend [37] to handle equivalence between nondeterminate processes with else branches, but the resulting algorithm is not (yet) implemented and is limited by the fact that it cannot treat cryptographic primitives appearing in electronic voting protocols.

Techniques based on Horn clauses have been extensively used, e.g. by Blanchet [25], Weidenbach [123] and Goubault [89], in the case of an unbounded number of sessions. However, all these approaches allow false attacks. In the case of confidentiality properties, precise encodings (without false attacks) into Horn clauses have been proposed and implemented in [6]. However, the encoding is for a fixed set of cryptographic primitives, it does not take into account that processes may block (due to a test failing) and it works only for trace properties. Another approach [74] based on resolution of clauses with *rigid* variables was proposed to solve reachability properties while avoiding false attacks.

5.1.2 Contribution

In this chapter we introduce an automated method for proving trace equivalence for a cryptographic process algebra without replication and without else branches where messages are modeled as terms equipped with an equational theory, similar to the applied pi calculus [3]. We define our process algebra in Section 5.2. Our approach is based on modeling traces as Horn clauses and can be seen as an extension of the procedure presented in Chapter 4. We give an informal overview of the procedure and we present the main difficulties in modeling traces as sets of Horn clauses in Section 5.3.

In Section 5.4, we formalize the modeling of traces into Horn clauses and in Section 5.5, we formally describe the saturation procedure applied to the set of Horn clauses. The goal of the saturation procedure is to produce a set of Horn clauses which make it easy to check trace equivalence. It is correct for arbitrary cryptographic primitives that have the strong finite variant property. We prove this by showing soundness (Section 5.6) and completeness (Section 5.7) of the saturation procedure. Due to the highly complex form of Horn clauses appearing during saturation, we were not able to prove that saturation always terminates. We conjecture that it always terminates when the rewrite system is subterm convergent.

In Section 5.5.3 we show how to decide *coarse trace equivalence* from the saturated sets of Horn clauses for the class of *determinate processes*. For this class of processes, we show (Section 5.2.3) that coarse trace equivalence coincides with trace equivalence. Furthermore, we know that for determinate processes, trace equivalence is the same as observational equivalence [59]. The proof that the algorithm given in Section 5.5.3 for deciding coarse trace equivalence for determinate processes is correct is given in Section 5.8. To overcome the limitation of determinate processes, we suggest a proof method in Section 5.9 for showing trace equivalence for processes which are not known to be determinate. Using this proof method, we give an automated proof of vote privacy for the electronic voting protocol FOO in Section 5.9.3.

We have implemented the saturation procedure and the trace equivalence checking algorithm in the tool AKiSs [46]. We describe our implementation in Section 5.9. We conclude and present directions for further work in Section 5.10. This chapter is based on an article that has been published in [35].

5.2 Process Algebra

In order to formally assess the security of cryptographic protocols, it is necessary to model the protocols and the expected security properties as mathematical objects. Cryptographic process calculi [5, 3] are a formalism derived from generic process calculi [108] that are designed specifically for modelling cryptographic protocols. In this thesis, we introduce a version of the applied pi calculus [3] that is particularly ammenable to automated analysis and to composability issues discussed in this thesis. Recall that we have fixed a first-order signature $\mathcal{F}$, a set $\mathcal{N}$ of *private names*, a set $\mathcal{M}$ of *public names*, a set $\mathcal{C}$ of public channels, and a set $\mathcal{W}$ of *parameters*. Processes are built from atomic building blocks that we call actions:

5.2.1 Actions

Definition 5.1. An action a is a term in $T(\mathcal{F} \cup \{\mathbf{receive}, \mathbf{send}, \stackrel{?}{=}\}, \mathcal{C}, \mathcal{N}, \mathcal{M}, \mathcal{X})$ of one of the following types:

$$\begin{array}{rcll} a & = & \mathbf{receive}(c, x) & \text{(receive action) or} \\ a & = & \mathbf{send}(c, t) & \text{(send action) or} \\ a & = & [s \stackrel{?}{=} t] & \text{(test action)} \end{array}$$

where $\mathbf{receive}, \mathbf{send}, \stackrel{?}{=} \notin \mathcal{F}$ are fresh function symbols of arity 2, where $c \in \mathcal{C}$ is a public channel and where $s, t \in T(\mathcal{F}, \mathcal{N}, \mathcal{M}, \mathcal{X})$. The function symbol $\stackrel{?}{=}$ is used with infix syntax for clarity:

$s \stackrel{?}{=} t$ is the same as $\stackrel{?}{=}(s,t)$. The square brackets around the $s \stackrel{?}{=} t$ action are just for presentation reasons and are not part of the action.

The action $\mathbf{receive}(c,x)$ intuitively denotes the fact that a message is expected on channel c. When a message is received, it is bound to the variable x. The action $\mathbf{send}(c,t)$ represents an agent that sends out a message t on the cannel c. The action $[s \stackrel{?}{=} t]$ blocks if the terms s and t represent different messages and executes successfully otherwise.

5.2.2 Traces

During a protocol run, actions are performed concurrently by the agents running the protocol.

Definition 5.2. A *trace* T is a finite sequence of actions $T = a_1.a_2.\ldots.a_n$.

Intuitively, $T = a_1.a_2.\ldots.a_n$ represents an agent which first performs the action a_1, then a_2, etc. As usual, a receive action $a_i = \mathbf{receive}(c,x)$ acts as a *binding construct* for the variable x in the rest of the trace $a_{i+1},\ldots,a_n$. We assume the usual definitions of free and bound variables for traces. We also identify traces which are equivalent up to α-renaming of bound variables.

If T is a trace with free variables $x_1,\ldots,x_n$, we write $T(x_1,\ldots,x_n)$ instead of T. As is usual, $T(m_1,\ldots,m_n)$ will then represent the trace T where the terms $m_1,\ldots,m_n$ are substituted for variables $x_1,\ldots,x_n$ such that variable-capturing is avoided.

> *Example* 5.1. Let c be a public channel name, let $k \in \mathcal{N}$ be a private name representing a key and let
>
> $$T(x) = \mathbf{send}(c, \mathrm{enc}(x,k)).\mathbf{send}(c,k).$$
>
> The trace $T(x)$ can represent the sequence of actions performed by a voter that wishes to vote x.

Semantics of traces

As is usual in security protocol analysis, we assume that all interactions between participants are mediated by the attacker. Therefore, whenever a participant is ready to send a message on a public channel, the attacker will be the one receiving the message, thereby increasing its knowledge. Similarly, every time an agent is ready to input a message from a public channel, it is the attacker that provides this message; the attacker will build it from its previous knowledge.

This fact is modeled in the operational semantics of a trace. A trace executes in the presence of a frame representing the messages that have already been exchanged on the network. If the term $l \in \{\mathbf{receive}(c,r), \mathbf{send}(c), \mathbf{test}\}$ is over the signature $\mathcal{F} \uplus \{\mathbf{receive}, \mathbf{send}, \mathbf{test}\}$ (with $c \in \mathcal{C}$ being a channel name and $r \in T(\mathcal{F}, \mathcal{W} \cup \mathcal{M})$ being a recipe), the relation $\xrightarrow{l}_{\mathcal{F},\mathsf{E}}$ denotes one execution step, and is defined as follows:

$$\textsc{Receive}\ \frac{\varphi \vdash^{r}_{\mathcal{F},\mathsf{E}} t}{(\mathbf{receive}(c,x).T, \varphi) \xrightarrow{\mathbf{receive}(c,r)}_{\mathcal{F},\mathsf{E}} (T\{x \mapsto t\}, \varphi)}$$

$$\textsc{Send}\ \frac{}{(\mathbf{send}(c,t).T, \varphi) \xrightarrow{\mathbf{send}(c)}_{\mathcal{F},\mathsf{E}} (T, \varphi \cup \{w_{|\mathcal{D}om(\varphi)|+1} \mapsto t\})}$$

$$\textsc{Test}\ \frac{s =_{\mathsf{E}} t}{([s \stackrel{?}{=} t].T, \varphi) \xrightarrow{\mathbf{test}}_{\mathcal{F},\mathsf{E}} (T, \varphi)}$$

When the signature $\mathcal{F}$ or the equational theory E are clear from context, we can drop them from the subscript and write $\xrightarrow{l}$, $\xrightarrow{l}_{\mathsf{E}}$ or $\xrightarrow{l}_{\mathcal{F}}$ instead of $\xrightarrow{l}_{\mathcal{F},\mathsf{E}}$.

The labels on the "$\rightarrow_{\mathcal{F},\mathsf{E}}$" arrow denote intuitively the actions performed by the attacker. If the trace expects a message on channel c, the intruder will construct using recipe r a term t from the terms currently known to the intruder (the terms in φ). Then the receive action is executed, the variable to which the message is bound being replaced in the rest of the trace by the message itself. The intruder knowledge does not increase and therefore no new term is added to the frame φ.

If the trace performs a send action, the message being sent goes to the intruder and is therefore added to the frame, modeling the fact that the intruder knowledge has grown. If the trace performs a test action, the test must be successful in order for the trace to continue. No new message is added to the frame in this case. If the test is not successful, the trace *blocks*. We write $\xrightarrow{l_1,\ldots,l_n}_{\mathcal{F},\mathsf{E}}$ for $\xrightarrow{l_1}_{\mathcal{F},\mathsf{E}}\xrightarrow{l_2}_{\mathcal{F},\mathsf{E}}\ldots\xrightarrow{l_n}_{\mathcal{F},\mathsf{E}}$ (where juxtaposition denotes composition of relations).

Example 5.2. Continuing the previous example ($T(x) = \mathbf{send}(c, \mathrm{enc}(x, k)).\mathbf{send}(c, k)$), we have that

$$(T(yes), \emptyset) \xrightarrow{\mathbf{send}(c),\mathbf{send}(c)} (0, \varphi)$$

where 0 denotes the empty trace (with 0 actions) and where $\varphi = \{w_1 \mapsto \mathrm{enc}(yes, k), w_2 \mapsto k\}$.

As **send** and **receive** actions are mediated by the intruder, they are *visible* to the intruder. However, test actions **test** are internal to the trace and are not visible to the intruder. As is usual in process calculi, we define a weak semantics to account for the lack of visibility by the intruder of **test** actions. In this semantics **test** actions can be performed spontaneously or skipped. This weak semantics is defined by the relation $\Rightarrow$. We shall write $(T, \varphi) \stackrel{l}{\Rightarrow} (T', \varphi')$ if

1. $(T, \varphi) \xrightarrow{\mathbf{test}^*,l,\mathbf{test}^*} (T', \varphi')$ when $l \neq \mathbf{test}$ and
2. $(T, \varphi) \xrightarrow{\mathbf{test}^*} (T', \varphi')$ when $l = \mathbf{test}$,

where $\mathbf{test}^*$ denotes an arbitrary number of **test** actions.

5.2.3 Processes

Traces describe actions executing sequentially in a protocol. However, in a security protocol, a different sequences of actions might be performed depending on the choices of agents and of the intruder. To represent this, we define processes to consist of any number of traces.

Definition 5.3. A *process* P is a set (possibly infinite) of traces.

The set of bound variables (resp. the set of free variables) of a process is the union of the sets of bound variables (resp. free variables) of all traces in the process. We identify processes which are equal up to α-renaming of bound variables. If $\{x_1, \ldots, x_n\}$ is the set of free variables of a process P, we write $P(x_1, \ldots, x_n)$ instead of P. In this case $P(m_1, \ldots, m_n)$ denotes the process P where the free variables $x_1, \ldots, x_n$ are replaced by the corresponding terms $m_1, \ldots, m_n$ in a variable-capture avoiding way. We identify singleton processes (consisting of a single trace) with the trace itself.

We write $(P, \emptyset) \xrightarrow{l_1,\ldots,l_n} (T, \varphi)$ (resp. $(P, \emptyset) \stackrel{l_1,\ldots,l_n}{\Longrightarrow} (T, \varphi)$) if there exists $S \in P$ such that $(S, \emptyset) \xrightarrow{l_1,\ldots,l_n} (T, \varphi)$ (resp. $(S, \emptyset) \stackrel{l_1,\ldots,l_n}{\Longrightarrow} (T, \varphi)$).

Therefore, a process behaves as any one of its traces. This definition is non-standard, in the sense that cryptographic process calculi usual contain a parallel operator "|" ($P \mid Q$ denoting two processes executing concurrently), a non-deterministic operator "+" ($P + Q$ denoting a process which executes either as P or as Q) and a replication operator "!" ($!P$ denoting the process that executes as an infinite number of copies of P in parallel).

However it is easy to see that any process defined using the parallel composition operator, the non-deterministic choice operator and the replication operator is trace equivalent to a process in

our calculus (obtained by considering all possible interleavings of actions for each of the operators). Therefore, from a theoretical point of view, this choice of representation of processes is not a restriction, as illustrated by the following example:

Example 5.3. We consider a toy voting protocol where each agent sends the vote (either 'yes' or 'no'), encrypted with a previously distributed key k on a public channel. We assume without modeling this that a trusted authority will then collect all votes, decrypt them and announce the result.

We consider the trace $S = \mathbf{send}(c, \mathrm{enc}(yes, k)).\mathbf{send}(c, k)$ run by an agent voting 'yes' and then accidently (or maliciously) sending the key k on the network. The trace $T = \mathbf{send}(c, \mathrm{enc}(no, k))$ is that of an agent voting 'no'. Then the following process

$$P = \{ \begin{array}{l} \mathbf{send}(c, \mathrm{enc}(yes, k)).\mathbf{send}(c, k).\mathbf{send}(c, \mathrm{enc}(no, k)), \\ \mathbf{send}(c, \mathrm{enc}(yes, k)).\mathbf{send}(c, \mathrm{enc}(no, k)).\mathbf{send}(c, k), \\ \mathbf{send}(c, \mathrm{enc}(no, k)).\mathbf{send}(c, \mathrm{enc}(yes, k)).\mathbf{send}(c, k) \end{array} \}$$

consisting of three traces represents all possible concurrent runs of S and T running in parallel.

However, in practice, the representation of a process as a collection of traces might be exponentially larger than a representation using the usual operators. We choose to work with a collection of traces since the procedure that we describe in this chapter is easier to state on traces. However, we note that before applying this procedure to a security protocol, an exponential blow-up could occur due to the change in representation.

Determinate processes

An important class of processes is the class of *determinate* processes. The class of determinate processes was first introduced by Engelfriet [81] for the π-calculus [108]. He showed that trace equivalence coincides with observational equivalence for this class of processes. Cortier and Delaune [59] extended the result of Engelfriet to the applied π-calculus [3]. As the core of our procedure shows trace equivalence for determinate process, we need to formally define determinate processes for our calculus.

Definition 5.4. A process P is *determinate* if whenever $(P, \emptyset) \xRightarrow{l_1,\ldots,l_n} (T, \varphi)$ and $(P, \emptyset) \xRightarrow{l_1,\ldots,l_n} (T', \varphi')$ then $\varphi \approx_s \varphi'$.

Intuitively this means that whenever the intruder performs a sequence of visible actions, its knowledge is completely determined up to static equivalence by the sequence of actions that it performed. The following is immediate from the definition.

Proposition 5.1. *A trace (i.e. a process consisting of a single trace) is determinate.*

5.2.4 Trace Equivalence

Security properties such as vote privacy can be defined in terms of equivalences between processes. We recall the standard definition of trace equivalence.

Definition 5.5 (Trace equivalence)**.** A process P is said to be *trace-included* in a process Q (written $P \sqsubseteq_t Q$) if whenever $(P, \emptyset) \xRightarrow{l_1,\ldots,l_n} (T, \varphi)$ there exist T', φ' such that $(Q, \emptyset) \xRightarrow{l_1,\ldots,l_n} (T', \varphi')$ and $\varphi \approx_s \varphi'$. Two processes P and Q are *trace-equivalent* (written $P \approx_t Q$) if $P \sqsubseteq_t Q$ and $Q \sqsubseteq_t P$.

Coarse Trace Equivalence and Determinate Processes

We now introduce an equivalence between processes that we call *coarse trace equivalence* and that is in general weaker than trace equivalence: two processes which are in trace equivalence are necessarily in coarse trace equivalence but not vice-versa. The reason for introducing this equivalence is that it is easier to reason about. Furthermore, we show here that the two notions of equivalence coincide for determinate processes.

Definition 5.6. Given two processes P and Q, we say that P is *coarse trace included* in Q (and we write $P \sqsubseteq_c Q$) if whenever $(P, \emptyset) \stackrel{l_1,\ldots,l_n}{\Longrightarrow} (T, \varphi)$ there exist T', φ' such that $(Q, \emptyset) \stackrel{l_1,\ldots,l_n}{\Longrightarrow} (T', \varphi')$ and $\varphi \sqsubseteq_s \varphi'$. We say that P is *coarse trace equivalent* to P (and we write $P \approx_c Q$) if $P \sqsubseteq_c Q$ and $Q \sqsubseteq_c P$.

It is easy to see from the above definition that $P \approx_t Q$ implies $P \approx_c Q$ for any processes P and Q. The following example show that the reverse implication is not always true.

Example 5.4. Let $a, b \in \mathcal{N}$ be private names, let $c \in \mathcal{C}$ be a public channel and let

$$P = \{\mathbf{send}(c, a).\mathbf{send}(c, a)\}, Q = \{\mathbf{send}(c, a).\mathbf{send}(c, a), \mathbf{send}(c, a).\mathbf{send}(c, b)\}.$$

Clearly $P \sqsubseteq_c Q$. Observe also that $Q \sqsubseteq_c P$. This is because $\{w_1 \mapsto a, w_2 \mapsto b\} \sqsubseteq_s \{w_1 \mapsto a, w_2 \mapsto a\}$. Thus, $P \approx_c Q$. But $P \not\approx_t Q$.

The following theorem shows that the reverse implication is true if P and Q are determinate.

Theorem 5.1. .
If P and Q are determinate processes then $P \approx_t Q$ iff $P \approx_c Q$.

Proof. Let P and Q be determinate processes.

($\Rightarrow$) Follows immediately from definition of $\approx_t$ and $\approx_c$.

($\Leftarrow$) We need to show that that $P \approx_c Q$ implies $P \approx_t Q$. We proceed by contradiction. Suppose that $P \approx_c Q$ and $P \not\approx_t Q$. We suppose $P \not\sqsubseteq_t Q$ (the case of $Q \not\sqsubseteq_t P$ being symmetric). As $P \not\sqsubseteq_t Q$ we have that there exist $l_1, \ldots, l_n$, T, φ, such that $(P, \emptyset) \stackrel{l_1,\ldots,l_n}{\Longrightarrow} (T, \varphi)$ and

1. either there exist no φ', T' such that $(Q, \emptyset) \stackrel{l_1,\ldots,l_n}{\Longrightarrow} (T', \varphi')$,
2. or for all φ', T' such that $(Q, \emptyset) \stackrel{l_1,\ldots,l_n}{\Longrightarrow} (T', \varphi')$ we have that $\varphi \not\approx_s \varphi'$.

In the first case, $P \not\approx_c Q$, contradicting our hypothesis. In the second case, as $\varphi \not\approx_s \varphi'$, there exist r, r' such that $(r = r')\varphi$ and $(r \neq r')\varphi'$ (or vice-versa, the other case is symmetric). As $P \sqsubseteq_c Q$, we have that there exist T'', φ'' such that $(Q, \emptyset) \stackrel{l_1,\ldots,l_n}{\Longrightarrow} (T'', \varphi'')$ and $\varphi \sqsubseteq_s \varphi''$. Hence, we have $(r = r')\varphi''$. As Q is determinate, we have that $\varphi' \approx_s \varphi''$. This yields a contradiction, as $(r \neq r')\varphi'$ and $(r = r')\varphi''$ would imply $\varphi' \not\approx_s \varphi''$.

□

5.3 Overview and Difficulties

Having already presented in Chapter 4 a successful procedure for static equivalence, we can try to extend it to handle trace equivalence. Unfortunately, a number of difficulties arise due to the active nature of the intruder.

5.3.1 Horn Clause Modeling

The most natural extension is to define an initial knowledge base for a symbolic trace. As in the previous chapter, the initial knowledge will be a Horn theory containing Horn clauses that we call statements. Such a knowledge base would contain the "context" statements which model that the intruder can apply function symbols to known terms. It would also contain statements modeling that the intruder knows all public names.

In addition to those statements, an initial knowledge base for a symbolic trace should also contain statements which model that if the intruder is able to provide inputs that pass the tests in the process, then whatever the process outputs enters the intruder knowledge. For example, consider the symbolic trace

$$P = \mathbf{send}(c, \mathrm{enc}(a, k)).\mathbf{receive}(c, x).\mathbf{send}(c, \mathrm{dec}(x, k))$$

where a and k are private names and $c \in \mathcal{C}$ is a public channel. We could model this symbolic trace by the Horn clauses

$$\begin{array}{l} \mathsf{k}(w_1, \mathrm{enc}(a, k)) \\ \mathsf{k}(w_2, \mathrm{dec}(x, k)) \Leftarrow \mathsf{k}(X, x), \end{array}$$

where the k predicate is similar to the k predicate using for static equivalence in Chapter 4: the fact $\mathsf{k}(R, t)$ intuitively states that R can be used by the intruder as a recipe for t.

The first clause states that the intruder knows the encryption of a with k (as it is the first message sent by the process). The second clause models that if the intruder can construct a message x, then the second output of the trace (the parameter w_2) is a recipe for the decryption of x with the private name k. This allows us to conclude that a is not secret, since $\mathsf{k}(w_2, a)$ can be derived by instantiating $\mathsf{k}(X, x)$ with $\mathsf{k}(w_1, \mathrm{enc}(a, k))$ in the second Horn clause above. Unfortunately, this modeling is not accurate. Consider the following trace:

$$\begin{array}{rcl} T & = & \mathbf{send}(c, \mathrm{enc}(a, k)).\mathbf{send}(c, \mathrm{enc}(a', k)).\mathbf{receive}(c, x).\mathbf{send}(c, \mathrm{dec}(x, k)). \\ & & \mathbf{receive}(c, y).[y \stackrel{?}{=} \mathrm{pair}(a, a')].\mathbf{send}(c, s).0 \end{array}$$

where $a, a', k, s \in \mathcal{N}$ are private names and c is a public channel. This trace first outputs the encryption of a and respectively a' with the key k and then offers to make *one* decryption with the key k. The intruder can therefore choose to get either the private name a or the private name a', but not both. After the decryption, the trace verifies if the intruder is able to provide a pair containing both a and a' and, if so, outputs the private name s. The above trace would be modeled as the following set of Horn clauses:

$$\begin{array}{l} \mathsf{k}(w_1, \mathrm{enc}(a, k)) \\ \mathsf{k}(w_2, \mathrm{enc}(a', k)) \\ \mathsf{k}(w_3, \mathrm{dec}(x, k)) \Leftarrow \mathsf{k}(X, x) \\ \mathsf{k}(w_4, s) \Leftarrow \mathsf{k}(X, x), \mathsf{k}(Y, y), y =_{\mathsf{E}} \mathrm{pair}(a, a')). \end{array}$$

The above set of Horn clauses, combined with the context statement that allows the intruder to construct pairs of known terms, allows to derive the fact $\mathsf{k}(w_4, s)$. This is clearly not sound, since the protocol does not reveal s. The problem in the above modeling is that the protocol allows the intruder to decrypt *exactly one message* with the key k, while the Horn clauses allow the intruder to decrypt any number of such messages.

To solve this issue, we annotate the intruder knowledge predicate with additional parameters to take into account the *entire history of interactions* with the protocol. We write the new arguments as a subscript of k. To illustrate this, consider again the symbolic trace

$$P = \mathbf{send}(c, \mathrm{enc}(a, k)).\mathbf{receive}(c, x).\mathbf{send}(c, \mathrm{dec}(x, k))$$

where a and k are private names and $c \in \mathcal{C}$ is a public channel. We will model this trace using the following Horn clauses

$$\begin{array}{l} \mathsf{k}_{\mathbf{send}(c)}(w_1, \mathrm{enc}(a, k)) \\ \mathsf{k}_{\mathbf{send}(c), \mathbf{receive}(c,x), \mathbf{send}(c)}(w_2, \mathrm{dec}(x, k)) \Leftarrow \mathsf{k}_{\mathbf{send}(c)}(X, x). \end{array}$$

The first Horn clause now states that *after one **send** action on channel* c, the parameter w_1 points to the encryption of a with k. The second Horn clause states that if the intruder can construct a message x after one **send** action of the protocol, then after one **send** action, one receive action where x (the same x as before) is being input by the trace and another **send** action, the parameter w_2 points to the decryption of x with k. The fact that a is revealed is still modeled by the Horn clauses since $\mathsf{k}_{\mathbf{send}(c),\mathbf{receive}(c,\mathrm{enc}(a,k)),\mathbf{send}(c)}(w_2, a)$ is derivable. Furthermore, we now avoid the accuracy problems. Consider again the trace

$$\begin{aligned} T \;=\; & \mathbf{send}(c,\mathrm{enc}(a,k)).\mathbf{send}(c,\mathrm{enc}(a',k)).\mathbf{receive}(c,x).\mathbf{send}(c,\mathrm{dec}(x,k)).\\ & \mathbf{receive}(c,y).[y \stackrel{?}{=} \mathrm{pair}(a,a')].\mathbf{send}(c,s).0 \end{aligned}$$

where $a, a', k, s \in \mathcal{N}$ are private names and c is a public channel. Using the new encoding, the above trace is modeled as the following set of Horn clauses:

$$\begin{aligned} & \mathsf{k}_{\mathbf{send}(c)}(w_1, \mathrm{enc}(a,k))\\ & \mathsf{k}_{\mathbf{send}(c),\mathbf{send}(c)}(w_2, \mathrm{enc}(a',k))\\ & \mathsf{k}_{\mathbf{send}(c),\mathbf{send}(c),\mathbf{receive}(c,x),\mathbf{send}(c)}(w_3, \mathrm{dec}(x,k)) \Leftarrow \mathsf{k}_{\mathbf{send}(c),\mathbf{send}(c)}(X,x)\\ & \mathsf{k}_{\mathbf{send}(c),\mathbf{send}(c),\mathbf{receive}(c,x),\mathbf{send}(c),\mathbf{receive}(c,y),\mathbf{send}(c)}(w_4, s) \Leftarrow\\ & \quad \mathsf{k}_{\mathbf{send}(c),\mathbf{send}(c)}(X,x), \mathsf{k}_{\mathbf{send}(c),\mathbf{send}(c),\mathbf{receive}(c,x),\mathbf{send}(c)}(Y,y), y =_{\mathsf{E}} \mathrm{pair}(a,a')). \end{aligned}$$

With the new encoding, we have that $\mathsf{k}_{\mathbf{send}(c),\mathbf{send}(c),\mathbf{receive}(c,\mathrm{enc}(a,k)),\mathbf{send}(c)}(w_3, a)$ is derivable and also that $\mathsf{k}_{\mathbf{send}(c),\mathbf{send}(c),\mathbf{receive}(c,\mathrm{enc}(a',k)),\mathbf{send}(c)}(w_3, a')$ is derivable. However, there exists no history H and no recipe R such that $\mathsf{k}_H(R, \mathrm{pair}(a,a'))$ is derivable. We are assuming of course that context statements only allow to apply contexts over terms that can be obtained by the intruder using the same history. For example, the context statements modeling the fact that the intruder can pair up know terms should now be

$$\mathsf{k}_W(\mathrm{pair}(X,Y), \mathrm{pair}(x,y)) \Leftarrow \mathsf{k}_W(X,x), \mathsf{k}_W(Y,y),$$

where W is a variable that matches the history and X, Y, x, y are regular variables.

A similar encoding was proposed for verifying secrecy properties [6] for a specific intruder theory. However, that encoding is not completely accurate as it does not take into account the fact that processes might block. Our encoding into Horn clauses is a *full abstraction* of the respective symbolic trace.

5.3.2 Getting Rid of Equational Antecedents

In the body of the Horn clauses above, we have introduced additional constraints of the form $u =_E v$ for some terms u, v. These antecedents appear because the process is performing some test $[u \stackrel{?}{=} v]$. Fortunately, there is an easy way to get rid of these antecedents by using *complete set of unifiers*. A complete set of unifiers modulo E of the terms u and v provides a finite symbolic representation of all instantiations of u and v which render the two terms equal modulo E. Therefore, we will simply replace a Horn clause of the form

$$C = \Big(H \Leftarrow B_1, \ldots, B_n, u =_{\mathsf{E}} V\Big)$$

with a set of Horn clauses

$$\begin{aligned} C_1 \;&=\; \Big((H \Leftarrow B_1, \ldots, B_n)\sigma_1\Big)\\ &\ldots\\ C_k \;&=\; \Big((H \Leftarrow B_1, \ldots, B_n)\sigma_k\Big). \end{aligned}$$

whenever $\sigma_1, \ldots, \sigma_k$ is a complete set of unifiers of u and v. Such a complete set of unifiers can be effectively computed using the procedure developed in Section 3.6 for any equational theory having the strong finite variant property. By the results in Chapter 3 all convergent optimally reducing equational theories have the strong finite variant property.

5.3.3 Termination

The goal of the saturation procedure is to produce by resolution a set of solved statements which is "equivalent" in some sense to the set of initial statements such that trace equivalence is easy to check from the solved set of statements. Unfortunately, if we naively extend the saturation procedure from Chapter 4, the saturation process will not terminate even in simple cases. We illustrate non-termination of saturation bellow. We consider the following subterm convergent rewrite system

$$\mathsf{R} = \{f(f(x)) \to f(x)\}$$

and the symbolic trace

$$S = \mathbf{receive}(c, x).\mathbf{send}(c, f(x)).$$

We have that the Horn clause

$$\mathsf{k}_{\mathbf{receive}(c,x),\mathbf{send}}(w_1, f(x)) \Leftarrow \mathsf{k}(X, x)$$

is in the initial knowledge base of the symbolic trace. The fact that $\mathsf{k}(X, x)$ does not have any subscript indicates that the intruder must construct x using recipe X *before any interaction* with the process, i.e. in the empty history.

As an instantiation of $f(x)$ might not be in normal form (depending on the value of x), there is a chance to perform a narrowing step on this clause using the (fresh) rewrite rule $f(f(y)) \to f(y)$. We would obtain the clause

$$\mathsf{k}_{\mathbf{receive}(c,f(y)),\mathbf{send}}(w_1, f(y)) \Leftarrow \mathsf{k}(X, f(y)).$$

The antecedent in the body of this Horn clause can be solved using a context fact and we would obtain the statement

$$\mathsf{k}_{\mathbf{receive}(c,f(y)),\mathbf{send}}(w_1, f(y)) \Leftarrow \mathsf{k}(X, y).$$

Unfortunately, at this point we can perform another narrowing step of $f(y)$ with the (fresh) rewrite rule $f(f(z)) \to f(z)$. As this would continue forever, the saturation procedure would have no chance to stop. Fortunately, there is a way to "precompute" all possible normal forms of the term $f(x)$ using the strong finite variant property described in Chapter 3 and therefore not have to apply narrowing steps on the fly. In the initial knowledge base, we will basically replace Horn clauses of the form

$$C = \Big(\mathsf{k}_H(R, t) \Leftarrow B_1, \ldots, B_n\Big)$$

by a set of clauses

$$\begin{aligned} C_1 &= \Big((\mathsf{k}_H(R, t))\sigma_1{\downarrow} \Leftarrow B_1\sigma_1{\downarrow}, \ldots, B_n\sigma_1{\downarrow}\Big) \\ &\ldots \\ C_k &= \Big((\mathsf{k}_H(R, t))\sigma_k{\downarrow} \Leftarrow B_1\sigma_k{\downarrow}, \ldots, B_n\sigma_k{\downarrow}\Big). \end{aligned}$$

whenever $\sigma_1, \ldots, \sigma_k$ is a strongly complete set of variants of t. This replacement allows us to get rid of the equational theory entirely during the saturation process; as an immediate consequence it also fixes the non-termination problem in the above example.

5.3.4 Canonical Forms and the Reachability Predicate

A key ingredient of our procedure for deciding static equivalence were the canonical forms of the statements. The canonicalization rule Rename allowed us to "collapse" two recipe variables when the recipes pointed to the same term:

$$\textsc{Rename}\quad \frac{\Big(\mathsf{k}(R, t) \Leftarrow \mathsf{k}(X_1, x_1), \ldots, \mathsf{k}(X_k, x_k)\Big)\ \ \{i, j\} \subseteq \{1, \ldots, n\}\ \ j \neq i \text{ and } x_j = x_i}{\Big(\mathsf{k}(R\{X_i \mapsto X_j\}, t) \Leftarrow \mathsf{k}(X_1, x_1), \ldots, \mathsf{k}(X_{i-1}, x_{i-1}), \mathsf{k}(X_{i+1}, x_{i+1}), \ldots, \mathsf{k}(X_k, x_k)\Big)}$$

In the presence of the history arguments, things get more complicated. We could adapt the rule RENAME to behave as follows:

$$\frac{\Big(\mathsf{k}_H(R,t) \Leftarrow \mathsf{k}_{H_1}(X_1,x_1),\ldots,\mathsf{k}_{H_k}(X_k,x_k)\Big)\ \ \{i,j\}\subseteq\{1,\ldots,n\}\ \ j\neq i \text{ and } x_j = x_i}{\Big(\mathsf{k}_H(R\{X_i\mapsto X_j\},t) \Leftarrow \mathsf{k}_{H_1}(X_1,x_1),\ldots,\mathsf{k}_{H_{i-1}}(X_{i-1},x_{i-1}),\mathsf{k}_{H_{i+1}}(X_{i+1},x_{i+1}),\ldots,\mathsf{k}_{H_k}(X_k,x_k)\Big)}$$

In all of the statements in any knowledge base, all history sequences $H_1,\ldots,H_k$ in the body of the clause are prefixes of the history H in the head of the clause and therefore H_i and H_j are one prefix of the other. If the two histories involved in the above canonicalization rule are equal ($H_i = H_j$), then there is no problem with the above rule. However, if one is a strict prefix of the other, we have to choose if we eliminate the ith or the jth antecedent (and therefore rename X_i to X_j or vice-versa). Assume w.l.o.g. that H_i is a strict prefix of H_j.

1. If we eliminate the ith antecedent we risk to lose completeness due to the fact that we cannot prove that f' implies f (where f' is obtained from f by the above rule).

2. Vice-versa, eliminating the jth antecedent is not sound (f does not imply f'), since a recipe which is valid after running history H_i is not valid after running history H_j if the trace blocks sometime between H_i and H_j.

We chose to solve this problem by adopting a "soft" semantics for the k predicate: $\mathsf{k}_H(R,t)$ intuitively denotes that if a trace can successfully execute throughout history H, then R is a recipe for t. If H cannot be executed (due to the fact that the trace blocks), then there is no guarantee. This semantics of the k predicate makes the second option above sound. As a consequence of the use of the "soft" semantics of k, we introduce a new predicate r_H (taking as arguments only the history), which denotes that the trace will successfully execute all commands in H. The initial knowledge base contains Horn clauses which state that r_H holds whenever all the tests in H hold and the intruder can provide recipes for the inputs appearing in H. The fact that the tests in H hold is modeled by applying complete sets of unifiers as previously described and the fact that the intruder has to provide recipes for the inputs appearing in H is modeled by the antecedents of the Horn clauses.

5.3.5 Identities and Reachable Identities

The intruder identity predicate i is, as expected, also decorated with history arguments. By symmetry with the "knows" predicate, it enjoys the "soft" semantics: $\mathsf{i}_H(R,R')$ intuitively denotes that if the trace can execute the commands in the history H, then R and R' will be recipes for the same term. In order to verify equivalence of two traces, we need however *reachable identities.*

We will therefore also use a predicate $\mathsf{ri}_H(R,R')$ which intuitively states that H is executable and R and R' are recipes for the same term after execution of H:

$$\mathsf{ri}_H(R,R') \text{ is logically equivalent to } \mathsf{r}_H \wedge \mathsf{i}_H(R,R').$$

After saturation takes place, the algorithm for checking trace equivalence will verify that all reachable identities in solved statements hold of the other trace and vice-versa.

We have briefly surveyed the difficulties associated with adapting our procedure to an active intruder. We have also informally presented the workarounds that we propose to circumvent these difficulties. We now proceed to fully formalize the intuitions given above, describe the main steps of our saturation procedure and prove that the new saturation procedure is sound and complete.

5.4 Modeling Traces as Horn Clauses

As we have already announced, the new saturation procedure will make use of four predicates.

5.4.1 Predicates and Semantics.

The intruder knowledge predicate $\mathsf{k}_{l_1,\ldots,l_k}(R,t)$ intuitively denotes that if the trace executed actions $l_1,\ldots,l_n$, then R is a recipe for t in the resulting frame. The intruder identity predicate $\mathsf{i}_{l_1,\ldots,l_n}(R,R')$ intuitively denotes that if the trace executed actions $l_1,\ldots,l_n$, then R and R' are recipes for the same term in the resulting frame.

The intruder reachability predicate $\mathsf{r}_{l_1,\ldots,l_n}$ denotes that the trace can indeed execute actions $l_1,\ldots,l_n$. The intruder reachable identity predicate $\mathsf{ri}_{l_1,\ldots,l_n}(R,R')$ is intuitively a conjunction of the intruder identity predicate $\mathsf{i}_{l_1,\ldots,l_n}(R,R')$ and the intruder reachability predicate $\mathsf{r}_{l_1,\ldots,l_n}$: $\mathsf{ri}_{l_1,\ldots,l_n}(R,R')$ intuitively states that the actions $l_1,\ldots,l_n$ can be executed by the trace and that after the execution R and R' are recipes for the same term in the resulting frame.

The predicates and their semantics are given in Figure 5.1. In formulas we also allow the usual logical connectives and existential and universal quantification of variables with the expected semantics. Formulas are interpreted over a symbolic trace T, a frame φ and a valuation σ.

When a formula f is ground we simply write $(T,\varphi)\models f$ to denote that this formula holds for (T,φ). If moreover, $\mathcal{D}om(\varphi)=\emptyset$ we simply write $T\models f$ for $(T,\varphi)\models f$.

Predicates :

$\mathsf{r}_{l_1,\ldots,l_k}$	(Reachability predicate)
$\mathsf{k}_{l_1,\ldots,l_k}(R,t)$	(intruder Knowledge predicate)
$\mathsf{i}_{l_1,\ldots,l_k}(R,R')$	(Identity predicate)
$\mathsf{ri}_{l_1,\ldots,l_k}(R,R')$	(reachable identity predicate)

Semantics:

$(T,\varphi_0)\models \mathsf{r}_{\ell_1,\ldots,\ell_i}$	if $(T,\varphi_0)\xrightarrow{L_1}(T_1,\varphi_1)\xrightarrow{L_2}\ldots\xrightarrow{L_n}(T_n,\varphi_n)$ such that $\ell_i =_{\mathsf{R}} L_i\varphi_{i-1}$ for all $1\le i\le n$
$(T,\varphi_0)\models \mathsf{k}_{\ell_1,\ldots,\ell_i}(R,t)$	if when $(T,\varphi_0)\xrightarrow{L_1}(T_1,\varphi_1)\xrightarrow{L_2}\ldots\xrightarrow{L_n}(T_n,\varphi_n)$ such that $\ell_i =_{\mathsf{R}} L_i\varphi_{i-1}$ for all $1\le i\le n$ then $\varphi_n\vdash^R t$
$(T,\varphi_0)\models \mathsf{i}_{\ell_1,\ldots,\ell_i}(R,R')$	if there exists t s.t. $(T,\varphi_0,)\models \mathsf{k}_{\ell_1,\ldots,\ell_i}(R,t)$ and $(T,\varphi_0,)\models \mathsf{k}_{\ell_1,\ldots,\ell_i}(R',t)$
$(T,\varphi_0)\models \mathsf{ri}_{\ell_1,\ldots,\ell_i}(R,R')$	if $(T,\varphi_0,)\models \mathsf{r}_{\ell_1,\ldots,\ell_i}$ and $(T,\varphi_0,)\models \mathsf{i}_{\ell_1,\ldots,\ell_i}(R,R')$

Figure 5.1: Semantics of predicates (for ground terms R,R',t)

Example 5.5. Throughout this chapter, we will consider as a running example the following traces

$$T=\mathbf{send}(c,\mathrm{enc}(a,k)).\mathbf{send}(c,\mathrm{enc}(a',k)).\mathbf{receive}(c,x).[x\stackrel{?}{=}\mathrm{enc}(\mathrm{dec}(x,k),k)].\mathbf{send}(c,ok)$$
$$S_1=\mathbf{send}(c,\mathrm{enc}(a,k)).\mathbf{send}(c,\mathrm{enc}(a',k)).\mathbf{receive}(c,x).[x\stackrel{?}{=}\mathrm{enc}(a,k)].\mathbf{send}(c,ok)$$
$$S_2=\mathbf{send}(c,\mathrm{enc}(a,k)).\mathbf{send}(c,\mathrm{enc}(a',k)).\mathbf{receive}(c,x).[x\stackrel{?}{=}\mathrm{enc}(a',k)].\mathbf{send}(c,ok)$$

where $a,a',k'\in\mathcal{N}$ are private names, where $ok\in\mathcal{M}$ is a public name and where $c\in\mathcal{C}$ is a public channel. We will also assume that we are using the classical Dolev-Yao term rewriting system $\mathsf{R}=\{\mathrm{dec}(\mathrm{enc}(x,y),y)\to x\}$. In the trace T, the test $[x\stackrel{?}{=}\mathrm{enc}(\mathrm{dec}(x,k),k)]$ is a trick in order to test if x is an encryption with the key k. If x is of the form $\mathrm{enc}(t,k)$ for some term t, the test will succeed; otherwise it will fail.

We have that the formulas $\mathsf{k}_{\mathbf{send}(c)}(w_1,\mathrm{enc}(a,k))$ and $\mathsf{k}_{\mathbf{send}(c),\mathbf{send}(c)}(w_2,\mathrm{enc}(a',k))$ are true in all three traces (equipped with the empty frame). We also have that the formula

$$\mathsf{i}_{\mathbf{send}(c),\mathbf{send}(c),\mathbf{receive}(\mathrm{enc}(a,k)),\mathbf{test},\mathbf{send}(c)}(w_3,ok)$$

is true of all three traces (equipped with the empty frame): for T and S_1 the history is executable and the third output is the public name ok, while for the third trace the history is not executable and the i predicate holds for any recipes, including w_3 and ok. The formula $\mathsf{r}_{\mathbf{send}(c),\mathbf{send}(c),\mathbf{receive}(\mathrm{enc}(a,k)),\mathbf{test},\mathbf{send}(c)}$ is true of T and S_1 (equipped with the empty frame), but not of S_2 (equipped with the empty frame). The formula

$$\mathsf{ri}_{\mathbf{send}(c),\mathbf{send}(c),\mathbf{receive}(\mathrm{enc}(a,k)),\mathbf{test},\mathbf{send}(c)}(w_3, ok)$$

is true of T and S_1 (equipped with the empty frame), but not of S_2 (equipped with the empty frame). The formula $\mathsf{k}(w_1, k)$ (with an empty sequence of history arguments) is true in the traces T, S_1, S_2 equipped with the frame $\varphi = \{w_1 \mapsto k\}$.

Statements. We now define a class of Horn clauses that we call statements and which are extensively used throughout this chapter.

Definition 5.7. A *statement* is a Horn clause of the form $H \Leftarrow B_1, \ldots, B_n$ where:

1. $H \in \{\mathsf{r}_{l_1,\ldots,l_k}, \mathsf{k}_{l_1,\ldots,l_k}(r,t), \mathsf{i}_{l_1,\ldots,l_k}(r,r'), \mathsf{ri}_{l_1,\ldots,l_k}(r,r')\}$ and
2. $B_i = \mathsf{k}_{l_1,\ldots,l_{j_i}}(X_i, t_i)$ $(1 \leq i \leq n)$

for some terms $l_1, \ldots l_k, t_1, \ldots, t_n$ such that $j_i \leq k$ $(1 \leq i \leq n)$ and for some distinct variables $X_1, \ldots, X_n \in \mathcal{X}$. Moreover, if $H = \mathsf{k}_{l_1,\ldots,l_k}(R,t)$ for some term t, then $\mathcal{V}ar(t) \subseteq \mathcal{V}ar(t_1, \ldots, t_n)$.

We implicitly assume that in a Horn clause all variables are universally quantified. Hence, all statements are closed formulas. The goal of the saturation procedure is to go from a set of initial statements to a set of *solved* statements which is equivalent to the initial set of statements.

Definition 5.8. A statement is *solved* if for all $1 \leq i \leq n$, $B_i = \mathsf{k}_{l_1,\ldots,l_{j_i}}(X_i, x_i)$ for some variable $x_i \in \mathcal{X}$.

Working with a set of solved statements will make checking trace equivalence easier.

Example 5.6. For the trace T in our running example, we have that the following statements hold in T (equipped with the empty frame):

$$\mathsf{k}_{\mathbf{send}(c)}(w_1, \mathrm{enc}(a,k)) \Leftarrow$$
$$\mathsf{k}_{\mathbf{send}(c),\mathbf{send}(c),\mathbf{receive}(\mathrm{enc}(x,k)),\mathbf{test},\mathbf{send}(c)}(w_3, ok) \Leftarrow \mathsf{k}_{\mathbf{send}(c),\mathbf{send}(c)}(X, \mathrm{enc}(x,k))$$
$$\mathsf{k}_{\mathbf{send}(c),\mathbf{send}(c),\mathbf{receive}(x),\mathbf{test},\mathbf{send}(c)}(ok, ok) \Leftarrow$$
$$\mathsf{i}_{\mathbf{send}(c),\mathbf{send}(c),\mathbf{receive}(\mathrm{enc}(x,k)),\mathbf{test},\mathbf{send}(c)}(w_3, ok) \Leftarrow \mathsf{k}_{\mathbf{send}(c),\mathbf{send}(c)}(X, \mathrm{enc}(x,k))$$
$$\mathsf{r}_{\mathbf{send}(c),\mathbf{send}(c),\mathbf{receive}(\mathrm{enc}(a,k)),\mathbf{test},\mathbf{send}(c)} \Leftarrow$$
$$\mathsf{k}_{\mathbf{send}(c),\mathbf{send}(c),\mathbf{receive}(z),\mathbf{test},\mathbf{send}(c)}(\mathrm{enc}(X,Y), \mathrm{enc}(x,y)) \Leftarrow$$
$$\quad \mathsf{k}_{\mathbf{send}(c),\mathbf{send}(c),\mathbf{receive}(x),\mathbf{test},\mathbf{send}(c)}(X,x),$$
$$\quad \mathsf{k}_{\mathbf{send}(c),\mathbf{send}(c),\mathbf{receive}(x),\mathbf{test},\mathbf{send}(c)}(Y,y)$$

5.4.2 The Seed Statements

To each trace T, we will associate a set of statements *Seed*(T) which we call the seed statements and which model the trace in a fully abstract manner. In this section. we explain how to compute the seed statements for any ground trace. Let $T = a_1.a_2.\ldots.a_n$ be a ground trace. We assume w.l.o.g. the following naming conventions:

1. if a_i is a receive action then $a_i = \mathbf{receive}(c_i, x_i)$,
2. if a_i is a send action then $a_i = \mathbf{send}(c_i, t_i)$,
3. if a_i is a test actions then $a_i = [s_i \stackrel{?}{=} t_i]$.

and that $x_i \neq x_j$ for any $i \neq j$. Moreover, for each $1 \leq i \leq n$ we define ℓ_i as follows:

$$\ell_i = \begin{cases} \mathbf{receive}(c_i, x_i) & \text{if } a_i = \mathbf{receive}(c_i, x_i) \\ \mathbf{send}(c_i) & \text{if } a_i = \mathbf{send}(c_i, t_i) \\ \mathbf{test} & \text{if } a_i = [s_i \stackrel{?}{=} t_i] \end{cases}$$

and let the sets $R(m)$, $S(m)$ and $T(m)$ denote the indices of the receive actions, send actions and respectively test actions of $a_1, \ldots, a_m$:

$$\begin{cases} R(m) &= \{i \mid 1 \leq i \leq m, a_i = \mathbf{receive}(c_i, x_i)\} \\ S(m) &= \{i \mid 1 \leq i \leq m, a_i = \mathbf{send}(c_i, t_i)\} \\ T(m) &= \{i \mid 1 \leq i \leq m, a_i = [s_i \stackrel{?}{=} t_i]\} \end{cases}$$

Given a trace T and a set of public names $\mathcal{M}_0 \subseteq \mathcal{M}$, the *set of seed statements* associated to T and $\mathcal{M}_0$, denoted $\mathit{Seed}(T, \mathcal{M}_0)$, is defined to be the set of statements given in Figure 5.2. If $\mathcal{M}_0 = \mathcal{M}$, the set $\mathit{Seed}(T, \mathcal{M}_0) = \mathit{Seed}(T, \mathcal{M})$ is said to be the set of seed statements associated to T and we write in this case $\mathit{Seed}(T)$ as a shortcut for $\mathit{Seed}(T, \mathcal{M})$.

Note that in the set of seed statements we apply complete sets of unifiers modulo R for all tests. We use the notation $\mathsf{mgu}_{\mathsf{R}}(\{s_i = t_i\}_{1 \leq i \leq n})$ for a complete set of unifiers which render s_i equal to t_i modulo R for all $1 \leq i \leq n$. Moreover, we apply strongly complete sets of variants. We use the notation $\mathcal{V}\mathit{ariants}(s_1, \ldots, s_n)$ for a strongly complete set of variants of $\mathsf{tuple}(s_1, \ldots, s_n)$, where tuple is a fresh free function symbol of arity n. The fact of applying complete sets of unifiers and strongly complete sets of variants allows us to *get rid* of the underlying term rewriting system and therefore work in the free algebra during the saturation process.

$\mathsf{r}_{\ell_1\sigma\tau\downarrow,\ldots,\ell_m\sigma\tau\downarrow} \Leftarrow \{\mathsf{k}_{\ell_1\sigma\tau\downarrow,\ldots,\ell_{j-1}\sigma\tau\downarrow}(X_j, x_j\sigma\tau\downarrow)\}_{j \in R(m)}$
for all $0 \leq m \leq n$
for all $\sigma \in \mathsf{mgu}_{\mathsf{R}}(\{s_k = t_k\}_{k \in T(m)})$
for all $\tau \in \mathcal{V}\mathit{ariants}(\ell_1\sigma, \ldots, \ell_m\sigma)$

$\mathsf{k}_{\ell_1\tau\downarrow,\ldots,\ell_m\tau\downarrow}(w_{|S(m)|}, t_m\tau\downarrow) \Leftarrow \{\mathsf{k}_{\ell_1\tau\downarrow,\ldots,\ell_{j-1}\tau\downarrow}(X_j, x_j\tau\downarrow)\}_{j \in R(m)}$
for all $m \in S(n)$
for all $\tau \in \mathcal{V}\mathit{ariants}(\ell_1, \ldots, \ell_m, t_m)$

$\mathsf{k}(c, c) \Leftarrow$
for all public names $c \in \mathcal{M}_0$

$\mathsf{k}_{\ell_1,\ldots,\ell_m}(f(Y_1, \ldots, Y_k), f(y_1, \ldots, y_k)\tau\downarrow) \Leftarrow \{\mathsf{k}_{\ell_1,\ldots,\ell_m}(Y_j, y_j\tau\downarrow)\}_{j \in \{1,\ldots,k\}}$
for all $0 \leq m \leq n$
for all function symbols f of arity k
for all $\tau \in \mathcal{V}\mathit{ariants}(f(y_1, \ldots, y_k))$.

Figure 5.2: Seed statements associated to T and $\mathcal{M}_0 \subseteq \mathcal{M}$

Example 5.7. Recall the trace T in our running example:

$$T = \mathbf{send}(c, \mathrm{enc}(a, k)).\mathbf{send}(c, \mathrm{enc}(a', k)).\mathbf{receive}(c, x).[x \stackrel{?}{=} \mathrm{enc}(\mathrm{dec}(x, k), k)].\mathbf{send}(c, ok).$$

Following the naming conventions above, we have that

$$T = \mathbf{send}(c_1, t_1).\mathbf{send}(c_2, t_2).\mathbf{receive}(c_3, x_3).[s_4 \stackrel{?}{=} t_4].\mathbf{send}(c_5, t_5),$$

where $c_1 = c_2 = c_3 = c_5 = c$, $x_3 = x$, $t_1 = \text{enc}(a, k)$, $t_2 = \text{enc}(a', k)$, $s_4 = x_3 = x$, $t_4 = \text{enc}(\text{dec}(x_3, k), k)) = \text{enc}(\text{dec}(x, k), k)$ and $t_5 = ok$. We also have that $\ell_1 = \mathbf{send}(c), \ell_2 = \mathbf{send}(c), \ell_3 = \mathbf{receive}(c, x), \ell_4 = \mathbf{test}$ and $\ell_5 = \mathbf{send}(c)$.

Using for example the equational unification algorithm described in Section 3.6, we find that $\mathsf{mgu}_{\mathsf{R}}(\{x = \text{enc}(\text{dec}(x, k), k)\}) = \{\{x \mapsto \text{enc}(y, k)\}\}$, i.e. the only equational unifier of the two terms is the substitution $\sigma = \{x \mapsto \text{enc}(y, k)\}$. We have that the set of variants $\mathcal{V}ariants(\ell_1\sigma, \ell_2\sigma, \ell_3\sigma, \ell_4\sigma, \ell_5\sigma) = \mathcal{V}ariants(\ell_1, \ell_2, \mathbf{receive}(c, \text{enc}(y, k)), \ell_4, \ell_5)$. The algorithm described in section 3.5 gives that the identity substitution $\tau = \{\}$ forms by itself a strongly complete set of variants of the tuple $\mathbf{send}(c), \mathbf{send}(c), \mathbf{receive}(c, \text{enc}(y, k)), \mathbf{test}$. We have therefore that the statement

$$\mathsf{r}_{\mathbf{send}(c),\mathbf{send}(c),\mathbf{receive}(c,\text{enc}(y,k)),\mathbf{test},\mathbf{send}(c)} \Leftarrow \mathsf{k}_{\mathbf{send}(c),\mathbf{send}(c)}(X_3, \text{enc}(y, k))$$

belongs to the set of seed statements of T (it is one of the statements of the first type in Figure 5.2).

Letting $m = 5 \in S(n)$ and $\sigma = \{\} \in \mathcal{V}ariants(\ell_1, \ldots, \ell_5, t_5)$, we have that

$$\mathsf{k}_{\mathbf{send}(c),\mathbf{send}(c),\mathbf{receive}(c,x),\mathbf{test},\mathbf{send}(c)}(w_3, ok) \Leftarrow \mathsf{k}_{\mathbf{send}(c),\mathbf{send}(c)}(X_3, x)$$

is in the set of seed statements of T (it is one of the statement of the second type in Figure 5.2). If $b \in M$ is a public name, then the statement

$$\mathsf{k}(b, b) \Leftarrow$$

is in the seed set of statements (third type of statement in Figure 5.2).

Finally, we consider the function symbol dec and a strongly complete set of variants of the term $\text{dec}(y_1, y_2)$. We can use the algorithm in Section 3.5 to compute $\mathcal{V}ariants(\text{dec}(y_1, y_2)) = \{\{\}, \{y_1 \mapsto \text{enc}(z, y_2)\}\}$. Therefore, the following two statements

$$\mathsf{k}_{\mathbf{send}(c)}(\text{dec}(Y_1, Y_2), \text{dec}(y_1, y_2)) \Leftarrow \mathsf{k}_{\mathbf{send}(c)}(Y_1, y_1), \mathsf{k}_{\mathbf{send}(c)}(Y_2, y_2) \text{ and}$$
$$\mathsf{k}_{\mathbf{send}(c)}(\text{dec}(Y_1, Y_2), z) \Leftarrow \mathsf{k}_{\mathbf{send}(c)}(Y_1, \text{enc}(z, y_2)), \mathsf{k}_{\mathbf{send}(c)}(Y_2, y_2)$$

are in the set of seed statements of T (fourth type of statement in Figure 5.2).

We will now show that the set of seed statements is a *fully abstract* encoding of the trace in the sense that is made precise by the following soundness and completeness lemmas.

Lemma 5.1 (Soundness of the set of seed statements)**.** *Let T be a ground trace. For any statement f in the set of seed statements Seed(T) we have that $T \models f$.*

Proof. We assume the same naming conventions for T as in the beginning of the section. We prove that for each statement $f \in \mathit{Seed}(T)$ we have that $T \models f$.

1. Let $m \in \{0, \ldots, n\}$, let $\sigma \in \mathsf{mgu}_{\mathsf{R}}(\{s_k = t_k\}_{k \in T(m)})$ and $\tau \in \mathcal{V}ariants(l_1\sigma, \ldots, l_m\sigma)$ be substitutions as in the first type of statements in Figure 5.2. We show that

$$f = \left(\mathsf{r}_{\ell_1\sigma\tau\downarrow,\ldots,\ell_m\sigma\tau\downarrow} \Leftarrow \{\mathsf{k}_{\ell_1\sigma\tau\downarrow,\ldots,\ell_{j-1}\sigma\tau\downarrow}(X_j, x_j\sigma\tau\downarrow)\}_{j \in R(m)})\right)$$

is a statement that is true in T. Let ω be an arbitrary substitution grounding for f. Assume furthermore that

$$T \models (\mathsf{k}_{\ell_1\sigma\tau\downarrow,\ldots,\ell_{j-1}\sigma\tau\downarrow}(X_j, x_j\sigma\tau\downarrow))\omega$$

for all $j \in R(m)$. We show that $T \models (\mathsf{r}_{\ell_1\sigma\tau\downarrow,\ldots,\ell_m\sigma\tau\downarrow})\omega$ by induction on m.

Base case: $m = 0$. We indeed have that $T \models (\mathsf{r}_{\ell_1\sigma\tau\downarrow,\ldots,\ell_m\sigma\tau\downarrow})\omega$ as $m = 0$ and therefore $(\mathsf{r}_{\ell_1\sigma\tau\downarrow,\ldots,\ell_m\sigma\tau\downarrow})\omega = \mathsf{r}$.

Inductive case: $m > 0$. We assume that $T \models (\mathsf{r}_{\ell_1\sigma\tau\downarrow,\ldots,\ell_{m-1}\sigma\tau\downarrow})\omega$ and we show that $T \models (\mathsf{r}_{\ell_1\sigma\tau\downarrow,\ldots,\ell_m\sigma\tau\downarrow})\omega$ by case analysis on a_m. Let $T_1 = T$ and $\varphi_1 = \emptyset$. As $T \models (\mathsf{r}_{\ell_1\sigma\tau\downarrow,\ldots,\ell_{m-1}\sigma\tau\downarrow})\omega$, we have that there exist $L_1, \ldots, L_{m-1}$ such that

$$(T_i, \varphi_i) \xrightarrow{L_i} (T_{i+1}, \varphi_{i+1})$$

and $L_i\varphi_i =_{\mathsf{R}} \ell_i\sigma\tau\downarrow\omega$ for all $1 \leq i < m$, where $T_i = (a_i.\ldots.a_n)\{x_j \mapsto x_j\sigma\tau\downarrow\omega\}_{j\in R(i-1)}$ and where φ_i extends φ_{i-1} (for all $1 < i \leq m$).

(a) if $a_m = \mathbf{send}(c_m, t_m)$, then $\ell_m = \mathbf{send}(c_m)$ by definition. Let the trace $T_{m+1} = (a_{m+1}.\ldots.a_n)\{x_j \mapsto x_j\sigma\tau\downarrow\omega\}_{j\in R(m)}$ and let $\varphi_{m+1} = \varphi_m \cup \{w_{|\mathcal{D}om(\varphi_m)|+1} \mapsto t_m\sigma\tau\downarrow\omega\}$. Let $L_m = \mathbf{send}(c_m)$. By the definition of $\rightarrow$, we have that

$$(T_m, \varphi_m) \xrightarrow{L_m} (T_{m+1}, \varphi_{m+1}),$$

which is was we wanted to prove.

(b) if $a_m = [s_m \stackrel{?}{=} t_m]$, then $\ell_m = \mathbf{test}$. Let $T_{m+1} = (a_{m+1}.\ldots.a_n)\{x_j \mapsto x_j\sigma\tau\downarrow\omega\}_{j\in R(m)}$ and let $\varphi_{m+1} = \varphi_m$. As $\sigma \in \mathsf{mgu}_{\mathsf{R}}(\{s_k = t_k\}_{k\in T(m)})$, we have that $s_m\sigma =_{\mathsf{R}} t_m\sigma$ and therefore $s_m\sigma\tau\downarrow\omega =_{\mathsf{R}} t_m\sigma\tau\downarrow\omega$. Hence the precondition in the definition of $\rightarrow$ holds and by letting $L_m = \mathbf{test}$, we have that

$$(T_m, \varphi_m) \xrightarrow{L_m} (T_{m+1}, \varphi_{m+1}),$$

which is what we wanted to prove.

(c) If $a_m = \mathbf{receive}(c_m, x_m)$, we know that $m \in R(m)$. Let $T_{m+1} = (a_{m+1}.\ldots.a_n)\{x_j \mapsto x_j\sigma\tau\downarrow\omega\}_{j\in R(m)}$ and let $\varphi_{m+1} = \varphi_m$. As $m \in R(m)$, we have that the trace $T \models (\mathsf{k}_{\ell_1\sigma\tau\downarrow,\ldots,\ell_{m-1}\sigma\tau\downarrow}(X_m, x_m\sigma\tau\downarrow))\omega$ (this is an antecedent of f). Therefore $\varphi_m \vdash^{X_m\omega} x_m\sigma\tau\downarrow\omega$ and, by letting $L_m = \mathbf{receive}(c_m, X_m\omega)$, we obtain by the definition of $\rightarrow$ that

$$(T_m, \varphi_m) \xrightarrow{L_m} (T_{m+1}, \varphi_{m+1}),$$

which is what we wanted to prove.

We have shown that $T \models (\mathsf{r}_{\ell_1\sigma\tau\downarrow,\ldots,\ell_m\sigma\tau\downarrow})\omega$, which means that the statement f holds in T.

2. Let $m \in S(n)$ and let $\sigma \in \mathcal{V}ariants(l_1, \ldots, l_n, t_m)$. We show that the statement

$$f = \Big((\mathsf{k}_{\ell_1\sigma\downarrow,\ldots,\ell_m\sigma\downarrow}(w_{|S(m)|}, (t_m\sigma)\downarrow) \Leftarrow \{\mathsf{k}_{\ell_1\sigma\downarrow,\ldots,\ell_{j-1}\sigma\downarrow}(X_j, x_j\sigma\downarrow)\}_{j\in R(m)})\Big)$$

is true in T. Let ω be a substitution grounding for f. We assume that

$$T \models (\mathsf{k}_{\ell_1\sigma\downarrow,\ldots,\ell_{j-1}\sigma\downarrow}(X_j, x_j\sigma\downarrow))\omega$$

for all $j \in R(m)$ and we show that $T \models (\mathsf{k}_{\ell_1\sigma\downarrow,\ldots,\ell_m\sigma\downarrow}(w_{|S(m)|}, (t_m\sigma)\downarrow))\omega$. Let $T_i = (a_i.\ldots.a_n)\{x_j \mapsto x_j\sigma\omega\}_{j\in R(i-1)}$ and $\varphi_i = \cup_{1\leq j\leq |S(i-1)|}\{w_j \mapsto t_{o(j)}\sigma\omega\}$, where $o(j) = \min\{x \mid |S(x)| = j\}$, *i.e.* $o(j)$ denotes the index of the jth send action. We distinguish two cases:

(a) if there exist $L_1, \ldots, L_m$ such that $(T_1, \varphi_1) \xrightarrow{L_1} (T_2, \varphi_2) \xrightarrow{L_2} \ldots \xrightarrow{L_m} (T_{l+1}, \varphi_{l+1})$ such that $L_i\varphi_i =_{\mathsf{R}} l_i\sigma\downarrow\omega$ for all $1 \leq i \leq m$, we have that

$$\varphi_m(w_{|S(m)|}) = t_{o(S(m))}\sigma\omega =_{\mathsf{R}} t_m\sigma\omega,$$

which implies $\varphi \vdash^{w_{|S(m)|}} (t_m\sigma)\downarrow\omega$. We obtain $T \models (\mathsf{k}_{\ell_1\sigma\downarrow,\ldots,\ell_m\sigma\downarrow}(w_{|S(m)|}, t_m\sigma\downarrow))\omega$ by the semantics of k, which is what we wanted to show.

(b) otherwise, we trivially have that $T \models \mathsf{k}_{\ell_1\sigma\downarrow,\ldots,\ell_m\sigma\downarrow}(w_{|S(m)|}, (t_m\sigma)\downarrow)\omega$ by the "soft" semantics of k.

We have shown that $T \models f$.

3. Let c be a public name. We show that

$$f = \Big(\mathsf{k}(c,c) \Leftarrow \Big)$$

is true in T. Indeed, let $\varphi = \emptyset$. We have that $\varphi \vdash^c c$ and therefore $T \models \mathsf{k}(c,c)$.

4. Let g be a function symbol of arity k, let m be an integer such that $0 \leq m \leq n$ and let $\sigma \in \mathcal{V}ariants(g(x_1,\ldots,x_k))$. We show that the statement

$$f = \Big(\mathsf{k}_{\ell_1,\ldots,\ell_m}(g(X_1,\ldots,X_k), g(x_1,\ldots,x_k)\sigma\downarrow) \Leftarrow \{\mathsf{k}_{\ell_1,\ldots,\ell_m}(X_j, x_j\sigma\downarrow)\}_{j\in\{1,\ldots,k\}}\Big)$$

is true in T. Let ω be an arbitrary substitution grounding for f. We assume that $T \models \mathsf{k}_{\ell_1,\ldots,\ell_m}(X_j, x_j\sigma\downarrow)\omega$ for all $1 \leq j \leq k$ and we show that

$$T \models (\mathsf{k}_{\ell_1,\ldots,\ell_m}(g(X_1,\ldots,X_k), g(x_1,\ldots,x_k)\sigma\downarrow))\omega.$$

Let $(T_1,\varphi_1) = (T,\emptyset)$. We distinguish between two cases:

(a) if

$$(T_1,\varphi_1) \xrightarrow{L_1} (T_2,\varphi_2) \xrightarrow{L_2} \ldots \xrightarrow{L_m} (T_{m+1},\varphi_{m+1})$$

such that $L_i\varphi_i =_{\mathsf{R}} \ell_i$ for all $1 \leq i \leq m$, we have that

$$\varphi_{m+1} \vdash^{X_j\omega} x_j\sigma\downarrow\omega$$

for all $1 \leq j \leq k$ by our hypothesis. But this implies

$$\varphi_{m+1} \vdash^{g(X_1\omega,\ldots,X_k\omega)} g(x_1\sigma\downarrow\omega,\ldots,x_k\sigma\downarrow\omega) =_{\mathsf{R}} g(x_1,\ldots,x_k)\sigma\downarrow\omega$$

which immediately implies that $T \models (\mathsf{k}_{\ell_1,\ldots,\ell_m}(g(X_1,\ldots,X_k), g(x_1,\ldots,x_k)\sigma\downarrow))\omega$.

(b) otherwise, we trivially have that $T \models (\mathsf{k}_{\ell_1,\ldots,\ell_m}(g(X_1,\ldots,X_k), g(x_1,\ldots,x_k)\sigma\downarrow)\omega$.

We have shown that $T \models f$.

We have shown for every statement $f \in \mathit{Seed}(T)$ that $T \models f$.

□

In order to state completeness we first define the least Herbrand model of a set of statements.

Definition 5.9. Let K be a set of statements. We define $\mathcal{H}(K)$ to be the smallest set of terms such that:

$$\textsc{Simple Consequence}\ \frac{f = \Big(H \Leftarrow B_1,\ldots,B_n\Big) \in K \qquad \sigma \text{ grounding for f} \qquad B_1\sigma \in \mathcal{H}(K) \qquad \ldots \qquad B_n\sigma \in \mathcal{H}(K)}{H\sigma \in \mathcal{H}(K)}$$

$$\textsc{ExtendK}\ \frac{\mathsf{k}_u(R,t) \in \mathcal{H}(K)}{\mathsf{k}_{uv}(R,t) \in \mathcal{H}(K)} \qquad \textsc{ExtendI}\ \frac{\mathsf{i}_u(R,R') \in \mathcal{H}(K)}{\mathsf{i}_{uv}(R,R') \in \mathcal{H}(K)}$$

Equivalently, $\mathcal{H}(K)$ is the least Herbrand model of

$$K \cup \{\mathsf{k}_{uv}(X,x) \Leftarrow \mathsf{k}_u(X,x)\} \cup \{\mathsf{i}_{uv}(X_1,X_2) \Leftarrow \mathsf{i}_u(X_1,X_2)\}_{n\in\mathbb{N}}.$$

By u and v we denote arbitrary sequences of terms. We will make use of this notation later on when it simplifies the presentation. Although the ExtendI rule is not needed for the completeness lemma below, it will be needed later when we state completeness of the saturation process.

Lemma 5.2 (Completeness of the set of seed statements)**.** *Let T and S be traces and let φ be a frame. If $(T,\emptyset) \xrightarrow{L_1,\ldots,L_n} (S,\varphi)$ then:*

1. $\mathsf{r}_{L_1\varphi\downarrow,\ldots,L_n\varphi\downarrow} \in \mathcal{H}(\mathit{Seed}(T))$
2. *if $\varphi \vdash^R t$ then $\mathsf{k}_{L_1\varphi\downarrow,\ldots,L_n\varphi\downarrow}(R, t\downarrow) \in \mathcal{H}(\mathit{Seed}(T))$*

Proof. We prove the two statements by induction on n. We assume that the two statements hold for any index less than n and we prove them for n. As $(T,\emptyset) \xrightarrow{L_1,\ldots,L_n} (S,\varphi)$, we have that:

1. there exists ω such that $(L_1\varphi\downarrow,\ldots,L_n\varphi\downarrow) = (\ell_1,\ldots,\ell_n)\omega$ and
2. $s_k\omega =_{\mathsf{R}} t_k\omega$ for all $k \in T(n)$.

1. As $s_k\omega =_{\mathsf{R}} t_k\omega$ for all $k \in T(n)$, it follows by the definition of $\mathsf{mgu}_{\mathsf{R}}$ that there exists $\sigma \in \mathsf{mgu}_{\mathsf{R}}(\{s_k \stackrel{?}{=} t_k\}_{k\in T(n)})$ such that:

   (a) $\mathcal{D}om(\sigma) \subseteq X$,
   (b) $s_k\sigma =_{\mathsf{R}} t_k\sigma$ for all $k \in T(n)$ and
   (c) $\omega[X] =_{\mathsf{R}} (\sigma\pi)[X]$ for some substitution π

   where $X = \mathcal{V}ar(\{s_k,t_k\}_{k\in T(n)})$.
   It follows that $(\ell_1,\ldots,\ell_n)\omega\downarrow = (\ell_1,\ldots,\ell_n)\sigma\pi\downarrow$ for some substitution π.
   By the definition of $\mathcal{V}ariants((\ell_1,\ldots,\ell_n)\sigma)$, we have that there exists the substitution $\tau \in \mathcal{V}ariants((\ell_1,\ldots,\ell_n)\sigma)$ such that $(\ell_1,\ldots,\ell_n)\sigma\pi\downarrow = (\ell_1,\ldots,\ell_n)\sigma\tau\downarrow\tau'$ for some substitution τ'.
   By the definition of the seed knowledge base $\mathit{Seed}(T)$, we have that the statement

$$f = \left(\mathsf{r}_{\ell_1\sigma\tau\downarrow,\ldots,\ell_n\sigma\tau\downarrow} \Leftarrow \mathsf{k}_{\ell_1\sigma\tau\downarrow,\ldots,\ell_{j-1}\sigma\tau\downarrow}(X_j, x_j\sigma\tau\downarrow)_{j\in R(n)}\right) \in \mathit{Seed}(T)$$

   is in the seed knowledge base $\mathit{Seed}(T)$.
   Let $\{R_j\}_{j\in R(n)}$ be terms such that $L_j = \mathbf{receive}(c_j, R_j)$ for all $1 \le j \le R(n)$. Let τ'' be a substitution equal to τ' except that it sends X_j to R_j for all $j \in R(n)$. We have by the induction hypothesis that each component $\mathsf{k}_{\ell_1\sigma\tau\downarrow,\ldots,\ell_{j-1}\sigma\tau\downarrow}(X_j, x_j\sigma\tau\downarrow)\tau''$ of the right hand side of $f\tau''$ is in $\mathcal{H}(\mathit{Seed}(T))$. Therefore the head $\mathsf{r}_{\ell_1\sigma\tau\downarrow,\ldots,\ell_n\sigma\tau\downarrow}\tau''$ of $f\tau''$ is also in $\mathcal{H}(\mathit{Seed}(T))$:

$$\mathsf{r}_{\ell_1\sigma\tau\downarrow\tau'',\ldots,\ell_n\sigma\tau\downarrow\tau''} = \mathsf{r}_{\ell_1\sigma\tau\downarrow\tau',\ldots,\ell_n\sigma\tau\downarrow\tau'} \in \mathcal{H}(\mathit{Seed}(T)).$$

2. By induction on R, we show that:

$$\mathsf{k}_{L_1\varphi\downarrow,\ldots,L_n\varphi\downarrow}(R, R\varphi\downarrow) \in \mathcal{H}(\mathit{Seed}(T))$$

   (a) If $R = c$ is a public name, and as the statement $f = \left(\mathsf{k}(c,c) \Leftarrow\right)$ is in the seed knowledge base by definition, we can instantiate f with ω to immediately obtain the solution and immediately get the conclusion by the ExtendK rule.
   (b) If $R = f(R_1,\ldots,R_k)$, let γ be the substitution of domain $\mathcal{D}om(\gamma) \subseteq \{y_1,\ldots,y_k\}$ that maps y_j to $R_j\varphi\downarrow$ for all $1 \le j \le k$.
   Let $\tau \in \mathcal{V}ariants(f(y_1,\ldots,y_k))$ and τ' be such that $R\varphi\downarrow = (f(y_1,\ldots,y_k)\tau)\downarrow\tau'$.
   By the definition of $\mathit{Seed}(T)$, we have that $g \in \mathit{Seed}(t)$, where g is the statement

$$g = \left(\mathsf{k}_{\ell_1,\ldots,\ell_n}(f(Y_1,\ldots,Y_k), f(y_1\tau\downarrow,\ldots,y_k\tau\downarrow)) \Leftarrow \{\mathsf{k}_{\ell_1,\ldots,\ell_n}(Y_j, y_j\tau\downarrow)\}_{j\in\{1,\ldots,k\}}\right).$$

Let $\tau'' = \omega\downarrow \cup \tau' \cup \{Y_j \mapsto R_j\}_{j\in\{1,\ldots,k\}}$. Every antecendent $\mathsf{k}_{\ell_1,\ldots,\ell_n}(Y_j, y_j\tau\downarrow)\tau''$ ($j \in \{1,\ldots,k\}$) in the body of $g\tau''$ is in $\mathcal{H}(\mathit{Seed}(T))$ by the induction hypothesis. Therefore, the head of $g\tau''$ is also in $\mathcal{H}(\mathit{Seed}(T))$:

$$\mathsf{k}_{\ell_1,\ldots,\ell_n}(f(Y_1,\ldots,Y_k), f(y_1\tau\downarrow,\ldots,y_k\tau\downarrow))\tau'' \in \mathcal{H}(\mathit{Seed}(T)),$$

which is equivalent, by the definition of τ'', to what we had to prove:

$$\mathsf{k}_{L_1\varphi\downarrow,\ldots,L_n\varphi\downarrow}(f(R_1,\ldots,R_k), R\varphi\downarrow) \in \mathcal{H}(\mathit{Seed}(T)).$$

(c) If $R = w_j$, let m be the smallest index such that $S(m) = j$ and let $t = t_m$ be the term such that $a_m = \mathbf{send}(c_m, t_m)$. Let $\tau \in \mathcal{V}\mathit{ariants}(\ell_1,\ldots,\ell_m,t)$ and τ' be substitutions such that $(\ell_1,\ldots,\ell_m,t)\omega\downarrow = (\ell_1,\ldots,\ell_m,t)\tau\downarrow\tau'$. We have by the definition of the seed knowledge base that the statement

$$h = \left(\mathsf{k}_{\ell_1\tau\downarrow,\ldots,\ell_m\tau\downarrow}(w_j, t_m\tau\downarrow) \Leftarrow \{\mathsf{k}_{\ell_1\tau\downarrow,\ldots,\ell_{k-1}\tau\downarrow}(X_k, x_k\tau\downarrow)\}_{k\in R(m)}\right) \in K(T).$$

Let R_k be recipes of $x_k\tau\downarrow\tau' =_{\mathsf{R}} x_k\omega$ in the smallest possible prefix of φ. Let $\tau'' = \tau' \cup \{X_k \mapsto R_k\}_{k\in R(m)}$. We have that every component $\mathsf{k}_{\ell_1\tau\downarrow,\ldots,\ell_{k-1}\tau\downarrow}(X_k, x_k\tau\downarrow)\tau''$ ($k \in R(m)$) of the body of $h\tau''$ is in $\mathcal{H}(\mathit{Seed}(T))$ by the induction hypothesis. Therefore the head $\mathsf{k}_{\ell_1\tau\downarrow,\ldots,\ell_m\tau\downarrow}(w_j, t_m\tau\downarrow)\tau''$ of $h\tau''$ is also a simple consequence of the seed knowledge base. By the definition of τ'', we obtain:

$$\mathsf{k}_{L_1\varphi\downarrow,\ldots,L_m\varphi\downarrow}(w_j, t_m\omega\downarrow) \in \mathcal{H}(\mathit{Seed}(T)).$$

By rule ExtendK, we immediately obtain our conclusion.

As $\varphi \vdash^R t$ implies by definition that $t\downarrow = R\varphi\downarrow$, we have proven that $\mathsf{k}_{L_1\varphi\downarrow,\ldots,L_n\varphi\downarrow}(R, t\downarrow) \in \mathcal{H}(K(T))$.

We have shown that the two conclusions hold and therefore the seed knowledge base is complete. □

In Lemma 5.1 and in Lemma 5.2 we prove soundness and respectively completeness of the set of seed statements. Therefore the set $\mathit{Seed}(T)$ is a *fully abstract* encoding of T into Horn clauses.

5.5 Procedure for Proving Trace Equivalence

We will now describe how to manipulate this set of statements in order to automatically check the equivalence of two processes. This is accomplished by the saturation procedure. The goal of the saturation procedure is to go from a sound and complete set of statements (the seed statements) to a sound and complete set of *solved* statements. Using the sound and complete set of solved statements, trace equivalence can then be checked algorithmically.

5.5.1 Knowledge Bases and Saturation

We now present the ingredients of the saturation procedure. During our saturation procedure, we will manipulate well-formed statements:

Definition 5.10. A statement $H \Leftarrow B_1,\ldots,B_n$ is called *well-formed* if whenever it is solved and $H = \mathsf{k}_{t_1,\ldots,t_k}(R,t)$, we have that $t \notin \mathcal{X}$. A set of *well-formed* statements a *knowledge base*.

If K is a knowledge base we define $K|_{\mathsf{solved}} = \{f \mid f \in K, f \text{ is solved }\}$ to be the knowledge base restricted to the solved statements. If $f = \left(H \Leftarrow B_1,\ldots,B_n\right)$ is a statement and $H = \mathsf{k}_u(R,t)$ for some u, R, t, we say that f is a *deduction statement*. If $H = \mathsf{i}_u(R,R')$ for some u, R, R', we say that f is an *equational statement* or an *identity statement*.

Canonical form. Given a solved deduction statement f, we define the *canonical form* of f to be the statement $f\Downarrow$ obtained by first applying Rule RENAME as much as possible and then Rule REMOVE as much as possible:

$$\textsc{Rename}\ \frac{H \Leftarrow \mathsf{k}_u(X,x), \mathsf{k}_{uv}(Y,x), B_1, \ldots, B_n}{(H \Leftarrow \mathsf{k}_u(X,x), B_1, \ldots, B_n)\{Y \mapsto X\}}$$

$$\textsc{Remove}\ \frac{H \Leftarrow \mathsf{k}_u(X,x), B_1, \ldots, B_n \qquad x \notin \mathcal{V}ar(H)}{H \Leftarrow B_1, \ldots, B_n}$$

For any other type of statement f, the canonical form $f\Downarrow$ is defined to be equal to f. Note that the order in which the rules are applied is needed for soundness.

Example 5.8. Let the statement f be:

$$f = \Big(\mathsf{k}_{\mathbf{send}(c)}(\mathrm{dec}(\mathrm{enc}(X,Y),Z),x) \Leftarrow \mathsf{k}_{\mathbf{send}(c)}(X,x), \mathsf{k}_{\mathbf{send}(c)}(Y,y), \mathsf{k}_{\mathbf{send}(c)}(Z,y)\Big).$$

We have that the variable y appears twice in the body of f and we can therefore apply rule RENAME. We obtain that

$$f\Downarrow = \Big(\mathsf{k}_{\mathbf{send}(c)}(\mathrm{dec}(\mathrm{enc}(X,Y),Y),x) \Leftarrow \mathsf{k}_{\mathbf{send}(c)}(X,x), \mathsf{k}_{\mathbf{send}(c)}(Y,y)\Big).$$

Consequence. Given a knowledge base K, we define the set of *consequences* of K, denoted $\mathbf{deriv}(K)$, to be the smallest set such that:

$$\textsc{Axiom}\ \frac{}{\mathsf{k}_{uv}(R,t) \Leftarrow \mathsf{k}_u(R,t), B_1, \ldots, B_m \in \mathbf{deriv}(K)}$$

$$\textsc{Res}\ \frac{\begin{array}{c} H \Leftarrow B_1, \ldots, B_n \in K \\ B_1\sigma \Leftarrow C_1, \ldots, C_m \in \mathbf{deriv}(K) \quad \ldots \quad B_n\sigma \Leftarrow C_1, \ldots, C_m \in \mathbf{deriv}(K) \end{array}}{H\sigma \Leftarrow C_1, \ldots, C_m \in \mathbf{deriv}(K)}$$

Note that if K is a finite knowledge base that only contains solved statements, given input $t, \ell_1, \ldots, \ell_k, i_1, \ldots i_n, x_1, \ldots, x_n, X_1, \ldots, X_n$, it is decidable whether

$$\exists R.\mathsf{k}_{\ell_1,\ldots,\ell_k}(R,t) \Leftarrow \mathsf{k}_{\ell_1,\ldots,\ell_{i_1}}(X_1,x_1), \ldots, \mathsf{k}_{\ell_1,\ldots,\ell_{i_n}}(X_n,x_n) \in \mathbf{deriv}(K).$$

The definition of $\mathbf{deriv}(K)$ yields a direct recursive algorithm for the decision problem above which moreover computes R:

- (Axiom) Check whether $t = x_j$ for $1 \leq j \leq n$. If this is the case return (yes, X_j).
- (Res) Otherwise, guess a (solved) statement $\mathsf{k}_u(R',t') \Leftarrow \mathsf{k}_{u_1}(Y_1,y_1), \ldots \mathsf{k}_{u_k}(Y_k,y_k) \in K$ and compute substitution σ such that $\mathsf{k}_{\ell_1,\ldots,\ell_k}(R',t) = \mathsf{k}_u(R',t')\sigma$. Check recursively whether

$$\exists R_i.\mathsf{k}_{u_i}(R_i,y_i)\sigma \Leftarrow \mathsf{k}_{\ell_1,\ldots,\ell_{i_1}}(X_1,x_1), \ldots, \mathsf{k}_{\ell_1,\ldots,\ell_{i_n}}(X_n,x_n) \in \mathbf{deriv}(K)$$

  for $1 \leq i \leq k$. In that case return (yes, $R'\{Y_i \mapsto R_i\}_{1\leq i\leq n}$). Otherwise return no.

Termination is ensured because the size of t when checking whether

$$\exists R.\mathsf{k}_{\ell_1,\ldots,\ell_k}(R,t) \Leftarrow \mathsf{k}_{\ell_1,\ldots,\ell_{i_1}}(X_1,x_1), \ldots, \mathsf{k}_{\ell_1,\ldots,\ell_{i_n}}(X_n,x_n) \in \mathbf{deriv}(K)$$

strictly decreases in each recursive call. Indeed, when $f = \Big(\mathsf{k}_u(R',t') \Leftarrow \mathsf{k}_{u_1}(Y_1,y_1), \ldots \mathsf{k}_{u_k}(Y_k,y_k)\Big) \in K$ we have that $t' \notin X$ because f is well-formed (by the definition of a knowledge base) and $y_i \in \mathcal{V}ar(t')$ by definition of a statement. Hence, $|y_i\sigma| < |t'\sigma| = |t|$.

Example 5.9. Let $f = \Big(\mathsf{k}_{\mathbf{send}(c),\mathbf{send}(c)}(d,d) \Leftarrow \Big)$ and $g = \Big(\mathsf{r}_{\mathbf{send}(c),\mathbf{send}(c),\mathbf{receive}(c,x)} \Leftarrow \mathsf{k}_{\mathbf{send}(c),\mathbf{send}(c)}(X,x)\Big)$. We have that the statement $h = \Big(\mathsf{r}_{\mathbf{send}(c),\mathbf{send}(c),\mathbf{receive}(c,d)}\Big) \in \mathbf{deriv}(\{f,g\})$ is a consequence of the knowledge base $\{f,g\}$.

Update. Given a knowledge base K and a statement f, the *update of K by f*, denoted $K \oplus f$, is defined to be $K \cup \{f\Downarrow\}$ if the head of f is not of the form $\mathsf{k}_{\ell_1,\ldots,\ell_k}(R,t)$. Otherwise, let

$$f\Downarrow = \mathsf{k}_{\ell_1,\ldots,\ell_k}(R,t) \Leftarrow \mathsf{k}_{\ell_1,\ldots,\ell_{i_1}}(X_1,t_1),\ldots,\mathsf{k}_{\ell_1,\ldots,\ell_{i_n}}(X_n,t_n)$$

and

$$K \oplus f = \begin{cases} (K \cup \{f\Downarrow\}) & \text{if } f \text{ is solved and for any } R' \\ & f'[R'] \notin K' \\ K \cup \{\mathsf{i}_{\ell_1,\ldots,\ell_k}(R,R') & \text{if } f \text{ is solved and for some } R' \\ \quad \Leftarrow \{\mathsf{k}_{\ell_1,\ldots,\ell_{i_j}}(X_j,t_j)\}_{j\in\{1,\ldots,n\}}\} & f'[R'] \in K' \\ K \cup \{f\Downarrow\} & \text{if } f \text{ is not solved} \end{cases}$$

where $K' = \mathbf{deriv}(K|_{\mathsf{solved}})$ and $f'[R'] = \mathsf{k}_{\ell_1,\ldots,\ell_k}(R',t) \Leftarrow \mathsf{k}_{\ell_1,\ldots,\ell_{i_1}}(X_1,t_1),\ldots,\mathsf{k}_{\ell_1,\ldots,\ell_{i_n}}(X_n,t_n)$ is the statement obtained from $f\Downarrow$ by replacing the occurnce of recipe R in the head by R'.

Example 5.10. If an arbitrary knowledge base is updated by

$$f = \Big(\mathsf{k}_{\mathbf{send}(c)}(\mathrm{dec}(\mathrm{enc}(X,Y),Y),x) \Leftarrow \mathsf{k}_{\mathbf{send}(c)}(X,x),\mathsf{k}_{\mathbf{send}(c)}(Y,y)\Big),$$

we have that $\Big(\mathsf{k}_{\mathbf{send}(c)}(X,x) \Leftarrow \mathsf{k}_{\mathbf{send}(c)}(X,x),\mathsf{k}_{\mathbf{send}(c)}(Y,y)\Big) \in \mathbf{deriv}(K)$. Therefore we are in the second case of the update operator and we will add an identity statement

$$\Big(\mathsf{i}_{\mathbf{send}(c)}(\mathrm{dec}(\mathrm{enc}(X,Y),Y),X) \Leftarrow \mathsf{k}_{\mathbf{send}(c)}(X,x),\mathsf{k}_{\mathbf{send}}(Y,y)\Big)$$

to the knowledge base instead of the deduction statement f.

Note that even if the statement f by which a knowledge base K is updated is not well-formed, the resulting set of statements $K \oplus f$ is still a knowledge base, since the case where f is not well-formed is captured by the second case in the definition of $K \oplus f$ and an equational statement is added instead of the not well-formed statement.

Initial knowledge base. The *initial knowledge base* associated to a set of seed statements $\mathit{Seed}(T,\mathcal{M}_0)$, denoted $K_i(\mathit{Seed}(T,\mathcal{M}_0))$, is defined to be the empty knowledge base updated by the set of seed statements:

$$K_i(\mathit{Seed}(T,\mathcal{M}_0)) = \emptyset \oplus_{f\in \mathit{Seed}(T,\mathcal{M}_0)} f.$$

When $\mathcal{M}_0 = \mathcal{M}$, we use $K_i(T)$ as a shortcut of $K_i(\mathit{Seed}(T,\mathcal{M}))$ and we say that $K_i(T)$ is the initial knowledge base associated to T.

Saturation. Given a knowledge base K, the saturated knowledge base $\mathsf{sat}(K)$ is obtained from K by applying the inference rules given in Figure 5.3, until reaching a fixed point (i.e. $\mathsf{sat}(K)$ is such that $K \Rightarrow^* \mathsf{sat}(K)$ and if $\mathsf{sat}(K) \Rightarrow^* K'$, then $K' \subseteq \mathsf{sat}(K)$).

$$\textsc{Resolution}\ \frac{f \in K, g \in K|_{\mathsf{solved}}, \quad f = \Big(H \Leftarrow \mathsf{k}_{uv}(X,t), B_1, \ldots, B_n\Big) \quad g = \Big(\mathsf{k}_w(R,t') \Leftarrow B_{n+1}, \ldots, B_m\Big) \quad \sigma = \mathsf{mgu}(\mathsf{k}_w(R,t'), \mathsf{k}_u(X,t)) \quad t \notin \mathcal{X}}{K \Rightarrow K \oplus h \text{ where } h = \Big((H \Leftarrow B_1, \ldots, B_m)\sigma\Big)}$$

$$\textsc{Equation}\ \frac{f, g \in K|_{\mathsf{solved}}, \quad f = \Big(\mathsf{k}_u(R,t) \Leftarrow B_1, \ldots, B_n\Big) \quad g = \Big(\mathsf{k}_{u'v'}(R',t') \Leftarrow B_{n+1}, \ldots, B_m\Big) \quad \sigma = \mathsf{mgu}(\mathsf{k}_u(\_,t), \mathsf{k}_{u'}(\_,t'))}{K \Rightarrow K \oplus h \text{ where } h = \Big((\mathsf{i}_{u'v'}(R,R') \Leftarrow B_1, \ldots, B_m)\sigma\Big)}$$

$$\textsc{Test}\ \frac{f, g \in K|_{\mathsf{solved}}, \quad f = \Big(\mathsf{i}_u(R,R') \Leftarrow B_1, \ldots, B_n\Big) \quad g = \Big(\mathsf{r}_{u'v'} \Leftarrow B_{n+1}, \ldots, B_m\Big) \quad \sigma = \mathsf{mgu}(u,u')}{K \Rightarrow K \oplus h \text{ where } h = \Big((\mathsf{ri}_{u'v'}(R,R') \Leftarrow B_1, \ldots, B_m)\sigma\Big)}$$

Figure 5.3: Saturation rules

We now state in what sense the saturation procedure is sound and complete. The proofs are detailed in Section 5.6 and Section 5.7.

Theorem 5.2 (soundness of saturation)**.** *If $f \in \mathsf{sat}(K_i(T))$ then $T \models f$.*

Before stating completeness, we have to extend the least Herbrand model associated to a set of statement by interpreting the predicate i_w as a monotonic (in w) congruence relation.

Definition 5.11. Let K be a set of statements. We define $\mathcal{H}_{\mathsf{e}}(K)$ to be the smallest set of terms such that $\mathcal{H}(K) \subseteq \mathcal{H}_{\mathsf{e}}(K)$ and such that:

$$\textsc{Refl}\ \frac{}{\mathsf{i}_w(R,R) \in \mathcal{H}_{\mathsf{e}}(K)} \qquad \textsc{Sym}\ \frac{\mathsf{i}_w(R_1,R_2) \in \mathcal{H}_{\mathsf{e}}(K)}{\mathsf{i}_w(R_2,R_1) \in \mathcal{H}_{\mathsf{e}}(K)} \qquad \textsc{Extend}\ \frac{\mathsf{i}_u(R,R') \in \mathcal{H}_{\mathsf{e}}(K)}{\mathsf{i}_{uv}(R,R') \in \mathcal{H}_{\mathsf{e}}(K)}$$

$$\textsc{Tran}\ \frac{\mathsf{i}_w(R_1,R_2) \in \mathcal{H}_{\mathsf{e}}(K) \qquad \mathsf{i}_w(R_1,R_3) \in \mathcal{H}_{\mathsf{e}}(K)}{\mathsf{i}_w(R_1,R_3) \in \mathcal{H}_{\mathsf{e}}(K)}$$

$$\textsc{Cong}\ \frac{\mathsf{i}_w(R_1,R_1') \in \mathcal{H}_{\mathsf{e}}(K), \cdots, \mathsf{i}_w(R_n,R_n') \in \mathcal{H}_{\mathsf{e}}(K) \qquad f \in \mathcal{F}, \mathsf{ar}(f) = n}{\mathsf{i}_w(f(R_1,\ldots R_n), f(R_1',\ldots R_n')) \in \mathcal{H}_{\mathsf{e}}(K)}$$

$$\textsc{Equational Consequence}\ \frac{\mathsf{k}_w(R,t) \in \mathcal{H}(K) \qquad \mathsf{i}_w(R,R') \in \mathcal{H}_{\mathsf{e}}(K)}{\mathsf{k}_w(R',t) \in \mathcal{H}_{\mathsf{e}}(K)}$$

We are now ready to state the completeness theorem.

Theorem 5.3 (Completeness of saturation)**.** *Let T and S be traces and let φ be a frame such that*

$$(T,\emptyset) \xrightarrow{L_1,\ldots,L_n} (S,\varphi).$$

Let K be a saturated knowledge base associated to T, i.e., $K = \mathsf{sat}(K_i(T))$. Then we have that

1. $\mathsf{r}_{L_1\varphi\downarrow,\ldots,L_n\varphi\downarrow} \in \mathcal{H}_{\mathsf{e}}(K|_{\mathsf{solved}})$,
2. *if $\varphi \vdash^R t$ then $\mathsf{k}_{L_1\varphi\downarrow,\ldots,L_n\varphi\downarrow}(R,t\downarrow) \in \mathcal{H}_{\mathsf{e}}(K|_{\mathsf{solved}})$ and*
3. *if $\varphi \vdash^R t$ and $\varphi \vdash^{R'} t$, then $\mathsf{i}_{L_1\varphi\downarrow,\ldots,L_n\varphi\downarrow}(R,R') \in \mathcal{H}_{\mathsf{e}}(K|_{\mathsf{solved}})$.*

5.5.2 Effectiveness of the Saturation

In the set of seed statements, there are an infinite number of statements due to the statements of the form

$$\mathsf{k}(m,m) \Leftarrow$$

for names $m \in \mathcal{M}$. We will show that all but a finite number of them do not interfere in the saturation process and that it is therefore sufficient to consider a finite set of seed statements for saturation:

Lemma 5.3. *Let T be a trace, let $\mathcal{M}_T = \{m \in \mathcal{M} \mid m \in \mathcal{N}ames(T)\}$ be the set of public names occurring in T and let $\mathcal{M}_{\bar{T}} = \mathcal{M} \setminus \mathcal{M}_T$ be the set of public names not occuring in T. Let*

$$K_{\mathcal{M}_{\bar{T}}} = \{\{\mathsf{k}(m,m) \Leftarrow\}_{m\in\mathcal{M}_{\bar{T}}} \cup \{\mathsf{i}(m,m) \Leftarrow\}_{m\in\mathcal{M}_{\bar{T}}} \cup \{\mathsf{ri}(m,m) \Leftarrow\}_{m\in\mathcal{M}_{\bar{T}}}\}.$$

Then $\mathsf{sat}(K_i(T)) = \mathsf{sat}(K_i(\mathit{Seed}(T,\mathcal{M}_T))) \uplus K_{\mathcal{M}_{\bar{T}}}$.

To prove this lemma, we first need two other helper lemmas. Let $\mathcal{M}_0 \subseteq \mathcal{M}$ be public names and let

$$K_{names} = \{\{\mathsf{k}(m,m) \Leftarrow\}_{m\in\mathcal{M}_0} \cup \{\mathsf{i}(m,m) \Leftarrow\}_{m\in\mathcal{M}_0} \cup \{\mathsf{ri}(m,m) \Leftarrow\}_{m\in\mathcal{M}_0}\}$$

be a set of statements involving names in $\mathcal{M}_0$. Recall that $\Rightarrow$ denotes the saturation relation; we will denote by $\Rightarrow^=$ its reflexive closure.

Lemma 5.4. *Let K be a knowledge base such that $\mathcal{N}ames(K) \cap \mathcal{M}_0 = \emptyset$. Let $K_1 \subseteq K_{names}$. If h is a statement such that $\mathcal{N}ames(h) \cap \mathcal{M}_0 = \emptyset$, then*

$$(K \uplus K_1) \oplus h = (K \oplus h) \uplus K_1.$$

Proof. If h is not solved or if it is not a deduction statement, we have that $(K \uplus K_1) \oplus h = (K \uplus K_1) \cup \{h\} = (K \cup \{h\}) \uplus K_1 = (K \oplus h) \uplus K_1$. If h is a solved deduction statement, let

$$h\Downarrow = \mathsf{k}_{\ell_1,\ldots,\ell_k}(R,t) \Leftarrow \mathsf{k}_{\ell_1,\ldots,\ell_{i_1}}(X_1,x_1),\ldots,\mathsf{k}_{\ell_1,\ldots,\ell_{i_n}}(X_n,x_n).$$

We distinguish two cases:

1. either $\mathsf{k}_{\ell_1,\ldots,\ell_k}(R',t) \Leftarrow \mathsf{k}_{\ell_1,\ldots,\ell_{i_1}}(X_1,x_1),\ldots,\mathsf{k}_{\ell_1,\ldots,\ell_{i_n}}(X_n,x_n) \notin \mathbf{deriv}((K \uplus K_1)|_{\mathsf{solved}})$ for any R', in which case

   $$(K \uplus K_1) \oplus h = (K \uplus K_1) \cup \{h\Downarrow\} = (K \cup \{h\Downarrow\}) \uplus K_1.$$

   It follows that $\mathsf{k}_{\ell_1,\ldots,\ell_k}(R',t) \Leftarrow \mathsf{k}_{\ell_1,\ldots,\ell_{i_1}}(X_1,x_1),\ldots,\mathsf{k}_{\ell_1,\ldots,\ell_{i_n}}(X_n,x_n) \notin \mathbf{deriv}(K|_{\mathsf{solved}})$ for any R' either (since $K \subseteq K \uplus K_1$). Therefore $K \oplus h = K \cup \{h\Downarrow\}$ and we immediately conclude by replacing $K \cup \{h\Downarrow\}$ by $K \oplus h$ in the equation above.

2. or $\mathsf{k}_{\ell_1,\ldots,\ell_k}(R',t) \Leftarrow \mathsf{k}_{\ell_1,\ldots,\ell_{i_1}}(X_1,x_1),\ldots,\mathsf{k}_{\ell_1,\ldots,\ell_{i_n}}(X_n,x_n) \in \mathbf{deriv}((K \uplus K_1)|_{\mathsf{solved}})$ for some R'. In this case, $(K \uplus K_1) \oplus h = (K \uplus K_1) \cup \{f\}$ where

   $$f = \Big(i_{\ell_1,\ldots,\ell_k}(R,R') \Leftarrow \{\mathsf{k}_{\ell_1,\ldots,\ell_{i_j}}(X_j,x_j)\}_{j\in\{1,\ldots,n\}}\Big)\}.$$

   To conclude we show the following claim.

   > If $\mathcal{N}ames(t) \cap \mathcal{M}_0 = \emptyset$ and
   >
   > $\mathsf{k}_{\ell_1,\ldots,\ell_k}(R',t) \Leftarrow \mathsf{k}_{\ell_1,\ldots,\ell_{i_1}}(X_1,x_1),\ldots,\mathsf{k}_{\ell_1,\ldots,\ell_{i_n}}(X_n,x_n) \in \mathbf{deriv}((K \uplus K_1)|_{\mathsf{solved}})$
   >
   > then
   >
   > $$\mathsf{k}_{\ell_1,\ldots,\ell_k}(R',t) \Leftarrow \mathsf{k}_{\ell_1,\ldots,\ell_{i_1}}(X_1,x_1),\ldots,\mathsf{k}_{\ell_1,\ldots,\ell_{i_n}}(X_n,x_n) \in \mathbf{deriv}(K|_{\mathsf{solved}})$$

   To proof this claim we proceed by induction on the size of the proof tree of

   $$\mathsf{k}_{\ell_1,\ldots,\ell_k}(R',t) \Leftarrow \mathsf{k}_{\ell_1,\ldots,\ell_{i_1}}(X_1,x_1),\ldots,\mathsf{k}_{\ell_1,\ldots,\ell_{i_n}}(X_n,x_n) \in \mathbf{deriv}((K \uplus K_1)|_{\mathsf{solved}}).$$

Base case: we need to consider two cases according to which rule has been applied.

- AXIOM: the rule does not depend on the knowledge base and we trivially conclude.
- RES: we have that $n = 0$, i.e., $(H \Leftarrow) \in (K \cup K_1)|_{\mathsf{solved}}$ and $H\sigma = \mathsf{k}_{\ell_1,\ldots,\ell_k}(R', t)$. As $\mathcal{N}ames(t) \cap \mathcal{M}_0 = \emptyset$ we have that $H \Leftarrow \in K|_{\mathsf{solved}}$. Hence, $(\mathsf{k}_{\ell_1,\ldots,\ell_k}(R', t) \Leftarrow) \in \mathbf{deriv}(K|_{\mathsf{solved}})$.

Inductive case: We suppose that the proof ends with an application of the RES rule. We have that $H \Leftarrow B_1, \ldots, B_m \ \in (K \cup K_1)|_{\mathsf{solved}}$, $B_i\sigma \Leftarrow \mathsf{k}_{\ell_1,\ldots,\ell_{i_1}}(X_1, x_1), \ldots, \mathsf{k}_{\ell_1,\ldots,\ell_{i_n}}(X_n, x_n) \in \mathbf{deriv}((K \uplus K_1)|_{\mathsf{solved}})$ and $H\sigma = \mathsf{k}_{\ell_1,\ldots,\ell_k}(R', t)$. Let $H = \mathsf{k}_u(S, t')$ and $B_i = \mathsf{k}_{u_i}(Y_i, y_i)$. As $H\sigma = \mathsf{k}_{\ell_1,\ldots,\ell_k}(R', t)$ and $\mathcal{N}ames(t) \cap \mathcal{M}_0 = \emptyset$, by inspection of the statements in K_1, it must be that $H \Leftarrow B_1, \ldots, B_m \ \in K|_{\mathsf{solved}}$. Moreover, as $t'\sigma = t$ we have by hypothesis that $\mathcal{N}ames(t'\sigma) \cap M_0 = \emptyset$ and hence $t' \cap \mathcal{N}ames(M_0) = \emptyset$. As $y_i \in \mathcal{V}ar(t')$ we have that $\mathcal{N}ames(y_i\sigma) \cap M_0 = \emptyset$ and we can apply our induction hypothesis to conclude that $B_i\sigma \Leftarrow \mathsf{k}_{\ell_1,\ldots,\ell_{i_1}}(X_1, x_1), \ldots, \mathsf{k}_{\ell_1,\ldots,\ell_{i_n}}(X_n, x_n) \in \mathbf{deriv}(K|_{\mathsf{solved}})$ for $1 \leq i \leq n$. Hence, as

$$H \Leftarrow B_1, \ldots, B_m \ \in K|_{\mathsf{solved}}$$

and

$$B_i\sigma \Leftarrow \mathsf{k}_{\ell_1,\ldots,\ell_{i_1}}(X_1, x_1), \ldots, \mathsf{k}_{\ell_1,\ldots,\ell_{i_n}}(X_n, x_n) \in \mathbf{deriv}(K|_{\mathsf{solved}})$$

for $1 \leq i \leq n$ we conclude that $(\mathsf{k}_{\ell_1,\ldots,\ell_k}(R', t) \Leftarrow \mathsf{k}_{\ell_1,\ldots,\ell_{i_1}}(X_1, x_1), \ldots, \mathsf{k}_{\ell_1,\ldots,\ell_{i_n}}(X_n, x_n)) \in \mathbf{deriv}(K)$.

□

The second helper lemma follows.

Lemma 5.5. *If K is a knowledge base such that $\mathcal{N}ames(K) \cap \mathcal{M}_0 = \emptyset$, $K_1 \subseteq K_{names}$ and*

$$K \uplus K_1 \Rightarrow K''$$

then $K'' = K' \uplus K_2$ with $K \Rightarrow^{=} K'$, $K_2 \subseteq K_{names}$ and $\mathcal{N}ames(K') \cap \mathcal{M}_0 = \emptyset$.

Proof. We perform a case distinction depending on which saturation rule triggered:

1. if rule RESOLUTION triggered, we will show that $f, g \in K$.

   Indeed, no statement $\left(\mathsf{k}(m, m) \Leftarrow \right) \in K_1$ can play the role of g in the RESOLUTION saturation rule since $t' = m$ must unify with $t \notin X$. Therefore t must be m, but $m \notin \mathcal{N}ames(K)$ by hypothesis and therefore t cannot be m.

   No statement in K_1 can play the role of f in the RESOLUTION saturation rule since they have no antecedents.

   Therefore $f, g \in K$ and $\mathcal{N}ames(h) \notin \mathcal{M}_0$. We choose $K' = K \oplus h$, $K_2 = K_1$ and we conclude by Lemma 5.4.

2. if rule EQUATION triggered, we distinguish three cases:

   (a) if a statement $\left(\mathsf{k}(m, m) \Leftarrow \right) \in K_1$ plays the role of f in the EQUATION saturation rule, we have that $t = m$. As t' unifies with m, we have that either $t' = m$ or that t' is a variable. The second case is not possible since g must be well-formed. Therefore $t' = m$. As $m \notin \mathcal{N}ames(K)$ by hypothesis it follows that $g \in K_1$ and therefore $g = \mathsf{k}(m, m)$. Therefore the resulting statement is $\mathsf{i}(m, m)$. We choose $K_2 = K_1 \cup \{\mathsf{i}(m, m)\}$, $K' = K$ to conclude.

   (b) if a statement $\left(\mathsf{k}(m, m) \Leftarrow \right) \in K_1$ plays the role of g, the reasoning is analogous to the case above

(c) otherwise $f, g \in K$. Therefore $\mathcal{N}ames(h) \cap \mathcal{M}_0 = \emptyset$. We choose $K' = K \oplus h$ and $K_2 = K_1$ to conclude.

3. if rule Test triggered, we distinguish two cases:

(a) if $\left(\mathsf{i}(m,m) \Leftarrow\right) \in K_1$ plays the role of f, we choose $K' = K$ and $K_2 = K_1 \cup \{\mathsf{ri}(m,m)\}$ to conclude.

(b) otherwise $f \in K$. The statement g must also be in K since g is a reachability statement and K_1 does not contain reachability statements. We choose $K' = K \oplus h$ and $K_2 = K_1$ to conclude.

□

We are now ready to prove Lemma 5.3.

Lemma 5.3. *Let T be a trace, let $\mathcal{M}_T = \{m \in \mathcal{M} \mid m \in \mathcal{N}ames(T)\}$ be the set of public names occurring in T and let $\mathcal{M}_{\bar{T}} = \mathcal{M} \setminus \mathcal{M}_T$ be the set of public names not occuring in T. Let*

$$K_{\mathcal{M}_{\bar{T}}} = \{\{\mathsf{k}(m,m) \Leftarrow\}_{m \in \mathcal{M}_{\bar{T}}} \cup \{\mathsf{i}(m,m) \Leftarrow\}_{m \in \mathcal{M}_{\bar{T}}} \cup \{\mathsf{ri}(m,m) \Leftarrow\}_{m \in \mathcal{M}_{\bar{T}}}\}.$$

Then $\mathsf{sat}(K_i(T)) = \mathsf{sat}(K_i(\mathit{Seed}(T, \mathcal{M}_T))) \uplus K_{\mathcal{M}_{\bar{T}}}$.

Proof. By Lemma 5.4, we have that

$$K_i(T) = K_i(\mathit{Seed}(T, \mathcal{M}_T)) \uplus \{k(m,m) \Leftarrow\}_{m \in \mathcal{M}_{\bar{T}}}.$$

Letting $\mathcal{M}_0 = \mathcal{M}_{\bar{T}}$ and iterating Lemma 5.5, we obtain that if

$$K_i(\mathit{Seed}(T, \mathcal{M}_T)) \uplus \{k(m,m) \Leftarrow\}_{m \in \mathcal{M}_0} \Rightarrow^* K'',$$

then $K'' = K' \uplus K_2$ with $K_i(\mathit{Seed}(T, \mathcal{M}_T)) \Rightarrow^* K'$, $\mathcal{N}ames(K') \cap \mathcal{M}_0 = \emptyset$ and $K_2 \subseteq K_{\bar{T}}$.

If K'' is saturated, K' and K_2 must be saturated as well. Therefore in the case K'' is saturated, we have $K_2 = K_{\mathcal{M}_{\bar{T}}}$ and $K' = \mathsf{sat}(K_i(\mathit{Seed}(T, \mathcal{M}_T)))$. We obtained therefore that $\mathsf{sat}(K_i(T)) = \mathsf{sat}(K_i(\mathit{Seed}(T, \mathcal{M}_T))) \uplus K_{\mathcal{M}_{\bar{T}}}$, which is what we wanted to show.

□

Therefore, to obtain the saturated knowledge base of the initial knowledge base $K_i(T)$ of T, it is sufficient to saturate starting from the initial knowledge $K_i(\mathit{Seed}(T, \mathcal{M}_T))$ of $\mathit{Seed}(T, \mathcal{M}_T)$ (which is finite) and add the statements in $K_{\mathcal{M}_{\bar{T}}}$ at the end.

5.5.3 Checking for Equivalence

We have shown how to saturate an initial knowledge base associated to a trace in a sound and complete manner. We will now show how to use the saturated knowledge base associated to a trace in order to prove trace inclusion for determinate processes. Trace equivalence can be obtained by verifying that trace inclusion holds in both directions. The procedure for checking trace inclusion for determinate processes directly follows from Theorem 5.4.

Theorem 5.4. *Let T be a trace and let P be a determinate process. Let K be the set of solved statements from a saturated knowledge base associated to T. Then $T \sqsubseteq_c P$ iff the following tests hold:*

$$\textsc{Reachability}\ \frac{\begin{array}{c}\left(\mathsf{r}_{l_1,\ldots,l_n} \Leftarrow \{\mathsf{k}_{w_i}(X_i, x_i)\}_{i\in\{1,\ldots,m\}}\right) \in K \\ c_1,\ldots,c_k \textit{ fresh constants} \qquad \sigma : \mathcal{V}ar(l_1,\ldots,l_n) \to \{c_1,\ldots,c_k\} \textit{ is a bijection} \\ \mathsf{k}_{l_1\sigma,\ldots,l_{i-1}\sigma}(R_i, t_i\sigma) \in \mathcal{H}(K) \textit{ for all } i \textit{ such that } l_i = \textbf{\textit{receive}}(d_i, t_i) \\ M_i = l_i \textit{ if } l_i \in \{\textbf{\textit{test}}, \textbf{\textit{send}}(\_)\} \\ M_i = \textbf{\textit{receive}}(d_i, R_i) \textit{ if } l_i = \textbf{\textit{receive}}(d_i, t_i)\end{array}}{(P,\emptyset) \xRightarrow{M_1,\ldots,M_n} (T',\varphi)}$$

$$\textsc{Identity}\ \frac{\begin{array}{c}\left(\mathsf{ri}_{l_1,\ldots,l_n}(R, R') \Leftarrow \{\mathsf{k}_{w_i}(X_i, x_i)\}_{i\in\{1,\ldots,m\}}\right) \in K \\ c_1,\ldots,c_k \textit{ fresh constants} \qquad \sigma : \mathcal{V}ar(l_1,\ldots,l_n) \to \{c_1,\ldots,c_k\} \textit{ is a bijection} \\ \mathsf{k}_{l_1\sigma,\ldots,l_{i-1}\sigma}(R_i, t_i\sigma) \in \mathcal{H}(K) \textit{ for all } i \textit{ such that } l_i = \textbf{\textit{receive}}(t_i) \\ M_i = l_i \textit{ if } l_i \in \{\textbf{\textit{test}}, \textbf{\textit{send}}(\_)\} \qquad M_i = \textbf{\textit{receive}}(d_i, R_i) \textit{ if } l_i = \textbf{\textit{receive}}(d_i, t_i)\end{array}}{(P,\emptyset) \xRightarrow{M_1,\ldots,M_n} (T',\varphi) \textit{ such that } (R\omega = R'\omega)\varphi \textit{ where } \omega = \{X_i \mapsto x_i\sigma\}}$$

Note that the infinity of statements $K_{\mathcal{M}_{\bar{T}}}$ which are guaranteed by Lemma 5.3 to be in the saturated knowledge base K do not need to be checked since they will trivially pass. It is therefore sufficient to check the (hopefully finite number of) statements from $\mathsf{sat}(\mathit{Seed}(T, \mathcal{M}_T))$.

The full proof of the above theorem is in Section 5.8. Theorem 5.4 directly implies that the following algorithm for checking if $T \subseteq P$ for a trace T and a determinate process P is correct:

1. let $K = \mathsf{sat}(K_i(T))|_{\mathsf{solved}} \setminus K_{\mathcal{M}_{\bar{T}}}$ denote the interesting solved statements in the saturated knowledge base associated to T (the statements in $K_{\mathcal{M}_{\bar{T}}}$ trivially pass all tests and therefore do not have to be checked); by Lemma 5.3, we have that $K = \mathsf{sat}(K_i(\mathit{Seed}(T, \mathcal{M}_T)))|_{\mathsf{solved}}$.
2. for every reachability statement $\mathsf{r}_{l_1,\ldots,l_n} \Leftarrow \{\mathsf{k}_{w_i}(X_i, x_i)\}_{1\le i\le m}$ in K:
   (a) choose fresh constants $c_1,\ldots,c_k$, one for each variable in $\mathcal{V}ar(l_1,\ldots,l_k)$ and a bijection σ between $\{c_1,\ldots,c_k\}$ and $\mathcal{V}ar(l_1,\ldots,l_k)$
   (b) for every i such that $l_i = \mathbf{receive}(d_i, t_i)$, find an R_i such that $\mathsf{k}_{l_1\sigma,\ldots,l_{i-1}\sigma}(R_i, t_i) \in \mathcal{H}(K)$ (the existence of R_i is guaranteed by the soundness and completeness of the saturation procedure). Furthermore, a simple recursive algorithm that finds R_i follows from the definition of $\mathcal{H}(K)$.
   (c) let $M_i = l_i$ if $l_i \in \{\mathbf{test}, \mathbf{send}(d_i)\}$ and $M_i = \mathbf{receive}(d_i, R_i)$ if $l_i = \mathbf{receive}(d_i, t_i)$
   (d) test if $(P,\emptyset) \xRightarrow{M_1,\ldots,M_n} (T',\varphi)$ for some T' and some φ. If the above transition does not take place, return 'no'; otherwise continue.
3. for every reachable identity statement $\mathsf{ri}_{l_1,\ldots,l_n}(R, R') \Leftarrow \{\mathsf{k}_{w_i}(X_i, x_i)\}_{i\in\{1,\ldots,m\}}$ in K:
   (a) choose fresh constants $c_1,\ldots,c_k$, one for each variable in $\mathcal{V}ar(l_1,\ldots,l_k)$ and a bijection σ between $\{c_1,\ldots,c_k\}$ and $\mathcal{V}ar(l_1,\ldots,l_k)$
   (b) for every i such that $l_i = \mathbf{receive}(d_i, t_i)$, find an R_i such that $\mathsf{k}_{l_1\sigma,\ldots,l_{i-1}\sigma}(R_i, t_i) \in \mathcal{H}(K)$ (the existence of R_i is guaranteed by the soundness and completeness of the saturation procedure)
   (c) let $M_i = l_i$ if $l_i \in \{\mathbf{test}, \mathbf{send}(d_i)\}$ and $M_i = \mathbf{receive}(d_i, R_i)$ if $l_i = \mathbf{receive}(d_i, t_i)$
   (d) test if $(P,\emptyset) \xRightarrow{M_1,\ldots,M_n} (T',\varphi)$ for some T' and some φ. If the above transition does not take place, return 'no'; otherwise continue.

(e) test if $(R\omega = R'\omega)\varphi$, where $\omega = \{X_i \mapsto x_i\sigma\}$. If the test fails, return 'no'; otherwise continue.

4. if all the above tests worked, return 'yes'.

Example 5.11. We recall the traces in our running example:

$$T = \mathbf{send}(c, \mathrm{enc}(a,k)).\mathbf{send}(c, \mathrm{enc}(a',k)).\mathbf{receive}(c,x).[x \stackrel{?}{=} \mathrm{enc}(\mathrm{dec}(x,k),k)].\mathbf{send}(c,ok)$$
$$S_1 = \mathbf{send}(c, \mathrm{enc}(a,k)).\mathbf{send}(c, \mathrm{enc}(a',k)).\mathbf{receive}(c,x).[x \stackrel{?}{=} \mathrm{enc}(a,k)].\mathbf{send}(c,ok)$$
$$S_2 = \mathbf{send}(c, \mathrm{enc}(a,k)).\mathbf{send}(c, \mathrm{enc}(a',k)).\mathbf{receive}(c,x).[x \stackrel{?}{=} \mathrm{enc}(a',k)].\mathbf{send}(c,ok)$$

We have that the process $\{S_1, S_2\}$ is determinate. It is true that T is trace included in $\{S_1, S_2\}$ and that $\{S_1, S_2\}$ is trace-included in T. This means that T is trace equivalent to $\{S_1, S_2\}$. While we cannot reproduce here all the tests performed by the algorithm that we described above, we can show that some interesting reachability statements found in the saturated knowledge base of T perform successfully in $\{S_1, S_2\}$. Indeed, the saturated knowledge base of T contains the statements

$$f_1 = \left(\mathsf{r}_{\mathbf{send}(c),\mathbf{send}(c),\mathbf{receive}(c,\mathrm{enc}(a,k)),\mathbf{test},\mathbf{send}(c)}\right) \text{ and}$$

$$f_2 = \left(\mathsf{r}_{\mathbf{send}(c),\mathbf{send}(c),\mathbf{receive}(c,\mathrm{enc}(a',k)),\mathbf{test},\mathbf{send}(c)}\right).$$

We have $\mathsf{k}_{\mathbf{send}(c),\mathbf{send}(c)}(w_1, \mathrm{enc}(a,k)) \in \mathcal{H}(K)$ and therefore we have to check, by the first type of test performed by the algorithm, if $(\{S_1, S_2\}, \emptyset) \xRightarrow{\mathbf{send}(c),\mathbf{send}(c),\mathbf{receive}(c,w_1),\mathbf{test},\mathbf{send}(c)} (T', \varphi)$ for some trace T' and some frame φ. This is indeed the case, since the above run is a valid run of the first trace $S_1 \subseteq \{S_1, S_2\}$ of the process $\{S_1, S_2\}$.

Similarly, we have $\mathsf{k}_{\mathbf{send}(c),\mathbf{send}(c)}(w_2, \mathrm{enc}(a',k)) \in \mathcal{H}(K)$ and therefore we have to check, by the first type of test performed by the algorithm, if

$$(\{S_1, S_2\}, \emptyset) \xRightarrow{\mathbf{send}(c),\mathbf{send}(c),\mathbf{receive}(c,w_2),\mathbf{test},\mathbf{send}(c)} (T', \varphi)$$

for some trace T'' and some frame φ'. This is indeed the case, since the above run is a valid run of the second trace $S_2 \subseteq \{S_1, S_2\}$ of the process $\{S_1, S_2\}$.

5.6 Soundness of Saturation

This section is dedicated to proving the lemmas needed to prove Theorem 5.2, i.e. the soundness of the saturation procedure. We first need a small helper lemma.

Lemma 5.6 (Monotonicity of the knowledge predicate)**.** *If $T \models \mathsf{k}_u(R,t)$ then $T \models \mathsf{k}_{uv}(R,t)$.*

Proof. Immediate because of the 'soft' semantics of k.

□

We now show that the canonicalization rules are sound.

Lemma 5.7 (Soundness of canonicalization)**.** *If $T \models f$ then $T \models f{\Downarrow}$.*

Proof. We will show that each canonicalization rule is sound:

1. For the RENAME rule, consider a statement

$$f = \Big(H \Leftarrow \mathsf{k}_{t_1,\ldots,t_k}(X,x), \mathsf{k}_{t_1,\ldots,t_l}(Y,x), B_1, \ldots, B_n\Big)$$

where $k \leq l$ and we show that if $T \models f$ then $T \models g$ where

$$g = \Big((H \Leftarrow \mathsf{k}_{t_1,\ldots,t_k}(X,x), B_1, \ldots, B_n)\{Y \mapsto X\}\Big)$$

Let τ be a grounding substitution for g such that $T \models \mathsf{k}_{t_1,\ldots,t_k}(X,x)\{Y \mapsto X\}\tau$, $B_1\{Y \mapsto X\}\tau, \ldots, B_n\{Y \mapsto X\}\tau$. We show that if $T \models f$ then $T \models H\{Y \mapsto X\}\tau$.

Let τ' be a substitution identical to τ, except for $\tau'(Y) = \tau(X)$. We will show that all the antecedents in $f\tau'$ are true in T.

Indeed, $\mathsf{k}_{t_1,\ldots,t_k}(X,x)\tau' = \mathsf{k}_{t_1,\ldots,t_k}(X,x)\{Y \mapsto X\}\tau$ holds by hypothesis. As $k \leq l$ and $T \models \mathsf{k}_{t_1,\ldots,t_k}(X,x)\tau'$, we have by Lemma 5.6 that $T \models \mathsf{k}_{t_1,\ldots,t_l}(X,x)\tau'$ and by the choice of τ', we have $\mathsf{k}_{t_1,\ldots,t_l}(X,x)\tau' = \mathsf{k}_{t_1,\ldots,t_l}(Y,x)\tau'$. Furthermore $T \models B_1\tau' = B_1\{Y \mapsto X\}\tau, \ldots, B_n\tau' = B_n\{Y \mapsto X\}\tau$ by hypothesis. As $T \models f$, and all antecedents of $f\tau'$ are true in T, we obtain that $T \models H\tau'$.

But $H\tau' = H\{Y \mapsto X\}\tau$ and therefore we have that $T \models H\{Y \mapsto X\}\tau$. As we have chosen τ arbitrarily, it follows that $T \models g$.

2. For the REMOVE rule, consider a solved statement

$$f = \Big(H \Leftarrow \mathsf{k}_{t_1,\ldots,t_k}(X,x), B_1, \ldots, B_n\Big)$$

such that the rule RENAME does not apply to f and such that $x \notin \mathcal{V}ar(H)$. We show that if $T \models f$ then $T \models g$ where

$$g = \Big(H \Leftarrow B_1, \ldots, B_n\Big)$$

Let τ be an arbitrary substitution such that $T \models B_1\tau, \ldots, B_n\tau$. We will show that $T \models H\tau$ and hence $T \models g$.

Let $(T_1, \varphi_1) = (T, \emptyset)$. We distinguish between two cases:

   (a) if

$$(T_1, \varphi_1) \xrightarrow{L_1} (T_2, \varphi_2) \xrightarrow{L_2} \ldots \xrightarrow{L_k} (T_{k+1}, \varphi_{k+1})$$

   such that $L_i\varphi_i = t_i\tau$ for all $1 \leq i \leq k$, we consider the substitution τ' to be identical to τ except for $\tau'(x) = (X\tau)\varphi_{k+1}$.

   As $x \notin \mathcal{V}ar(H)$ and because f is solved and the rule RENAME does not apply, we have that $x \notin \mathcal{V}ar(B_1, \ldots, B_n)$ and therefore $T \models B_1\tau' = B_1\tau, \ldots, B_n\tau' = B_n\tau$.

   Furthermore, we have that $T \models \mathsf{k}_{t_1,\ldots,t_k}(X,x)\tau'$ by the definition of k.

   As all antecedents of $f\tau'$ are true in T and $T \models f$, it follows that $T \models H\tau'$. But $H\tau = H\tau'$ since $x \notin \mathcal{V}ar(H)$ and therefore $T \models H\tau$.

   (b) otherwise, we trivially have that $T \models \mathsf{k}_{t_1,\ldots,t_k}(X,x)\tau$. We have that all antecedents of $f\tau$ are true in T and therefore, as $T \models f$, it follows that $T \models H\tau$.

   We have shown that $T \models g$, therefore the rule REMOVE is sound.

We have shown that both rules for computing the canonical form are sound and therefore $T \models f{\Downarrow}$ whenever $T \models f$.

□

Next we show that the consequence rules are sound.

Lemma 5.8 (Soundness of the consequence)**.** *If for all $f \in K$ we have that $T \models f$, then for all $f \in \mathbf{deriv}(K)$ we have that $T \models f$.*

Proof. We show that both inference rules are sound.

For the Axiom rule, soundness follows immediately from the semantics of k.

For the Res rule, let

$$f = \Big(H \Leftarrow B_1, \ldots, B_n\Big)$$

and

$$g_i = \Big(B_i\sigma \Leftarrow C_1, \ldots, C_m\Big)$$

for $1 \leq i \leq n$ be statements such that $T \models f$ and $T \models g_i$ $(1 \leq i \leq n)$.

We will show that

$$T \models \Big(H\sigma \Leftarrow C_1, \ldots, C_m)$$

by letting τ be a substitution such that $T \models C_1\tau, \ldots, C_m\tau$ and proving that $T \models H\sigma\tau$.

Indeed, as $T \models C_1\tau, \ldots, C_m\tau$ and as $T \models g_i$ $(1 \leq i \leq n)$, we have that $T \models B_i\sigma\tau$ $(1 \leq i \leq n)$. But $T \models f$ and therefore $T \models H\sigma\tau$ as well, which is what we had to show.

□

We now show that each rule of the saturation procedure is sound.

Lemma 5.9 (Soundness of the resolution saturation rule)**.** *Let f, g and h be defined as in the* Resolution *rule. If $T \models f$ and $T \models g$ then $T \models h$.*

Proof. We consider the following statements:

$$\begin{aligned} f &= \Big(H \Leftarrow \mathsf{k}_{\ell_1,\ldots,\ell_i}(X,t), B_1, \ldots, B_n\Big) \\ g &= \Big(\mathsf{k}_{\ell'_1,\ldots,\ell'_j}(R,t') \Leftarrow B_{n+1}, \ldots, B_m\Big) \\ h &= \Big((H \Leftarrow B_1, \ldots, B_m)\sigma\Big) \end{aligned}$$

with $j \leq i$ and where $\sigma = \mathsf{mgu}(\mathsf{k}_{\ell'_1,\ldots,\ell'_j}(R,t'), \mathsf{k}_{\ell_1,\ldots,\ell_j}(X,t))$. We will show that if $T \models f$ and $T \models g$ then $T \models h$.

Indeed, let τ be an arbitrary substitution grounding for h and assume that $T \models B_1\sigma\tau, \ldots, B_m\sigma\tau$. We will show that $T \models H\sigma\tau$.

As $T \models B_{n+1}\sigma\tau, \ldots, B_m\sigma\tau$ and because $T \models g$, we have that $T \models \mathsf{k}_{\ell'_1,\ldots,\ell'_j}(R,t')\sigma\tau$. But $\mathsf{k}_{\ell'_1,\ldots,\ell'_j}(R,t')\sigma\tau = \mathsf{k}_{\ell_1,\ldots,\ell_j}(X,t)\sigma\tau$ by choice of $\sigma = \mathsf{mgu}(\mathsf{k}_{\ell'_1,\ldots,\ell'_j}(R,t'), \mathsf{k}_{\ell_1,\ldots,\ell_j}(X,t))$.

As $j \leq i$, it follows by Lemma 5.6 that $T \models \mathsf{k}_{\ell_1,\ldots,\ell_i}(X,t)\sigma\tau$ as well. As all antecedents of $f\sigma\tau$ are true in T and because $T \models f$, we have that $T \models H\sigma\tau$.

As τ was chosen arbitrarily, it follows that $T \models h$. □

Lemma 5.10 (Soundness of the equational saturation rule)**.** *Let f, g and h be defined as in the* Equation *rule. If $T \models f$ and $T \models g$ then $T \models h$.*

Proof. We consider the following statements:

$$\begin{aligned} f &= \Big(\mathsf{k}_u(R,t) \Leftarrow B_1, \ldots, B_n\Big) \\ g &= \Big(\mathsf{k}_{u'v'}(R',t') \Leftarrow B_{n+1}, \ldots, B_m\Big) \\ h &= \Big((\mathsf{i}_{u'v'}(R,R') \Leftarrow B_1, \ldots, B_m)\sigma\Big) \end{aligned}$$

where $\sigma = \mathsf{mgu}(\mathsf{k}_u(\_,t), \mathsf{k}'_u(\_,t'))$.

We will show that if $T \models f$ and $T \models g$ then $T \models h$. Let τ be an arbitrary substitution grounding for h. We assume that $T \models B_1\sigma\tau, \ldots, B_m\sigma\tau$ and we show that $T \models \mathsf{i}_{u'v'}(R,R')\sigma\tau$.

As $T \models B_1\sigma\tau, \ldots, B_n\sigma\tau$ and because $T \models f$ we have that $T \models \mathsf{k}_u(R,t)\sigma\tau$. By monotonicity of k (Lemma 5.6) we also have that $T \models \mathsf{k}_{u'v'}(R,t)\sigma\tau$.

As $T \models B_1\sigma\tau \ldots, B_n\sigma\tau$ and because $T \models g$ we also obtain that $T \models \mathsf{k}_{u'v'}(R',t')\sigma\tau$.

But this is exactly the definition of $T \models \mathsf{i}_{u'v'}(R,R')\sigma\tau$.

We have shown that the head of $h\tau$ is true in T. As τ was chosen arbitrarily, it follows that h holds in T.

□

Lemma 5.11 (Soundness of the test saturation rule)**.** *Let f, g, h be statements as in the* Test *saturation rule. If $T \models f$ and $T \models g$ then $T \models h$.*

Proof. We consider the following statements:

$$\begin{aligned} f &= \Big(\mathsf{i}_u(R,R') \Leftarrow B_1, \ldots, B_n\Big) \\ g &= \Big(\mathsf{r}_{u'v'} \Leftarrow B_{n+1}, \ldots, B_m\Big) \\ h &= \Big((\mathsf{ri}_{u'v'}(R,R') \Leftarrow B_1, \ldots, B_m)\sigma\Big) \end{aligned}$$

where $\sigma = \mathsf{mgu}(u,u')$.

Let τ be an arbitrary substitution grounding for h. We assume that $T \models B_1\sigma\tau, \ldots, B_m\sigma\tau$ and we show that $T \models \mathsf{ri}_u(R,R')\tau$. Indeed, as $T \models B_1\sigma\tau, \ldots, B_n\sigma\tau$ and as $T \models f$, we have that

$$T \models \mathsf{i}_u(R,R')\sigma\tau. \tag{5.1}$$

As $T \models B_{n+1}\sigma\tau, \ldots, B_m\sigma\tau$ and as $T \models g$, we have that

$$T \models \mathsf{r}_{u'v'}\sigma\tau. \tag{5.2}$$

But $\sigma = \mathsf{mgu}(u,u')$ and therefore $u\sigma\tau = u'\sigma\tau$. Therefore, by Equations (5.1) and (5.2), we immediately obtain $T \models \mathsf{ri}_{u'v'}(R,R')\sigma\tau$, which is what we wanted. As τ was chosen arbitrarily, it follows that $T \models h$. □

We have shown soundness of all saturation rules. Now we prove soundness of the $\oplus$ operator.

Lemma 5.12 (Soundness of the update)**.** *If for all $f \in K$ we have that $T \models f$ and if $T \models g$, then for any $f \in (K \oplus g)$ we have that $T \models f$.*

Proof. If $K \oplus g = K \cup \{g\Downarrow\}$, we immediately conclude by Lemma 5.8. Otherwise, it must be that

$$g\Downarrow = \Big(\mathsf{k}_{\ell_1,\ldots,\ell_k}(R,t) \Leftarrow \mathsf{k}_{\ell_1,\ldots,\ell_{i_1}}(X_1,x_1), \ldots, \mathsf{k}_{\ell_1,\ldots,\ell_{i_n}}(X_n,x_n)\Big)$$

for some $R, t, \ell_1, \ldots, \ell_k, i_1, \ldots, i_n, X_1, \ldots, X_n, x_1, \ldots, x_n$ and $K \oplus g = K \cup \{h\}$, where

$$h = \Big(\mathsf{i}_{\ell_1,\ldots,\ell_k}(R,R') \Leftarrow \mathsf{k}_{\ell_1,\ldots,\ell_{i_1}}(X_1,x_1), \ldots, \mathsf{k}_{\ell_1,\ldots,\ell_{i_n}}(X_n,x_n)\Big)$$

and where

$$g' = \Big(\mathsf{k}_{\ell_1,\ldots,\ell_k}(R',t) \Leftarrow \mathsf{k}_{\ell_1,\ldots,\ell_{i_1}}(X_1,x_1), \ldots, \mathsf{k}_{\ell_1,\ldots,\ell_{i_n}}(X_n,x_n)\Big) \in \mathbf{deriv}(K|_{\mathsf{solved}}).$$

It is sufficient to show that $T \models h$. As $K|_{\mathsf{solved}} \subseteq K$, it immediately follows that $g' \in \mathbf{deriv}(K)$ and, by Lemma 5.8, $T \models g'$.

We now show that $T \models h$. Let τ be an arbitrary substitution grounding for h such that the antecedents of $h\tau$ are true in T. As the antecedents of $h\tau$ are the same as the antecedents of $g\Downarrow\tau$ and those of $g'\tau$, and as $T \models g$ and $T \models g'$ we have that $T \models \mathsf{k}_{\ell_1,\ldots,\ell_k}(R,t)\tau$ and $T \models \mathsf{k}_{\ell_1,\ldots,\ell_k}(R',t)\tau$.

But this immediately implies that $T \models \mathsf{i}_{\ell_1,\ldots,\ell_k}(R,R')\tau$ (the head of $h\tau$). As τ was chosen arbitrarily, it follows that $T \models h$.

□

We are now ready to prove Theorem 5.2.

Theorem 5.2 (soundness of saturation)**.** *If* $f \in \mathsf{sat}(K_i(T))$ *then* $T \models f$.

Proof. Immediate by Lemma 5.1 to Lemma 5.12.

□

5.7 Completeness of Saturation

This section is dedicated to proving the lemmas needed to prove Theorem 5.3, i.e. the completeness of the saturation procedure. We first show completeness for statements which are consequences of the knowledge base.

Proposition 5.2. *Let* K *be a knowledge base, let* $f = \left(H' \Leftarrow C_1, \ldots, C_m\right)$ *be a statement such that* $f \in \mathbf{deriv}(K)$ *and let* τ *be a substitution grounding for* f *such that* $C_i\tau \in \mathcal{H}(K)$ *for all* $1 \leq i \leq n$*. Then* $H'\tau \in \mathcal{H}(K)$*.*

Proof. Induction on the proof tree of $f \in \mathbf{deriv}(K)$.

If the Axiom rule was used, then we have $H\sigma \in \mathcal{H}(K)$ by the ExtendK rule.

If the Res rule was used, we have that there exists $\left(H \Leftarrow B_1, \ldots, B_n\right) \in K$ and a substitution σ such that $H' = H\sigma$ and $B_i\sigma \Leftarrow C_1, \ldots, C_m \in \mathbf{deriv}(K)$ $(1 \leq i \leq n)$.

By the induction hypothesis, we have that $B_i\sigma\tau \in \mathcal{H}(K)$. As $\left(H \Leftarrow B_1, \ldots, B_n\right) \in K$, it follows that $H\sigma\tau = H'\tau \in \mathcal{H}(K)$, which is what we had to show.

□

We need a technical definition that will be useful in inductive proofs.

Definition 5.12. Let $\mathcal{S}(H, K)$ be the *size* of the smallest proof tree of $H \in \mathcal{H}(K)$ (defined only when $H \in \mathcal{H}(K)$).

Before proceeding, we show the following technical result.

Proposition 5.3. *Let* K *be a knowledge base. If* $\mathsf{k}_w(R, t) \in \mathcal{H}_\mathsf{e}(K)$ *and* $\mathsf{i}_w(R, R') \in \mathcal{H}_\mathsf{e}(K)$*, then*

$$\mathsf{k}_w(R', t) \in \mathcal{H}_\mathsf{e}(K).$$

Proof. As $\mathsf{k}_w(R, t) \in \mathcal{H}_\mathsf{e}(K)$, it follows that there exist R'' such that

$$\mathsf{k}_w(R'', t) \in \mathcal{H}(K) \tag{5.3}$$

and such that $\mathsf{i}_w(R, R'') \in \mathcal{H}_\mathsf{e}(K)$. But $\mathsf{i}_w(R, R') \in \mathcal{H}_\mathsf{e}(K)$ and therefore, by the symmetry and transitivity of $\mathsf{i}_w(\_, \_)$, we have that

$$\mathsf{i}_w(R'', R') \in \mathcal{H}_\mathsf{e}(K). \tag{5.4}$$

Using Equations 5.3 and 5.4 we immediately obtain by the definition of $\mathcal{H}_\mathsf{e}$ that $\mathsf{k}_w(R', T) \in \mathcal{H}_\mathsf{e}(K)$.

□

The following definition and helper lemma help characterize the facts in $\mathcal{H}(K)$.

Definition 5.13. We write $w \sqsubseteq w'$ whenever w is a prefix of w': i.e. there exists $\ell_1, \ldots, \ell_n$ such that $w' = \ell_1, \ldots, \ell_n$ and $w = \ell_1, \ldots, \ell_m$ for some $0 \leq m \leq n$.

Proposition 5.4. *If* $\mathsf{k}_w(R,t) \in \mathcal{H}(K)$ *(resp.* $\mathsf{i}_w(R,S) \in \mathcal{H}(K)$*) then there exist a statement* $f = \left(\mathsf{k}_{w'}(R',t') \Leftarrow B_1,\ldots,B_m\right) \in K$ *(resp.* $f = \left(\mathsf{i}_{w'}(R',S') \Leftarrow B_1,\ldots,B_m\right) \in K$*) and a substitution* σ *such that* $R'\sigma = R$*,* $t'\sigma = t$ *(resp.* $S'\sigma = S$*),* $w'\sigma \sqsubseteq w$*,* $B_i\sigma \in \mathcal{H}(K)$ *for all* $1 \leq i \leq m$ *and* $\sum_{1\leq i\leq m} \mathcal{S}(B_i\sigma,K) < \mathcal{S}(\mathsf{k}_w(R,t),K)$ *(resp.* $\sum_{1\leq i\leq m} \mathcal{S}(B_i\sigma,K) < \mathcal{S}(\mathsf{i}_w(R,S),K)$*).*

Proof. We prove the proposition by induction on the smallest proof tree of $H = \mathsf{k}_w(R,t) \in \mathcal{H}(K)$ (resp. $H = \mathsf{i}_w(R,S) \in \mathcal{H}(K)$). We proceed by case distinction on the last proof rule that has been applied.

- Simple Consequence: In this case we have that there exist a statement $f = \left(H' \Leftarrow B_1,\ldots,B_m\right) \in K$ and a substitution σ such that $H'\sigma = H$, $B_i\sigma \in \mathcal{H}(K)$ for all $1 \leq i \leq m$ and $\sum_{1\leq i\leq m} \mathcal{S}(B_i\sigma,K) + 1 = \mathcal{S}(\mathsf{k}_{w'}(R',t')\sigma,K)$. Hence we directly conclude.
- ExtendK: In this case $H = \mathsf{k}_w(R,t)$ and we have that $w = uv$ for some u, v and $\mathsf{k}_u(R,t) \in \mathcal{H}(K)$. By induction hypothesis, we have that there exists $f = \mathsf{k}_{u'}(R',t') \Leftarrow B_1,\ldots,B_m \in K$ and σ such that $R'\sigma = R$, $t'\sigma = t$, $u'\sigma \sqsubseteq u$, $B_i\sigma \in \mathcal{H}(K)$ for all $1 \leq i \leq m$ and $\sum_{1\leq i\leq m} \mathcal{S}(B_i\sigma,K) < \mathcal{S}(\mathsf{k}_w(R,t),K)$. As $u \sqsubseteq w$, we also have that $u'\sigma \sqsubseteq w$. Moreover, $\mathcal{S}(\mathsf{k}_w(R,t),\mathcal{H}(K)) = \mathcal{S}(\mathsf{k}_u(R,t),\mathcal{H}(K)) + 1 > \sum_{1\leq i\leq m} \mathcal{S}(B_i\sigma,K)$ which allows us to conclude.
- ExtendI: In this case $H = \mathsf{i}_w(R,S)$ and we have that $w = uv$ for some u, v and $\mathsf{i}_u(R,S) \in \mathcal{H}(K)$. By induction hypothesis, we have that there exists $f = \mathsf{i}_{u'}(R',S') \Leftarrow B_1,\ldots,B_m \in K$ and σ such that $R'\sigma = R$, $S'\sigma = S$, $u'\sigma \sqsubseteq u$, $B_i\sigma \in \mathcal{H}(K)$ for all $1 \leq i \leq m$ and $\sum_{1\leq i\leq m} \mathcal{S}(B_i\sigma,K) < \mathcal{S}(\mathsf{k}_w(R,t),K)$. As $u \sqsubseteq w$, we also have that $u'\sigma \sqsubseteq w$. Moreover, $\mathcal{S}(\mathsf{i}_w(R,S),\mathcal{H}(K)) = \mathcal{S}(\mathsf{i}_u(R,S),\mathcal{H}(K)) + 1 > \sum_{1\leq i\leq m} \mathcal{S}(B_i\sigma,K)$ which allows us to conclude.

□

We are now ready to show completeness with respect to equational statements.

Lemma 5.13. *Let* K *be a saturated knowledge base, let* $f \in K$ *be a statement*

$$f = \left(\mathsf{i}_w(R,R') \Leftarrow B_1,\ldots,B_n\right)$$

and let σ *be a substitution grounding for* f *such that* $B_i\sigma \in \mathcal{H}(K|_{\mathsf{solved}})$ *for all* $1 \leq i \leq n$*. Then we have that*

$$(\mathsf{i}_w(R,R'))\sigma \in \mathcal{H}(K|_{\mathsf{solved}}).$$

Proof. Let $\mathcal{G} = \sum_{i\in\{1,\ldots,n\}} \mathcal{S}(B_i\sigma,K|_{\mathsf{solved}})$. We prove the lemma by induction on $\mathcal{G}$. If f is a solved statement, the conclusion is immediate by the definition of $\mathcal{H}$.

Otherwise, if f is not a solved statement, there exists some B_j $(1 \leq j \leq n)$ such that $B_j = \mathsf{k}_{w_j}(X_j,t_j)$ and $t_j \notin \mathcal{X}$.

As $B_j\sigma \in \mathcal{H}(K|_{\mathsf{solved}})$, it follows by Proposition 5.4 that $w_j = u_jv_j$ for some u_j, v_j and that there exists

$$g = \left(\mathsf{k}_{u'_j}(R'_j,t'_j) \Leftarrow B_{n+1},\ldots,B_m\right) \in K|_{\mathsf{solved}}$$

and a substitution σ' grounding for g such that $B_{n+1}\sigma',\ldots,B_m\sigma' \in \mathcal{H}(K|_{\mathsf{solved}})$, $R'_j\sigma' = X_j\sigma$, $t'_j\sigma' = t_j\sigma$, $u'_j\sigma' = u_j\sigma$ and $\mathcal{S}(B_j\sigma) = 1 + \sum_{i\in\{n+1,\ldots,m\}} \mathcal{S}(B_i\sigma')$.

As $\omega = \sigma \cup \sigma'$ is a unifier of $H = \mathsf{k}_{u'_j}(R'_j,t'_j)$ and $\mathsf{k}_{u_j}(X_j,t_j)$, it follows that the two terms are unifiable. Let $\tau = \mathsf{mgu}(H,\mathsf{k}_{u_j}(X_j,t_j))$ denote their most general unifier. As K is saturated, it follows that the Resolution saturation rule was applied to f and g and therefore the resulting equational statement

$$h = \left(\mathsf{i}_w(R,R') \Leftarrow B_1,\ldots,B_{j-1},B_{j+1},\ldots,B_m\right)\tau \in K$$

must be in K (by the update function, equational statements are added to the knowledge base).

As ω is a unifier of H and $\mathsf{k}_{u_j}(X_j, t_j)$ and as $\tau = \mathsf{mgu}(H, \mathsf{k}_{u_j}(X_j, t_j))$, it follows that there exists ω' such that $\omega = \tau\omega'$. We have that ω' is a substitution grounding for h, that

$$B_i\tau\omega' \in \mathcal{H}(K|_{\mathsf{solved}})$$

for $i \in \{1, \ldots, j-1, j+1, \ldots, m\}$ and that $\sum_{i\in\{1,\ldots,j-1,j+1,\ldots,m\}} \mathcal{S}(B_i\tau\omega', K|_{\mathsf{solved}}) \leq \mathcal{G} - 1$.

Therefore we can apply the induction hypothesis to h and ω' and conclude.

□

We now prove completeness with respect to the intruder reachable identity predicate.

Lemma 5.14. *Let K be a saturated knowledge base, let $f \in K$ be a statement*

$$f = \Big(\mathsf{ri}_w(R, R') \Leftarrow B_1, \ldots, B_n\Big)$$

and let σ be a substitution grounding for f such that $B_i\sigma \in \mathcal{H}(K|_{\mathsf{solved}})$ for all $1 \leq i \leq n$. Then we have that

$$(\mathsf{ri}_w(R, R'))\sigma \in \mathcal{H}(K|_{\mathsf{solved}}).$$

Proof. Identical to Lemma 5.13.

□

Lemma 5.15. *Let K be a saturated knowledge base, let $f \in K$ be a statement*

$$f = \Big(\mathsf{r}_w \Leftarrow B_1, \ldots, B_n\Big)$$

and let σ be a substitution grounding for f such that $B_i\sigma \in \mathcal{H}(K|_{\mathsf{solved}})$ for all $1 \leq i \leq n$.

Then we have that

$$\mathsf{r}_w\sigma \in \mathcal{H}(K|_{\mathsf{solved}}).$$

Proof. Identical to Lemma 5.13 and Lemma 5.14.

□

We now show completeness of the intruder reachable identity predicate.

Lemma 5.16. *Let K be a saturated knowledge base. If $\mathsf{r}_u \in \mathcal{H}(K|_{\mathsf{solved}})$ and $\mathsf{i}_u(R, R') \in \mathcal{H}(K|_{\mathsf{solved}})$, then $\mathsf{ri}_u(R, R') \in \mathcal{H}(K|_{\mathsf{solved}})$.*

Proof. As $\mathsf{r}_u \in \mathcal{H}(K|_{\mathsf{solved}})$, there exists a solved statement $f = \Big(\mathsf{r}_v \Leftarrow B_1, \ldots, B_n\Big) \in K|_{\mathsf{solved}}$ and a substitution σ grounding for f such that $B_i\sigma \in \mathcal{H}(K|_{\mathsf{solved}})$ for all $1 \leq i \leq n$ and such that $u = v\sigma$.

As $\mathsf{i}_u(R, R') \in \mathcal{H}(K|_{\mathsf{solved}})$, there exists by Proposition 5.4 a solved statement $g = \Big(\mathsf{i}_w(T, T') \Leftarrow B_{n+1}, \ldots, B_m\Big)$ and a substitution τ grounding for g such that $B_i\tau \in \mathcal{H}(K|_{\mathsf{solved}})$ for all $n+1 \leq i \leq m$ and such that $u \sqsupseteq w\tau$, $R = T\tau$ and $R' = T'\tau$.

As $v\sigma = u \sqsupseteq w\tau$, it follows that $v = v_0v_1$ such that v_0 and w are unifiable ($\sigma \cup \tau$ is such a unifier). Let $\omega = \mathsf{mgu}(v_0, w)$ and let π be such that $\sigma \cup \tau = \omega \circ \pi$.

As the knowledge base is saturated, the TEST saturation rule must have fired for f and g and therefore K must have been updated by h where

$$h = \Big((\mathsf{ri}_v(T, T') \Leftarrow B_1, \ldots, B_m)\omega\Big).$$

But as h is not a deduction fact, the update must have simply added h to K and therefore $h \in K$.

We have that $B_i\omega\pi = B_i\sigma \in \mathcal{H}(K|_{\mathsf{solved}})$ for all $1 \leq i \leq n$ and that $B_i\omega\pi = B_i\tau \in \mathcal{H}(K|_{\mathsf{solved}})$ for all $n+1 \leq i \leq m$. By applying Lemma 5.14 to the statement h and the substitution π, we obtain that $\mathsf{ri}_v(T, T')\omega\pi = \mathsf{ri}_u(R, R') \in \mathcal{H}(K|_{\mathsf{solved}})$.

□

We next show completeness of the intruder identity predicate.

Lemma 5.17. *Let K be a saturated knowledge base such that $\mathsf{k}_u(R,t) \in \mathcal{H}(K|_{\mathsf{solved}})$ and $\mathsf{k}_{uv}(R',t) \in \mathcal{H}(K|_{\mathsf{solved}})$. Then we have that $\mathsf{i}_{uv}(R,R') \in \mathcal{H}(K|_{\mathsf{solved}})$.*

Proof. Let $u = \ell_1, \ldots, \ell_k$ and $v = \ell_{k+1}, \ldots, \ell_l$. As $\mathsf{k}_u(R,t) \in \mathcal{H}(K|_{\mathsf{solved}})$, it follows by Proposition 5.4 that there exist

$$f = \Big(\mathsf{k}_w(S,s) \Leftarrow B_1, \ldots, B_n\Big) \in K|_{\mathsf{solved}}$$

and a substitution σ grounding for f such that $B_i\sigma \in \mathcal{H}(K|_{\mathsf{solved}})$ $(1 \leq i \leq n)$ and $\mathsf{k}_w(S,s)\sigma = \mathsf{k}_{u'}(R,t)$ for some $u' \sqsubseteq u$ a prefix of u.

Similarly, as $\mathsf{k}_{uv}(R',t) \in \mathcal{H}(K|_{\mathsf{solved}})$, it follows that there exist

$$f' = \Big(\mathsf{k}_{w'}(S',s') \Leftarrow B'_1, \ldots, B'_m\Big) \in K|_{\mathsf{solved}}$$

and a substitution σ' grounding for f' such that $B'_i\sigma' \in \mathcal{H}(K|_{\mathsf{solved}})$ $(1 \leq i \leq m)$ and $\mathsf{k}_{w'}(S',s')\sigma' = \mathsf{k}_{u''}(R',t)$ for $u'' \sqsubseteq uv$ a prefix of uv.

We have that $w\sigma \sqsubseteq u$, which trivially implies $w\sigma \sqsubseteq uv$. We also have $w'\sigma' \sqsubseteq uv$. Let $w = \ell'_1, \ldots, \ell'_p$ and $w' = \ell''_1, \ldots, \ell''_q$ and let $r = \min\{p,q\}$. We have that $(\ell'_1, \ldots, \ell'_r)\sigma = (\ell''_1, \ldots, \ell''_r)\sigma'$.

We have that $\sigma \cup \sigma'$ is a unifier of $\mathsf{k}_{\ell'_1,\ldots,\ell'_r}(\_,s)$ and $\mathsf{k}_{\ell''_1,\ldots,\ell''_r}(\_,s')$, it follows that the substitution $\tau = \mathsf{mgu}(\mathsf{k}_{\ell'_1,\ldots,\ell'_r}(\_,s), \mathsf{k}_{\ell''_1,\ldots,\ell''_r}(\_,s'))$ exists. In the following, let $\bar{\ell}'''$ denote the sequence $\ell'_1, \ldots, \ell'_p$ if $p \geq q$ or the sequence $\ell''_1, \ldots, \ell''_q$ if $p < q$. As K is saturated, it follows that the equational fact

$$h = \Big(\mathsf{i}_{\bar{\ell}'''}(S,S') \Leftarrow B_1, \ldots, B_n, B'_1, \ldots, B'_m\Big)\tau \in K$$

resulting from applying the EQUATION saturation rule to f and f' is in K.

As $\sigma \cup \sigma'$ is a unifier of $\mathsf{k}_{\ell'_1,\ldots,\ell'_r}(\_,s)$ and $\mathsf{k}_{\ell''_1,\ldots,\ell''_r}(\_,s')$ and as the substitution τ is the most general unifier $\tau = \mathsf{mgu}(\mathsf{k}_{\ell'_1,\ldots,\ell'_r}(\_,s), \mathsf{k}_{\ell''_1,\ldots,\ell''_r}(\_,s'))$, it follows that there exists ω such that $\sigma \cup \sigma' = \tau\omega$.

We have that ω is grounding for h and that $B_1\tau\omega, \ldots, B_n\tau\omega, B'_1\tau\omega, \ldots, B'_m\tau\omega \in \mathcal{H}(K|_{\mathsf{solved}})$. Therefore, we have by Lemma 5.13 that

$$\mathsf{i}_{\bar{\ell}'''}(S,S')\tau\omega = \mathsf{i}_{\bar{\ell}'''(\sigma\cup\sigma')}(R,R') \in \mathcal{H}(K|_{\mathsf{solved}}).$$

But $\bar{\ell}'''(\sigma \cup \sigma')$ is a prefix of uv and therefore

$$\mathsf{i}_{uv}(R,R') \in \mathcal{H}(K|_{\mathsf{solved}})$$

by the EXTENDI rule, which is what we wanted to prove.

□

The next three lemmas show completeness with respect to the intruder knowledge predicate.

Lemma 5.18. *Let K be a saturated knowledge base, let*

$$f = \Big(\mathsf{k}_w(R,t) \Leftarrow B_1, \ldots, B_n\Big)$$

be a statement such that $f\Downarrow \in K|_{\mathsf{solved}}$ and let σ be a substitution grounding for f such that $B_i\sigma \in \mathcal{H}(K|_{\mathsf{solved}})$ for all $1 \leq i \leq n$. Then we have that

$$(\mathsf{k}_w(R,t))\sigma \in \mathcal{H}_{\mathsf{e}}(K|_{\mathsf{solved}}).$$

Proof. We prove this by induction on the number of canonicalization steps.

If f is already in canonical form, then the conclusion is immediately true by definition of $\mathcal{H}$.

Otherwise, there must be a canonicalization rule which can be applied to f. We distinguish between two cases:

1. If the RENAME canonicalization rule can be applied, then f must be of the form:

$$f = \Big(\mathsf{k}_w(R,t) \Leftarrow \mathsf{k}_u(X,x), \mathsf{k}_{uv}(Y,x), B_3, \ldots, B_n\Big).$$

Let us consider the statement f' obtained by applying RENAME to f:

$$f' = \Big(\mathsf{k}_w(R,t) \Leftarrow \mathsf{k}_u(X,x), B_3, \ldots B_n\Big)\{Y \mapsto X\}.$$

By the definition of a statement, Y has at most one occurence in $B_1, \ldots, B_n$ and therefore we have that $(B_1, B_3, \ldots B_n)\{Y \mapsto X\} = (B_1, B_3, \ldots, B_n)$. Therefore $(B_1, B_3, \ldots B_n)\{Y \mapsto X\}\sigma = (B_1, B_3, \ldots, B_n)\sigma$.

We can therefore apply the induction hypothesis on f' and σ to obtain that

$$\mathsf{k}_w(R,t)\{Y \mapsto X\}\sigma \in \mathcal{H}_{\mathsf{e}}(K|_{\mathsf{solved}}). \tag{5.5}$$

But $\mathsf{k}_u(X,x)\sigma \in \mathcal{H}(K|_{\mathsf{solved}})$ and $\mathsf{k}_{uv}(Y,x)\sigma \in \mathcal{H}(K|_{\mathsf{solved}})$. By Lemma 5.17, we have that

$$\mathsf{i}_{uv}(X,Y)\sigma \in \mathcal{H}(K|_{\mathsf{solved}}). \tag{5.6}$$

From Equation 5.5 and Equation 5.6 and as uv is a prefix of w by the definition of a statement, we conclude by Proposition 5.3 that

$$\mathsf{k}_w(R,t)\sigma \in \mathcal{H}_{\mathsf{e}}(K|_{\mathsf{solved}}).$$

2. If the REMOVE canonicalization rule can be applied, then f must be of the form:

$$f = \Big(\mathsf{k}_w(R,t) \Leftarrow \mathsf{k}_u(X,x), B_2, \ldots, B_n\Big).$$

Let f' be the statement obtained from f by applying REMOVE. We have that

$$f' = \Big(\mathsf{k}_w(R,t) \Leftarrow B_2, \ldots, B_n\Big).$$

By applying the induction hypothesis on f' and σ, we immediately obtain our conclusion:

$$\mathsf{k}_w(R,t)\sigma \in \mathcal{H}_{\mathsf{e}}(K|_{\mathsf{solved}}).$$

□

Lemma 5.19. *Let K be a saturated knowledge base, let*

$$f = \Big(\mathsf{k}_w(R,t) \Leftarrow B_1, \ldots, B_n\Big)$$

be a statement such that $f\Downarrow = \Big(\mathsf{k}_w(R',t) \Leftarrow C_1, \ldots, C_m\Big)$ for some $R', C_1, \ldots, C_m$ and let R'' be a recipe such that

$$g = \big(\mathsf{k}_w(R'',t) \Leftarrow C_1, \ldots, C_m\big) \in \mathbf{deriv}(K|_{\mathsf{solved}})$$

and such that

$$h = \Big(\mathsf{i}_w(R'',R') \Leftarrow C_1, \ldots, C_m\Big) \in K|_{\mathsf{solved}}.$$

Let σ be a substitution grounding for f such that $B_i\sigma \in \mathcal{H}(K|_{\mathsf{solved}})$ for all $1 \leq i \leq n$. Then we have that

$$(\mathsf{k}_w(R,t))\sigma \in \mathcal{H}_{\mathsf{e}}(K|_{\mathsf{solved}}).$$

Proof. We prove the lemma by induction on the number of steps to reach the canonical form.

If f is already in canonical form we have that $R = R'$, $B_1, \ldots, B_n = C_1, \ldots, C_m$ and, by applying Proposition 5.2 to g and σ, we have that

$$\mathsf{k}_w(R'', t)\sigma \in \mathcal{H}(K|_{\mathsf{solved}}).$$

Furthermore, as $h \in K|_{\mathsf{solved}}$ and as the body of $h\sigma$ are simple consequence of $K|_{\mathsf{solved}}$, we have that

$$\mathsf{i}_w(R'', R')\sigma \in \mathcal{H}(K|_{\mathsf{solved}}).$$

It immediately follows that

$$\mathsf{k}_w(R', t)\sigma \in \mathcal{H}_{\mathsf{e}}(K|_{\mathsf{solved}}),$$

which is what we had the prove since $R = R'$.

Otherwise, there must be a canonicalization rule which can be applied to f. We distinguish between two cases:

1. If the RENAME canonicalization rule can be applied, then f must be of the form:

$$f = \Big(\mathsf{k}_w(R, t) \Leftarrow \mathsf{k}_u(X, x), \mathsf{k}_{uv}(Y, x), B_3, \ldots, B_n\Big).$$

Let us consider the statement f' obtained by applying RENAME to f:

$$f' = \Big(\mathsf{k}_w(R, t) \Leftarrow \mathsf{k}_u(X, x), B_3, \ldots B_n\Big)\{Y \mapsto X\}.$$

By the definition of a statement, Y has at most one occurence in $B_1, \ldots, B_n$ and therefore we have that $(B_1, B_3, \ldots B_n)\{Y \mapsto X\} = (B_1, B_3, \ldots, B_n)$. Therefore $(B_1, B_3, \ldots B_n)\{Y \mapsto X\}\sigma = (B_1, B_3, \ldots, B_n)\sigma$.

We can therefore apply the induction hypothesis on f' and σ to obtain that

$$\mathsf{k}_w(R, t)\{Y \mapsto X\}\sigma \in \mathcal{H}_{\mathsf{e}}(K|_{\mathsf{solved}}). \tag{5.7}$$

But $\mathsf{k}_u(X, x)\sigma \in \mathcal{H}(K|_{\mathsf{solved}})$ and $\mathsf{k}_{uv}(Y, x)\sigma \in \mathcal{H}(K|_{\mathsf{solved}})$. By Lemma 5.17, we have that

$$\mathsf{i}_{uv}(X, Y)\sigma \in \mathcal{H}(K|_{\mathsf{solved}}). \tag{5.8}$$

From Equation 5.7 and Equation 5.8 and as uv is a prefix of w by the definition of a statement, we conclude by Proposition 5.3 that

$$\mathsf{k}_w(R, t)\sigma \in \mathcal{H}_{\mathsf{e}}(K|_{\mathsf{solved}}).$$

2. If the REMOVE canonicalization rule can be applied, then f must be of the form:

$$f = \Big(\mathsf{k}_w(R, t) \Leftarrow \mathsf{k}_u(X, x), B_2, \ldots, B_n\Big).$$

Let f' be the statement obtained from f by applying REMOVE. We have that

$$f' = \Big(\mathsf{k}_w(R, t) \Leftarrow B_2, \ldots, B_n\Big).$$

By applying the induction hypothesis on f' and σ, we immediately obtain our conclusion:

$$\mathsf{k}_w(R, t)\sigma \in \mathcal{H}_{\mathsf{e}}(K|_{\mathsf{solved}}).$$

□

This lemma is the main lemma that shows completeness of the intruder knowledge predicate.

Lemma 5.20. *Let K be a saturated knowledge base, let $f \in K$ be a statement*

$$f = \Big(\mathsf{k}_w(R,t) \Leftarrow B_1, \ldots, B_n\Big)$$

and let σ be a substitution grounding for f such that $B_i\sigma \in \mathcal{H}(K|_{\mathsf{solved}})$ for all $1 \leq i \leq n$. Then we have that

$$(\mathsf{k}_w(R,t))\sigma \in \mathcal{H}_{\mathsf{e}}(K|_{\mathsf{solved}}).$$

Proof. Let $\mathcal{G} = \sum_{i\in\{1,\ldots,n\}} \mathcal{S}(B_i\sigma, K|_{\mathsf{solved}})$. We prove the lemma by induction on $\mathcal{G}$.

If f is a solved statement, the conclusion is trivial by the definitions of $\mathcal{H}$, $\mathcal{H}_{\mathsf{e}}$.

Otherwise, there exists some $B_j = \mathsf{k}_{w_j}(X_j, t_j)$ (with $1 \leq j \leq n$) such that $t_j \notin \mathcal{X}$.

As $B_j\sigma \in \mathcal{H}(K|_{\mathsf{solved}})$, we have by Proposition 5.4 that there exist

$$g = \Big(\mathsf{k}_{u'}(R',t') \Leftarrow B'_1, \ldots, B'_m\Big) \in K|_{\mathsf{solved}},$$

a substitution σ' grounding for g such that $B'_1\sigma', \ldots, B'_m\sigma' \in \mathcal{H}(K|_{\mathsf{solved}})$, $\mathsf{k}_{u'}(R',t')\sigma' = \mathsf{k}_u(X_j,t_j)\sigma$ for some prefix $u \sqsubseteq w_j$ of w_j and $\mathcal{S}(B_j\sigma, K|_{\mathsf{solved}}) > \sum_{i\in\{1,\ldots,m\}} \mathcal{S}(B'_i\sigma', K|_{\mathsf{solved}})$.

As $\sigma \cup \sigma'$ is a unifier of $\mathsf{k}_u(X_j,t_j)$ and $\mathsf{k}_{u'}(R',t')$, it follows that $\tau = \mathsf{mgu}(\mathsf{k}_u(X_j,t_j), \mathsf{k}_{u'}(R',t'))$ exists. Let $\sigma \cup \sigma'$ must be an instance of the most general unifier, let ω be a substitution such that $\sigma \cup \sigma' = \tau\omega$.

As K is saturated, it follows that the RESOLUTION saturation rule was applied to f and g. Let h be the resulting statement:

$$h = \Big(\mathsf{k}_w(R,t) \Leftarrow B_1, \ldots, B_{j-1}, B_{j+1}, \ldots, B_n, B'_1, \ldots, B'_m\Big)\tau.$$

We distinguish two cases:

1. if h is not solved we have that $h \in K$ by the update function (as K is saturated).

   We can therefore apply the induction hypothesis on h and on the substitution ω to immediately conclude.

2. if h is solved, we distinguish two cases:

   (a) either $h\Downarrow \in K$, in which case we conclude by applying Lemma 5.18 to h and ω.

   (b) or $h\Downarrow = \Big(\mathsf{k}_w(R'',t) \Leftarrow C_1, \ldots, C_k\Big)$ and

   $$h' = \Big(\mathsf{k}_w(R''',t) \Leftarrow C_1, \ldots, C_k\Big) \in \mathbf{deriv}(K|_{\mathsf{solved}})$$

   and

   $$h'' = \Big(\mathsf{i}_w(R''',R'') \Leftarrow C_1, \ldots, C_k\Big) \in K|_{\mathsf{solved}}$$

   for some R''', in which case we conclude by applying Lemma 5.19.

□

The following proposition follows from the definition of $\mathcal{H}_{\mathsf{e}}$.

Proposition 5.5. *If $\mathsf{k}_u(R,t) \in \mathcal{H}_{\mathsf{e}}(K)$, then $\mathsf{k}_{uv}(R,t) \in \mathcal{H}_{\mathsf{e}}(K)$.*

Proof. As $\mathsf{k}_u(R,t) \in \mathcal{H}_{\mathsf{e}}(K)$, it follows that $\mathsf{k}_u(R',t) \in \mathcal{H}(K)$ and $\mathsf{i}_u(R',R) \in \mathcal{H}_{\mathsf{e}}(K)$ for some R'. By the EXTENDK rule, we have that $\mathsf{k}_{uv}(R',t) \in \mathcal{H}(K)$ and by the EXTEND rule, we have that $\mathsf{i}_{uv}(R',R) \in \mathcal{H}_{\mathsf{e}}(K)$. We conclude by rule EQUATIONAL CONSEQUENCE that $\mathsf{k}_{uv}(R,t) \in \mathcal{H}_{\mathsf{e}}(K)$, which is what we had to show.

□

The following lemma shows completeness of the saturation procedure.

Lemma 5.21. *Let K be a set of statements and let K' be the saturation of the knowledge base obtained by updating the empty knowledge base by each statement in K.*
Then $\mathcal{H}(K) \subseteq \mathcal{H}_{\mathsf{e}}(K'|_{\mathsf{solved}})$

Proof. Let $H \in \mathcal{H}(K)$. We will prove by induction on the proof tree of $H \in \mathcal{H}(K)$ that each node of the tree is in $\mathcal{H}_{\mathsf{e}}(K'|_{\mathsf{solved}})$. We distinguish two cases:

1. if $H = \mathsf{k}_w(R,t)$ and $\mathsf{k}_u(R,t) \in \mathcal{H}(K)$ for some prefix u of w, in which case by the induction hypothesis we have that $\mathsf{k}_u(R,t) \in \mathcal{H}_{\mathsf{e}}(K'|_{\mathsf{solved}})$ and we conclude by Proposition 5.5.

2. if $H = \mathsf{i}_w(R,R')$ and $\mathsf{i}_u(R,R') \in \mathcal{H}(K)$ for some prefix u of w, we have that $\mathsf{i}_u(R,R') \in \mathcal{H}_{\mathsf{e}}(K'|_{\mathsf{solved}})$ by the induction hypothesis and therefore $\mathsf{i}_w(R,R') \in \mathcal{H}_{\mathsf{e}}(K'|_{\mathsf{solved}})$ by rule Extend.

3. otherwise there is a statement

$$f = \left(H' \Leftarrow B'_1, \ldots, B'_n\right) \in K$$

and a substitution σ grounding for f such that $H = H'\sigma$ and $B'_i\sigma \in \mathcal{H}(K)$.

By the induction hypothesis, we have that $B'_i\sigma \in \mathcal{H}_{\mathsf{e}}(K'|_{\mathsf{solved}})$. W.l.o.g. assume that $B'_i = \mathsf{k}_{w'_i}(X_i, t'_i)$. As $B'_i\sigma \in \mathcal{H}_{\mathsf{e}}(K'|_{\mathsf{solved}})$, we have by definition of $\mathcal{H}_{\mathsf{e}}$ that there exist R'_i such that

$$\mathsf{k}_{w'_i\sigma}(R'_i, t_i\sigma) \in \mathcal{H}(K'|_{\mathsf{solved}}), \tag{5.9}$$

$$\mathsf{i}_{w'_i\sigma}(R'_i, X_i\sigma) \in \mathcal{H}_{\mathsf{e}}(K'|_{\mathsf{solved}}) \tag{5.10}$$

for all $1 \leq i \leq n$.

But $w'_i\sigma$ is a prefix of w, where w is such that $H = \mathit{predicate}_w(\ldots)$ with $\mathit{predicate} \in \{\mathsf{r}, \mathsf{i}, \mathsf{ri}, \mathsf{k}\}$. Therefore, by applying the Extend rule to Equation (5.10), we obtain

$$\mathsf{i}_w(R'_i, X_i\sigma) \in \mathcal{H}_{\mathsf{e}}(K|_{\mathsf{solved}}). \tag{5.11}$$

Let σ' be the substitution defined to be σ except that it maps X_i to R'_i for all $1 \leq i \leq n$.

We will show that $H'\sigma' \in \mathcal{H}_{\mathsf{e}}(K'|_{\mathsf{solved}})$. As K' was updated by f, there are three cases:

(a) if $f \in K'$, we conclude by Lemma 5.20, Lemma 5.13, Lemma 5.14 or Lemma 5.15.

(b) or $f\Downarrow \in K'$ and $f \notin K'$, in which case f must be a solved deduction statement. In this case, by Lemma 5.18, we obtain that $H'\sigma' \in \mathcal{H}_{\mathsf{e}}(K'|_{\mathsf{solved}})$.

(c) or $f\Downarrow = \left(\mathsf{k}_w(R,t) \Leftarrow C_1, \ldots, C_m\right)$ and there exists R' such that

$$\left(\mathsf{k}_w(R',t) \Leftarrow C_1, \ldots, C_m\right) \in \mathbf{deriv}(K'|_{\mathsf{solved}})$$

and such that

$$\left(\mathsf{i}_w(R,R') \Leftarrow C_1, \ldots, C_m\right) \in K'|_{\mathsf{solved}}.$$

In this case, we have that $H'\sigma' \in \mathcal{H}_{\mathsf{e}}(K'|_{\mathsf{solved}})$ by Lemma 5.19.

We have shown that $H'\sigma' \in \mathcal{H}_{\mathsf{e}}(K'|_{\mathsf{solved}})$. But $H'\sigma' = H'\{X_i \mapsto R'_i\}\sigma$ and therefore $H'\{X_i \mapsto R_i\}\sigma \in \mathcal{H}_{\mathsf{e}}(K'|_{\mathsf{solved}})$. By Equation (5.11), we obtain that $H'\{X_i \mapsto X_i\sigma\}\sigma = H'\sigma \in \mathcal{H}_{\mathsf{e}}(K'|_{\mathsf{solved}})$, which is what we had to show.

□

We can now combine the completeness of the initial set of seed statements with the complete of the saturation procedure to prove Theorem 5.3.

Theorem 5.3 (Completeness of saturation)**.** *Let T and S be traces and let φ be a frame such that*

$$(T, \emptyset) \xrightarrow{L_1,\ldots,L_n} (S, \varphi).$$

Let K be a saturated knowledge base associated to T, i.e., $K = \mathsf{sat}(K_i(T))$. Then we have that

1. $\mathsf{r}_{L_1\varphi\downarrow,\ldots,L_n\varphi\downarrow} \in \mathcal{H}_{\mathsf{e}}(K|_{\mathsf{solved}})$,
2. *if $\varphi \vdash^R t$ then $\mathsf{k}_{L_1\varphi\downarrow,\ldots,L_n\varphi\downarrow}(R, t\downarrow) \in \mathcal{H}_{\mathsf{e}}(K|_{\mathsf{solved}})$ and*
3. *if $\varphi \vdash^R t$ and $\varphi \vdash^{R'} t$, then $\mathsf{i}_{L_1\varphi\downarrow,\ldots,L_n\varphi\downarrow}(R, R') \in \mathcal{H}_{\mathsf{e}}(K|_{\mathsf{solved}})$.*

Proof. The first two items are immediate consequences of Lemma 5.2 and of Lemma 5.21.

The third item follows immediately from the second item and Lemma 5.17.

□

5.8 Correctness of the Trace Inclusion Algorithm

This section is dedicated to proving Theorem 5.4. In order to prove it, we need a few technical lemmas.

First we show that each intruder reachability statement and each deduction statement in the saturated knowledge base has a certain form.

Lemma 5.22. *Let T be a trace and let K be a saturated knowledge base associated to T. Then for any statement $f \in K$, we have that:*

1. *if $f = \left(\mathsf{r}_{l_1,\ldots,l_n} \Leftarrow \{\mathsf{k}_{w_i}(X_i, t_i)\}_{i\in\{1,\ldots,m\}}\right)$ and $x \in \mathcal{V}ar(l_k)$ then there exists $w_j = l_1, \ldots, l_{k'}$ with $k' < k$ such that $x \in \mathcal{V}ar(t_j)$.*
2. *if $f = \left(\mathsf{k}_{l_1,\ldots,l_n}(R, t) \Leftarrow \{\mathsf{k}_{w_i}(X_i, t_i)\}_{i\in\{1,\ldots,m\}}\right)$ and $x \in \mathcal{V}ar(t)$ then $x \in \mathcal{V}ar(t_1, \ldots, t_m)$.*

Proof. The seed knowledge base satisfies the above properties and they are preserved by canonicalization, update and saturation.

□

The next lemma states that by extending a frame, we keep all existing deductions.

Lemma 5.23. *If $\mathcal{D}om(\varphi) \subseteq \mathcal{D}om(\varphi')$ and $\varphi \vdash^r t$, then $\varphi' \vdash^r t$.*

Proof. The same rewrite steps to obtain t from $r\varphi$ are used to obtain t from $r\varphi'$.

□

We now show that operational semantics is stable by replacement of names with recipes.

Lemma 5.24. *Let T_0 be a trace, $\varphi_0 = \emptyset$ the empty frame, and $\{c_1, \ldots, c_k\}$ public names such that $c_i \notin \mathcal{N}ames(T_0)$ for all $1 \le i \le k$. If*

$$(T_0, \varphi_0) \stackrel{L_1}{\Longrightarrow} (T_1, \varphi_1) \stackrel{L_2}{\Longrightarrow} \ldots \stackrel{L_n}{\Longrightarrow} (T_n, \varphi_n)$$

and $\forall 1 \le i \le k$

- *either $c_i \notin \mathcal{N}ames(L_1, \ldots, L_n)$*
- *or $\varphi_{idx(c_i)-1} \vdash^{R_i} t_i$ for some t_i where $idx(c_i) = \min\{j \mid c_i \in \mathcal{N}ames(L_j)\}$*

then

$$(T_0, \varphi_0) \xRightarrow{L_1\pi'} (T_1\pi, \varphi_1\pi) \xRightarrow{L_2\pi'} \dots \xRightarrow{L_n\pi'} (T_n\pi, \varphi_n\pi),$$

where $\pi' = \{c_i \mapsto R_i\}_{i\in\{1,\dots,k\}, c_i \in \mathcal{N}ames(L_1,\dots,L_n)}$ *and* $\pi = \{c_i \mapsto t_i\}_{i\in\{1,\dots,k\}, c_i\in\mathcal{N}ames(L_1,\dots,L_n)}$.

Proof. By induction on n, the same operational steps will take place with the new labels. □

The next lemma shows that deduction is stable by replacement of names with recipes in the recipe and the corresponding deducible terms in the term.

Lemma 5.25. *Let T be a trace, let $\{c_1, \dots, c_k\}$ be public names not appearing in T and let the functions $\pi : \{c_1, \dots, c_k\} \to T(\mathcal{F}, \mathcal{N}, \mathcal{M})$ and $\pi' : \{c_1, \dots, c_k\} \to T(\mathcal{F}, \mathcal{M}, \mathcal{W})$ be mappings from names to terms. If $T \models \mathsf{k}_{l_1,\dots,l_i}(R_{i+1}, t_{i+1})$ and $T \models \mathsf{k}_{l_1\pi,\dots,l_i\pi}(c_i\pi', c_i\pi)$ then $T \models \mathsf{k}_{l_1\pi,\dots,l_i\pi}(R_{i+1}\pi', t_{i+1}\pi)$.*

Proof. By induction on R. □

We now show that operational semantics is stable by replacement of recipes with equivalent recipes.

Lemma 5.26. *Let T be a trace and φ a frame such that $(T, \varphi) \xRightarrow{L} (T', \varphi')$ and such that*

1. *either $M = L$,*
2. *or $M = \mathbf{receive}(d, R')$ and $L = \mathbf{receive}(d, R)$ such that $(R = R')\varphi$.*

Then we have that $(T, \varphi) \xRightarrow{M} (T', \varphi')$.

Proof. If $M = L$ then the result is obvious. Otherwise, R and R' are recipes for the same term in φ and therefore the transition still holds. □

We are now ready to prove the correctness of the trace-inclusion checking algorithm:

Theorem 5.4. *Let T be a trace and let P be a determinate process. Let K be the set of solved statements from a saturated knowledge base associated to T. Then $T \sqsubseteq_c P$ iff the following tests hold:*

$$\textsc{Reachability}\ \frac{\begin{array}{c}\left(\mathsf{r}_{l_1,\dots,l_n} \Leftarrow \{\mathsf{k}_{w_i}(X_i, x_i)\}_{i\in\{1,\dots,m\}}\right) \in K \\ c_1, \dots, c_k \text{ fresh constants} \qquad \sigma : \mathcal{V}ar(l_1, \dots, l_n) \to \{c_1, \dots, c_k\} \text{ is a bijection} \\ \mathsf{k}_{l_1\sigma,\dots,l_{i-1}\sigma}(R_i, t_i\sigma) \in \mathcal{H}(K) \text{ for all } i \text{ such that } l_i = \mathbf{receive}(d_i, t_i) \\ M_i = l_i \text{ if } l_i \in \{\mathbf{test}, \mathbf{send}(\_)\} \\ M_i = \mathbf{receive}(d_i, R_i) \text{ if } l_i = \mathbf{receive}(d_i, t_i)\end{array}}{(P, \emptyset) \xRightarrow{M_1,\dots,M_n} (T', \varphi)}$$

$$\textsc{Identity}\ \frac{\begin{array}{c}\left(\mathsf{ri}_{l_1,\dots,l_n}(R, R') \Leftarrow \{\mathsf{k}_{w_i}(X_i, x_i)\}_{i\in\{1,\dots,m\}}\right) \in K \\ c_1, \dots, c_k \text{ fresh constants} \qquad \sigma : \mathcal{V}ar(l_1, \dots, l_n) \to \{c_1, \dots, c_k\} \text{ is a bijection} \\ \mathsf{k}_{l_1\sigma,\dots,l_{i-1}\sigma}(R_i, t_i\sigma) \in \mathcal{H}(K) \text{ for all } i \text{ such that } l_i = \mathbf{receive}(t_i) \\ M_i = l_i \text{ if } l_i \in \{\mathbf{test}, \mathbf{send}(\_)\} \qquad M_i = \mathbf{receive}(d_i, R_i) \text{ if } l_i = \mathbf{receive}(d_i, t_i)\end{array}}{(P, \emptyset) \xRightarrow{M_1,\dots,M_n} (T', \varphi) \text{ such that } (R\omega = R'\omega)\varphi \text{ where } \omega = \{X_i \mapsto x_i\sigma\}}$$

Proof. We first prove that if any of the tests fail, $T \not\sqsubseteq P$. Indeed, if the Reachability test fails, we have that $(P, \emptyset) \not\xRightarrow{M_1,\dots,M_n} (\_, \_)$, but, by the soundness of K, we have that $(T, \emptyset) \xrightarrow{M_1,\dots,M_n} (\_, \_)$ and therefore $T \not\sqsubseteq P$.

If the Identity test fails, we have that:

1. either $(P,\emptyset) \stackrel{M_1,\ldots,M_n}{\not\Longrightarrow} (\_,\_)$, in which case, by the soundness of K, we have that $(T,\emptyset) \stackrel{M_1,\ldots,M_n}{\longrightarrow} (\_,\_)$ and therefore $T \not\sqsubseteq P$.

2. or for any φ' such that $(P,\emptyset) \stackrel{M_1,\ldots,M_n}{\Longrightarrow} (\_,\varphi')$ we have $(R\pi \neq R'\pi)\varphi'$. By the soundness of K, we have however that $(T,\emptyset) \stackrel{M_1,\ldots,M_n}{\longrightarrow} (\_,\varphi)$ and $(R\pi = R'\pi)\varphi$. Therefore $T \not\sqsubseteq P$.

Next, we prove that if $T \not\sqsubseteq P$, then at least one test fails. We assume by contradiction that $T \not\sqsubseteq P$, that all tests pass and we derive a contradiction.

As $T \not\sqsubseteq P$, it follows that there exist $L_1,\ldots,L_n,\varphi$ such that either:

$$\begin{array}{ll}(T,\emptyset) \stackrel{L_1,\ldots,L_n}{\longrightarrow} (T',\varphi) & \text{and}\\ \forall S \in P : (S,\emptyset) \stackrel{L_1,\ldots,L_n}{\not\Longrightarrow} (S',\psi) & \end{array}$$

or:

$$\begin{array}{ll}(T,\emptyset) \stackrel{L_1,\ldots,L_n}{\longrightarrow} (T',\varphi) \text{ and } (R = R')\varphi & \text{and}\\ \forall S \in P, (S,\emptyset) \stackrel{L_1,\ldots,L_n}{\Longrightarrow} (S',\psi) \text{ implies } (R \neq R')\psi & \end{array}$$

Let n be the smallest index such that one of the above holds. We then have that:

$$(T,\emptyset) \stackrel{L_1}{\longrightarrow} (T_1,\varphi_1) \stackrel{L_2}{\longrightarrow} \ldots \stackrel{L_{n-1}}{\longrightarrow} (T_{n-1},\varphi_{n-1}) \stackrel{L_n}{\longrightarrow} (T_n,\varphi_n)$$

and there exists $S \in P$ such that:

$$(S,\emptyset) \stackrel{L_1}{\Longrightarrow} (S_1,\psi_1) \stackrel{L_2}{\Longrightarrow} \ldots \stackrel{L_{n-1}}{\Longrightarrow} (S_{n-1},\psi_{n-1}),$$

such that for all R,R' we have $(R = R')\varphi_i \implies (R = R')\psi_i$ $(1 \leq i \leq n-1)$ and:

1. either for all $U \in P$ we have $(U,\emptyset) \stackrel{L_1,\ldots,L_n}{\not\Longrightarrow} (\_,\psi)$

2. or there exist recipes R,R' such that for any $U' \in P$ such that $(U',\emptyset) \stackrel{L_1,\ldots,L_n}{\Longrightarrow} (\_,\psi')$ we have $(R \neq R')\psi'$.

We consider each of the cases separately:

1. If we are in the case of Item 1, as $(T,\emptyset) \stackrel{L_1,\ldots,L_n}{\longrightarrow} (T_n,\varphi_n)$, we have by Theorem 5.3 that $\mathsf{r}_{L_1\varphi_n\downarrow,\ldots,L_n\varphi_n\downarrow} \in \mathcal{H}_{\mathsf{e}}(K)$. By the definition of $\mathcal{H}_{\mathsf{e}}$, we have that it contains no reachability statements in addition to those in $\mathcal{H}$: therefore $\mathsf{r}_{L_1\varphi_n\downarrow,\ldots,L_n\varphi_n\downarrow} \in \mathcal{H}(K)$.

   Therefore there exist a statement $f = \left(\mathsf{r}_{l_1,\ldots,l_n} \Leftarrow \mathsf{k}_{w_i}(X_i,x_i)_{i\in\{1,\ldots,m\}}\right) \in K$ and a substitution τ grounding for f such that $l_i\tau = L_i\varphi_n\downarrow$ (for all $1 \leq i \leq n$) and such that $\mathsf{k}_{w_i\tau}(X_i\tau,x_i\tau) \in \mathcal{H}(K)$.

   Let $c_1,\ldots,c_k$ be fresh public names and let $\sigma : \mathcal{V}ar(l_1,\ldots,l_n) \to \{c_1,\ldots,c_k\}$ be a bijection. As $\left(\mathsf{k}_{\ell_1,\ldots,\ell_{|w_i|}}(c_j,c_j) \Leftarrow\right) \in K(T)$ for all $1 \leq i \leq m$ and all $1 \leq j \leq k$ and $w_i\sigma$ is an instance of $\ell_1,\ldots,\ell_{|w_i|}$, it follows by Theorem 5.3 that there exists a substitution σ' such that $\mathsf{k}_{w_i\sigma}(X_i\sigma',x_i\sigma) \in \mathcal{H}_{\mathsf{e}}(K)$ for all $1 \leq i \leq m$, which implies by the definition of $\mathcal{H}_{\mathsf{e}}$ that there exists a substitution σ'' such that $\mathsf{k}_{w_i\sigma}(X_i\sigma'',x_i\sigma) \in \mathcal{H}(K)$ for all $1 \leq i \leq m$.

   By instantiating f with $\sigma \cup (\sigma''[\{X_1,\ldots,X_m\}])$, we obtain that $\mathsf{r}_{l_1\sigma,\ldots,l_n\sigma} \in \mathcal{H}(K)$. By the soundness of K and of $\mathcal{H}$, it follows that $T \models \mathsf{r}_{l_1\sigma,\ldots,l_n\sigma}$. Therefore, there exist recipes R'_i (for all $1 \leq i \leq n$ such that $l_i = \mathbf{receive}(\_,t_i)$) such that $T \models \mathsf{k}_{l_1\sigma,\ldots,l_{i-1}\sigma}(R'_i,t_i\sigma)$. By the completeness of K, it follows that there exist recipes R_i such that $\mathsf{k}_{l_1\sigma,\ldots,l_{i-1}\sigma}(R_i,t_i\sigma) \in \mathcal{H}(K)$.

Let $M_i = l_i$ if $l_i \in \{\mathbf{test}, \mathbf{send}(\_)\}$ and let $M_i = \mathbf{receive}(d_i, R_i)$ if $l_i = \mathbf{receive}(d_i, t_i)$ for all $1 \leq i \leq n$. Because the REACHABILITY test worked, it follows that there exists $S'_0 \in P$ such that, if we let $\psi'_0 = \emptyset$, we have

$$(S'_0, \psi'_0) \xRightarrow{M_1} (S'_1, \psi'_1) \xRightarrow{M_2} \ldots \xRightarrow{M_n} (S'_n, \psi'_n).$$

We already have that $\mathsf{k}_{w_j\tau}(X_j\tau, x_j\tau) \in \mathcal{H}(K)$ by choice of f and of τ. By the soundness of $\mathcal{H}$ and K, we obtain $T \models \mathsf{k}_{w_j\tau}(X_j\tau, x_j\tau)$. By Lemma 5.22, we have that $|w_j| < i$, and as w_j is a prefix of $l_1, \ldots, l_{i-1}$, we have that $T \models \mathsf{k}_{l_1\tau,\ldots,l_{i-1}\tau}(X_j\tau, x_j\tau)$.

Let $\pi : \{c_1, \ldots, c_k\} \to T(\mathcal{F}, \mathcal{N}, \mathcal{M})$ and $\pi' : \{c_1, \ldots, c_k\} \to T(\mathcal{F}, \mathcal{M}, \mathcal{W})$ be defined such that $\pi(c_l) = x_j\tau$ and $\pi'(c_l) = X_j\tau$ when $\sigma(x_j) = c_l$. As $X_j\tau = c_l\pi'$, $x_j\tau = c_l\pi$ and $l_1\tau, \ldots, l_{i-1}\tau = l_1\sigma\pi, \ldots, l_{i-1}\sigma\pi$ therefore we have that $T \models \mathsf{k}_{l_1\sigma\pi,\ldots,l_{i-1}\sigma\pi}(c_l\pi', c_l\pi)$.

We also have that $\mathsf{k}_{l_1\sigma,\ldots,l_{i-1}\sigma}(R_i, t_i\sigma) \in \mathcal{H}(K)$. By the soundness of $\mathcal{H}$ and K, we have that $T \models \mathsf{k}_{l_1\sigma,\ldots,l_{i-1}\sigma}(R_i, t_i\sigma)$. We apply Lemma 5.25 to obtain that $T \models \mathsf{k}_{l_1\sigma\pi,\ldots,l_{i-1}\sigma\pi}(R_i\pi', t_i\sigma\pi)$. But $t_i\sigma\pi = t_i\tau$ and $l_1\sigma\pi, \ldots, l_{i-1}\sigma\pi = l_1\tau, \ldots, l_{i-1}\tau$ and therefore we have that $T \models \mathsf{k}_{l_1\tau,\ldots,l_{i-1}\tau}(R_i\pi', t_i\tau)$.

Let R''_i be such that $L_i = \mathbf{receive}(d_i, R''_i)$ for some channel d_i for all i such that the ith action of T is a receive. By the definition of $\models$, we have therefore that $T \models \mathsf{k}_{l_1\tau,\ldots,l_{i-1}\tau}(R''_i, t_i\tau)$.

We conclude that $T \models \mathsf{i}_{l_1\tau,\ldots,l_{i-1}\tau}(R_i\pi', R''_i)$, or, equivalently, $(R_i\pi' = R''_i)\varphi_{i-1}$. By the hypothesis, we have that $(R_i\pi' = R''_i)\psi_{i-1}$ as well.

From Lemma 5.24, we obtain that

$$(S'_0, \psi'_0) = (S'_0\pi, \psi_0\pi) \xRightarrow{M_1\pi'} (S'_1\pi, \psi'_1\pi) \xRightarrow{M_2\pi'} \ldots \xRightarrow{M_n\pi'} (S'_n\pi, \psi'_n\pi).$$

We will show by induction on n that

$$(S'_0\pi, \psi_0\pi) \xRightarrow{L_1} (S'_1\pi, \psi'_1\pi) \xRightarrow{L_2} \ldots \xRightarrow{L_n} (S'_n\pi, \psi'_n\pi).$$

We assume by the induction hypothesis that

$$(S'_0\pi, \psi_0\pi) \xRightarrow{L_1} (S'_1\pi, \psi'_1\pi) \xRightarrow{L_2} \ldots \xRightarrow{L_i} (S'_i\pi, \psi'_i\pi)$$

and we show that

$$(S'_i\pi, \psi'_i\pi) \xRightarrow{L_{i+1}} (S'_{i+1}\pi, \psi'_{i+1}\pi).$$

We will show that L_{i+1} and $M_{i+1}\pi'$ satisfy the conditions of Lemma 5.26 and then we can easily conclude by it. Indeed, either $L_{i+1} = M_{i+1}\pi'$ (in the case of a **test** or **send** action), or $L_{i+1} = \mathbf{receive}(d_{i+1}, R''_{i+1})$ and $M_{i+1}\pi' = \mathbf{receive}(d_{i+1}, R_{i+1}\pi')$ (in the case of a **receive** action). In the second case, as we have shown $(R_{i+1}\pi' = R''_{i+1})\psi_i$, by determinacy of P it follows that $\psi_i\pi \approx_s \psi'_i\pi$ and therefore $(R_{i+1}\pi' = R''_{i+1})\psi'_i$ as well. As the hypothesis of Lemma 5.26 are satisfied, we can conclude.

Let $U = S'_0$. We have shown that $(U, \emptyset) \xRightarrow{L_1,\ldots,L_n} (\_, \_)$, therefore obtaining a contradiction. Therefore Item 1 cannot hold.

2. In the case of Item 2, we assume that for all $U \in P$ such that $(U, \emptyset) \xRightarrow{L_1,\ldots,L_n} (\_, \psi)$ we have $(R \neq R')\psi$ to obtain a contradiction.

   As $(R = R')\varphi_n$, it follows by Theorem 5.3 that $\mathsf{i}_{L_1\varphi_n\downarrow,\ldots,L_n\varphi_n\downarrow}(R, R') \in \mathcal{H}_{\mathsf{e}}(K)$.

   As $(R \neq R')\psi$ it follows that there exist recipes Q, Q' such that $\mathsf{i}_{L_1\varphi_n\downarrow,\ldots,L_n\varphi_n\downarrow}(Q, Q') \in \mathcal{H}(K)$ but $(Q \neq Q')\psi$.

As $\mathsf{r}_{L_1\varphi_n\downarrow,\ldots,L_n\varphi_n\downarrow} \in \mathcal{H}(K)$, we have by Lemma 5.16 that $\mathsf{ri}_{L_1\varphi\downarrow,\ldots,L_n\varphi\downarrow}(Q,Q') \in \mathcal{H}(K)$. Therefore there exists a statement

$$f = \left(\mathsf{ri}_{l_1,\ldots,l_n}(P,P') \Leftarrow \{\mathsf{k}_{w_i}(X_i,x_i)\}_{i\in\{1,\ldots,m\}}\right) \in K$$

and a substitution τ grounding for f such that $\mathsf{k}_{w_i\tau}(X_i\tau,x_i\tau) \in \mathcal{H}(K)$ (for all $1 \leq i \leq m$), $l_1\tau,\ldots,l_n\tau = L_1\varphi_n\downarrow,\ldots,L_n\varphi_n\downarrow$, $P\tau = Q$ and $P'\tau = Q'$.

Let $c_1,\ldots,c_k$ be fresh public names and let $\sigma : \mathcal{V}ar(l_1,\ldots,l_n) \to \{c_1,\ldots,c_k\}$ be a bijection. As $\left(\mathsf{k}_{\ell_1,\ldots,\ell_{|w_i|}}(c_j,c_j) \Leftarrow\right) \in K(T)$ for all $1 \leq i \leq m$ and all $1 \leq j \leq k$ and $w_i\sigma$ is an instance of $\ell_1,\ldots,\ell_{|w_i|}$, it follows by Theorem 5.3 that there exists a substitution σ' such that $\mathsf{k}_{w_i\sigma}(X_i\sigma',x_i\sigma) \in \mathcal{H}_{\mathsf{e}}(K)$ for all $1 \leq i \leq m$, which implies by the definition of $\mathcal{H}_{\mathsf{e}}$ that there exists a substitution σ'' such that $\mathsf{k}_{w_i\sigma}(X_i\sigma'',x_i\sigma) \in \mathcal{H}(K)$ for all $1 \leq i \leq m$.

By instantiating f with $\sigma \cup (\sigma''[\{X_1,\ldots,X_m\}])$, we obtain that $\mathsf{r}_{l_1\sigma,\ldots,l_n\sigma} \in \mathcal{H}(K)$. By the soundness of K and of $\mathcal{H}$, it follows that $T \models \mathsf{r}_{l_1\sigma,\ldots,l_n\sigma}$. Therefore, there exist recipes R'_i (for all $1 \leq i \leq n$ such that $l_i = \mathbf{receive}(\_,t_i)$) such that $T \models \mathsf{k}_{l_1\sigma,\ldots,l_{i-1}\sigma}(R'_i,t_i\sigma)$. By the completeness of K, it follows that there exist recipes R_i such that $\mathsf{k}_{l_1\sigma,\ldots,l_{i-1}\sigma}(R_i,t_i\sigma) \in \mathcal{H}(K)$.

Let $M_i = l_i$ if $l_i \in \{\mathbf{test},\mathbf{send}(\_)\}$ and let $M_i = \mathbf{receive}(d_i,R_i)$ if $l_i = \mathbf{receive}(d_i,t_i)$ for all $1 \leq i \leq n$. Because the REACHABILITY test worked, it follows that there exists $S'_0 \in P$ such that, if we let $\psi'_0 = \emptyset$, we have

$$(S'_0,\psi'_0) \stackrel{M_1}{\Longrightarrow} (S'_1,\psi'_1) \stackrel{M_2}{\Longrightarrow} \ldots \stackrel{M_n}{\Longrightarrow} (S'_n,\psi'_n).$$

Similarly to Item 1, we can show that there exists $S'_0 \in P$ such that

$$(S'_0,\psi'_0) = (S'_0\pi,\psi_0\pi) \stackrel{L_1}{\Longrightarrow} (S'_1\pi,\psi'_1\pi) \stackrel{L_2}{\Longrightarrow} \ldots \stackrel{L_n}{\Longrightarrow} (S'_n\pi,\psi'_n\pi)$$

where π and π' are defined as in Item 1.

Furthermore, as the IDENTITY test worked, it follows that $(P\omega = P'\omega)\psi'_n$, where $\omega = \{X_i \mapsto x_i\sigma\}$. As the equational theory is stable by substitution of terms for names, we have that $(P\omega\pi' = P'\omega\pi')\psi'_n\pi$. But $P\omega\pi' = P\tau = Q$ and $P'\omega\pi' = P'\tau = Q'$, which means that $(Q = Q')\psi'_n\pi$.

We have shown that there exists $S'_0 \in P$ such that $(S'_0,\emptyset) \stackrel{L_1,\ldots,L_n}{\Longrightarrow} (\_,\psi'_n\pi)$ and $(Q = Q')\psi'_n\pi$. By determinacy of P, it follows that $(Q = Q')\psi$ for any ψ and any $U \in P$ such that $(U,\emptyset) \stackrel{L_1,\ldots,L_n}{\Longrightarrow} (\_,\psi)$, therefore we obtain a contradiction.

As both cases yield a contradiction, it follows that if $T \not\sqsubseteq P$, there exists at least one test that fails.

□

5.9 Tool Support and Termination

We conjecture that our saturation procedure always terminates. Unfortunately, we have not been able to prove this due to the highly complex form of the statements that are obtained during saturation.

We have however implemented the saturation procedure and the trace inclusion checking procedure in the tool AKISS (Active Knowledge IN Security protocols). The tool AKISS is written in OCaml and has around 2500 lines of source code, including code for computing strongly complete sets of finite variants and complete sets of equational unifiers for optimally reducing convergent equational theories (as described in Chapter 3).

The tool AKISS takes as input two processes P and Q in the form of a set of traces $P = \{S_1,\ldots,S_n\}$ and $Q = \{T_1,\ldots,T_m\}$. It then tries to check if the two processes are coarse trace equivalent. There are three possible results:

1. the tool does not terminate, in which case we cannot draw any conclusion. We conjecture that this situation never happens for subterm convergent rewrite systems (i.e. that AKiSs always terminates for such rewrite systems). However, due to the highly complex forms of the statements generated during the saturation process, we have not been able to prove it.

2. the tool terminates, tells that $P \not\approx_c Q$ and outputs a *proof* of non-trace-equivalence in the form of a sequence of labels $l_1, \ldots, l_n$ and two recipes r and r' such that $(P, \emptyset) \xrightarrow{l_1,\ldots,l_n} (S', \varphi)$, $(r = r')\varphi$ and either $(Q, \emptyset) \not\xrightarrow{l_1,\ldots,l_n}$ or $(Q, \emptyset) \xrightarrow{l_1,\ldots,l_n} (T', \varphi')$ for any φ' such that $(r \neq r')\varphi'$ (or vice-versa, the roles of P and Q being swapped).

   In this case, we are sure that $P \not\approx_t Q$ since $P \approx_t Q$ implies $P \approx_c Q$ (and we can hand-check the proof).

3. the tool terminates, tells that $P \approx_c Q$. There are two possibilities:

   (a) either P and Q are determinate, in which case by the correctness proof $P \approx_t Q$ by Theorem 5.1,

   (b) or at least one of P and Q is not determinate, in which case there is no guarantee.

5.9.1 Handling Non-determinate Processes

The above has a significant weakness: it does not allow to conclude that two processes which are not known to be determinate are equivalent. In order to circumvent this weakness, we propose in Figure 5.4 a sound way of checking (in)equivalence of two processes $P = \{S_1, \ldots, S_n\}$ and $Q = \{T_1, \ldots, T_m\}$ which are not known to be determinate. Unfortunately, the procedure is not complete in the sense that there are non-determinate processes P and Q for which we cannot conclude that $P \approx_t Q$ or that $P \not\approx_t Q$.

1. run AKiSs to check if $P \approx_c Q$. If AKiSs shows that $P \not\approx_c Q$, then the two processes are not trace equivalent.

2. otherwise, run AKiSs to check if $S_i \approx_c T_j$ for every trace $S_i \in P$ and every trace $T_j \in Q$. Since any individual trace S_i (or T_j) is determinate, we have by Theorem 5.1 that $S_i \approx_c T_j$ iff $S_i \approx_t T_j$ and therefore AKiSs will soundly determine if $S_i \approx_t T_j$ or not.

   It for every $i \in \{1, \ldots, n\}$ there exists a $j \in \{1, \ldots, m\}$ such that $S_i \approx_t T_j$ (and vice-versa, for every $j \in \{1, \ldots, m\}$ there exists an $i \in \{1, \ldots, n\}$ such that $S_i \approx_t T_j$), then it easily follows that $P \approx_t Q$.

3. otherwise, AKiSs cannot prove anything about P and Q and therefore some other technique needs to be used to show trace (in)equivalence of P and Q.

Figure 5.4: How to use AKiSs to check (in)equivalence of processes $P = \{S_1, \ldots, S_n\}$ and $Q = \{T_1, \ldots, T_m\}$ which are not known to be determinate.

We refer the reader to Example 5.4 for an example of two non-determinate processes that AKiSs cannot prove to be equivalent or inequivalent. This is because the two processes are coarse trace equivalent, but not trace equivalent. However, in our experiments on security protocols, we have not encountered such as situation (i.e. AKiSs was able to prove or dis-prove all equivalences for the examples that we have tried).

Also, on all our experiments with subterm convergent rewrite systems, the saturation process always terminated. We explain in the following sections some of our experiments.

5.9.2 The Running Example

The running example, consisting of the following traces

$$\begin{aligned} T &= \mathbf{send}(c, \mathrm{enc}(a,k)).\mathbf{send}(c, \mathrm{enc}(a',k)).\mathbf{receive}(c,x).[x \stackrel{?}{=} \mathrm{enc}(\mathrm{dec}(x,k),k)].\mathbf{send}(c,ok) \\ S_1 &= \mathbf{send}(c, \mathrm{enc}(a,k)).\mathbf{send}(c, \mathrm{enc}(a',k)).\mathbf{receive}(c,x).[x \stackrel{?}{=} \mathrm{enc}(a,k)].\mathbf{send}(c,ok) \\ S_2 &= \mathbf{send}(c, \mathrm{enc}(a,k)).\mathbf{send}(c, \mathrm{enc}(a',k)).\mathbf{receive}(c,x).[x \stackrel{?}{=} \mathrm{enc}(a',k)].\mathbf{send}(c,ok) \end{aligned}$$

where $a, a', k' \in \mathcal{N}$ are private names, where $ok \in \mathcal{M}$ is a public name and where $c \in \mathcal{C}$ is a public channel is interesting because it illustrates a subtlety of trace equivalence: even if the processes T and $\{S_1, S_2\}$ are trace equivalent, note that the trace T is trace-included neither in S_1 nor in S_2. However, T is included in the process consisting of both traces (S_1 and S_2).

This shows that even if two processes are trace-equivalent, there might be symbolic traces of the first process which are not included in any of the individual symbolic traces of the other process. Rather, the second process as a whole trace-includes each symbolic trace of the first process.

It can be easily proved that $\{S_1, S_2\}$ is a determinate process. Therefore our tool can directly prove that $T \approx_t \{S_1, S_2\}$. When given as input the two processes (T on one side and $\{S_1, S_2\}$ on the other side), it terminates instantaneously and proves their trace-equivalence.

When asked to check for the equivalence between T and either S_1 or S_2, the tool immediately shows that the two are not trace-equivalent and provides an appropriate counter-example.

If we did not know that $\{S_1, S_2\}$ is determinate, we could have used the workflow described in Figure 5.4 to check if $\{S_1, S_2\}$ and T are trace-equivalent or not. Unfortunately, since neither S_1 nor S_2 is trace-equivalent to T, we would not have been able to draw any conclusion.

5.9.3 The FOO E-voting Protocol

In this section we analyze vote privacy in a voting protocol (from hereon referred to as the 'FOO protocol') due to Fujioka, Okamoto and Ohta [87]. We provide an informal description of the protocol and we then proceed to show that it satisfies vote privacy in our symbolic model.

Description

The FOO voting protocol relies on two relatively non-standard cryptographic primitives:

1. commitments: by constructing the message $\mathrm{commit}(v, r)$ for some value v and some key r and sending it to some party, a participant *commits* to the value v with respect to the party having received the commitment.

   The commitment itself does not reveal the value v or the key r.

   Later on, when the participant is ready to reveal the value v to which he has previously commited, he sends in clear the key r to the other party. This party can now*open* the commitment with the key r obtaining $\mathrm{open}(\mathrm{commit}(v,r), r) =_{\mathsf{E}} v$ and obtain the value that was commited to.

2. blind signatures: a participant who wants to have an authority sign some message m, but without revealing the contents of m to the authority, can first *blind* the message m with a random *blinding factor* B to obtain $\mathrm{blind}(m, b)$.

   The authority can then sign the blinded message (if it chooses to do so) using its private key k to obtain $\mathrm{sign}(\mathrm{blind}(m,b),k)$. Later on, the participant can use the blinding factor to unblind the message signed by the authority, thereby obtaining $\mathrm{unblind}(\mathrm{sign}(\mathrm{blind}(m,b),k),b) =_{\mathsf{E}} \mathrm{sign}(m,k)$ the signature of the authority on the unblinded message.

We have briefly described the cryptographic primitives involved. We now describe how the protocol works. The voting takes place in three *phases* described bellow.

1. In the first phase, each voter V chooses a vote v, computes a commitment $\mathrm{commit}(v, r)$ of the vote v with a fresh key r, blinds the commitment using a random blinding factor b (obtaining $\mathrm{blind}(\mathrm{commit}(v, r), b)$ and signs the result with his private key, thereby obtaining the blob $\mathrm{sign}(\mathrm{blind}(\mathrm{commit}(v, r), b), k_V)$.

   The voter then sends this blob to an administrator accompanied by V's identity. The administrator verifies that V has the right to vote and did not vote before, and then signs using the administrator signature the blob (the blob is not signed by V at this point) and sends the result $\mathrm{sign}(\mathrm{blind}(\mathrm{commit}(v, r), b), k_A)$ to V.

   Once the voter V receives this message (presumably the signature of the administrator on his blinded vote), he then performs atomically the following actions: it checks that it was the administrator who has signed the received message (using the administrator's public key) and then unblinds that message to obtain $\mathrm{unblind}(\mathrm{sign}(\mathrm{blind}(\mathrm{commit}(v, r), b), k_A), b) =_{\mathsf{E}} \mathrm{sign}(\mathrm{commit}(v, r), k_A)$, i.e. his own commitment signed by the administrator. Note that in this process, V did not reveal his vote to the administrator.

2. The second phase is the actual voting phase: V sends the unblinding to the collector C on an anonymous channel, the collector verifies that the unblinding was signed by the administrator and, if the test succeeds, enters the commitment and the unblinding into the database.

3. The third and last phase starts once the collector decides it has received all votes (i.e. after the election ends).

   The collector publishes the database of votes. Each voter V verifies that their commitment is in the list and if so sends r, the commitment factor, to C via an anonymous channel.

   The collector C opens each ballot with the random r and publishes the vote v.

Modeling and Verifying the Protocol

To model the commitment scheme and the blind signatures, we use the following convergent optimally reducing term rewriting system:

$$\mathsf{R} = \left\{ \begin{array}{l} \mathrm{open}(\mathrm{commit}(x, y), y) \to x \\ \mathrm{check}(\mathrm{sign}(x, y), \mathrm{pk}(y)) \to x \\ \mathrm{unblind}(\mathrm{sign}(\mathrm{blind}(x, y), z), y), \mathrm{pk}(z)) \to \mathrm{sign}(x, z) \end{array} \right\}$$

that models the cryptographic primitives used in this case.

As we do not trust any of the election authorities, we assume they are dishonest participants and we give away their private key to the attacker by modeling the private key $k_{auth} \in \mathcal{M}$ of the authority as a public name. We model a voter $V \in \{A, B\} \subseteq \mathcal{C}$ (we identity voters with channels) voting $v \in \{\mathrm{yes}, \mathrm{no}\} \subseteq \mathcal{M}$ (each vote is represented by a public name) by a trace

$$P(V, v) = A_1(V, v).A_2(V, v).A_3(V, v).A_4(V, v)$$

where

$$\begin{array}{lcl} A_1(V, v) & = & \mathbf{send}(V, \mathrm{sign}(\mathrm{blind}(\mathrm{commit}(v, r_{V,v}), b_{V,v}), k_V)) \\ A_2(V, v) & = & \mathbf{receive}(V, x_{V,v}).[\mathrm{checksign}(x_{V,v}, \mathrm{pk}(k_{auth})) = \mathrm{blind}(\mathrm{commit}(v, r_{V,v}), b_{V,v})] \\ A_3(V, v) & = & \mathbf{send}(c, \mathrm{unblind}(x_{V,v}, b_{V,v})) \\ A_4(V, v) & = & \mathbf{send}(c, r_{V,v}) \end{array}$$

are the *atomic* actions of the honest participants. The names $b_{V,v}, r_{V,v} \in \mathcal{N}$ are private, $V \in \{A, B\}$ is the channel of voter V and $c \in \mathcal{C}$ is a public channel (so that the signed commitment can be transmitted anonymously). The actions A_1 and A_2 correspond to the first phase of the protocol, the action A_3 corresponds to the second phase and the action A_4 to the third phase.

Different voting situations can then be considered by constructing interleavings of traces $P(V, v)$ for several values of V and v in a phase-respecting manner. To prove that an attacker cannot tell

the difference between the situation where agent A votes yes and agent B votes no from the situation where the two agents swap their votes: agent A votes no and agent B votes yes, we consider the set $\Sigma(V, v, U, u)$ to contains all 24 traces resulting from phase-respecting interleavings of $P(V, v)$ and $P(U, u)$ as defined in Figure 5.5.

$$
\begin{array}{ll}
\mathit{Phase}_1(V, v, U, u) = & \{ \\
& A_1(V, v).A_2(V, v).A_1(U, u).A_2(U, u) \\
& A_1(V, v).A_1(U, u).A_2(V, v).A_2(U, u) \\
& A_1(V, v).A_1(U, u).A_2(U, u).A_2(V, v) \\
& A_1(U, u).A_1(V, v).A_2(V, v).A_2(U, u) \\
& A_1(U, u).A_1(V, v).A_2(U, u).A_2(V, v) \\
& A_1(U, u).A_2(U, u).A_1(V, v).A_2(V, v) \\
& \} \\
\mathit{Phase}_2(V, v, U, u) = & \{ \\
& A_3(V, v).A_3(U, u) \\
& A_3(U, u).A_3(V, v) \\
& \} \\
\mathit{Phase}_3(V, v, U, u) = & \{ \\
& A_4(V, v).A_4(U, u) \\
& A_4(U, u).A_4(V, v) \\
& \} \\
\Sigma(V, v, U, u) = & \{ \\
& P_1.P_2.P_3 \mid P_1 \in \mathit{Phase}_1, P_2 \in \mathit{Phase}_2, P_3 \in \mathit{Phase}_3, \\
& \}
\end{array}
$$

Figure 5.5: The set $\Sigma(V, v, U, u)$ of all 24 traces of the FOO e-voting protocol

We next consider a process modeling all traces of the FOO protocol restricted to two honest participants: a participant V who votes v and another participant U who votes u:

$$Q(V, v, U, u) = \{\mathbf{send}(c, \mathrm{pk}(k_A)).\mathbf{send}(c, \mathrm{pk}(k_B)).T \mid T \in \Sigma(V, v, U, u)\}.$$

In addition to the set of traces $\Sigma(V, v, U, u)$, the process $Q(V, v, U, u)$ also models the fact that the intruder has the public keys of the honest agents. In order to check that the FOO protocol satisfies vote privacy for two participants in our symbolic model, it is sufficient to establish (similarly to [72]) that

$$Q(A, \mathrm{yes}, B, \mathrm{no}) \approx_t Q(A, \mathrm{no}, B, \mathrm{yes}).$$

Unfortunately, we cannot directly apply our tool AKiSs to check trace equivalence because the two processes $Q(A, \mathrm{yes}, B, \mathrm{no})$ and $Q(A, \mathrm{no}, B, \mathrm{yes})$ are not determinate. However, the sufficient condition described in the second step of Figure 5.4 holds, i.e. AKiSs shows in a little under 3 minutes that for any trace $T \in Q(A, \mathrm{yes}, B, \mathrm{no})$ there exists $S \in Q(A, \mathrm{no}, B, \mathrm{yes})$ such that $T \approx_t S$ and vice-versa and therefore

$$Q(A, \mathrm{yes}, B, \mathrm{no}) \approx_t Q(A, \mathrm{no}, B, \mathrm{yes}).$$

We conclude that the FOO voting protocol indeed satisfies vote privacy in our symbolic model.

Note that our model only accounts for two voters and two votes. However, the same methodology can be used to establish vote privacy results for several voters and several possible votes (at the cost of an increase in running time).

5.9.4 Checking for Strong Secrecy

Usually, secrecy is checked in the Dolev-Yao model as a reachability property [107]. However, it is possible to define *stronger forms of secrecy* in the Dolev-Yao model based on process equivalences.

Blanchet [26] provides such a definition of strong secrecy based on observational equivalence in a variant of the applied-pi calculus. According to Blanchet [26], a process $P(\tilde{x})$ with a set of free variables $\tilde{x}$ preserves the strong secrecy of its free variables $\tilde{x}$ if and only if,

$$\text{for all grounding substitutions } \sigma, \sigma' \text{ of domain } \tilde{x}, \text{ we have } P\sigma \approx P\sigma'.$$

In the above definition, the substitution application avoids name capture (i.e. σ, σ' are not allowed to contain private names). Also, by the definition of [26], $\approx$ denotes observational equivalence. We can model the same type of strong secrecy in our process calculus, but relying on trace equivalence instead of observational equivalence:

Definition 5.14. Let P be a process with free variables $\{x_1, \ldots, x_n\}$. The process P satisfies strong secrecy of its free variables if and only if:

$$\begin{gathered}\mathbf{receive}(c, x_1).\ldots.\mathbf{receive}(c, x_n).\mathbf{receive}(c, x'_1).\ldots.\mathbf{receive}(c, x'_n).P \\ \approx_t \\ \mathbf{receive}(c, x_1).\ldots.\mathbf{receive}(c, x_n).\mathbf{receive}(c, x'_1).\ldots.\mathbf{receive}(c, x'_n).P\{x_i \mapsto x'_i\}_{1\le i\le n},\end{gathered}$$

where $c \in \mathcal{C}$ is a public channel.

We can see that the two definitions (the one from [26] and Definition 5.14) coincide (other than the change from $\approx$ to $\approx_t$) by noting that the universal quantification on σ, σ' and the fact that σ, σ' do not capture private names are modeled by having the intruder choose the values of $x_1, \ldots, x_n, x'_1, \ldots, x'_n$ at the beginning of each process. We consider the running example from [26], a key-distribution protocol:

$$\begin{aligned} A \to B: &\quad \mathrm{aenc}(\mathrm{sign}(\mathrm{pair}(\mathrm{pk}_A, \mathrm{pair}(\mathrm{pk}_B, k)), \mathrm{sk}_A), \mathrm{pk}_B) \\ B \to A: &\quad \mathrm{enc}(x, k). \end{aligned}$$

In this protocol, the two participants A and B wish to establish a shared key k for subsequent communication. The participant A creates the key k, and sends it to B together with the identity pk_A of A and pk_B of B. The entire message is signed by A using his private key sk_A and encrypted under the public key pk_B of B. The participant B can then send to A a message $\mathrm{enc}(x, k)$ encrypted under the freshly created key k. The variable x is free and represents the contents of the message. We will next analyze the strong secrecy of x in this protocol.

We model the cryptographic primitives used (public key encryption, digital signatures, symmetric encryption and pairs) in a standard fashion using the following convergent optimally reducing term rewriting system R:

$$\begin{aligned} \mathsf{R} = \{ \quad & \mathrm{adec}(\mathrm{aenc}(x, \mathrm{pk}(y)), y) \to x \\ & \mathrm{dec}(\mathrm{enc}(x, y), y) \to x \\ & \mathrm{checksign}(\mathrm{sign}(x, y), \mathrm{pk}(y)) \to \mathrm{ok} \\ & \mathrm{msg}(\mathrm{sign}(x, y)) \to x \\ & \mathrm{fst}(\mathrm{pair}(x, y)) \to x \\ & \mathrm{snd}(\mathrm{pair}(x, y)) \to x \\ \}. & \end{aligned}$$

We model one session of this protocol as follows. We let $k, \mathrm{sk}_B, \mathrm{sk}_A \in \mathcal{N}$ by private names representing the freshly chosen symmetric k and respectively the secret keys of A and of B. We consider a single public channel $c \in \mathcal{C}$ and two processes

$$\begin{aligned} &P_A = \mathbf{send}(c, \mathrm{aenc}(\mathrm{sign}(\mathrm{pair}(\mathrm{pk}(\mathrm{sk}_A), \mathrm{pair}(\mathrm{pk}(\mathrm{sk}_B), k)), \mathrm{sk}_A), \mathrm{pk}(\mathrm{sk}_B))) \\ &P_B(x) = \mathbf{receive}(c, y). \\ &\quad [\mathrm{checksign}(\mathrm{adec}(y, \mathrm{sk}_B), \mathrm{pk}(\mathrm{sk}_A)) \stackrel{?}{=} \mathrm{ok}]. \\ &\quad [\mathrm{fst}(\mathrm{msg}(\mathrm{adec}(y, \mathrm{sk}_B))) \stackrel{?}{=} \mathrm{pk}(\mathrm{sk}_A)]. \\ &\quad [\mathrm{fst}(\mathrm{snd}(\mathrm{msg}(\mathrm{adec}(y, \mathrm{sk}_B)))) \stackrel{?}{=} \mathrm{pk}(\mathrm{sk}_B)]. \\ &\quad \mathbf{send}(c, \mathrm{enc}(x, \mathrm{snd}(\mathrm{snd}(\mathrm{msg}(\mathrm{adec}(y, \mathrm{sk}_B)))))) \end{aligned}$$

modeling the actions of participant A communicating to B and that of participant B communicating to A. We denote by $P(x)$ all 6 interleavings of actions of P_A and of $P_B(x)$. The strong secrecy of x (for one session of the protocol) is then modeled by the following equivalence:

$$\begin{gathered}\mathbf{receive}(c,x).\mathbf{receive}(c,x').\mathbf{send}(c,\mathrm{pk}(\mathrm{sk}_A)).\mathbf{send}(c,\mathrm{pk}(\mathrm{sk}_B)).P(x)\\ \approx_t\\ \mathbf{receive}(c,x).\mathbf{receive}(c,x').\mathbf{send}(c,\mathrm{pk}(\mathrm{sk}_A)).\mathbf{send}(c,\mathrm{pk}(\mathrm{sk}_B)).P(x').\end{gathered}$$

Sending $\mathrm{pk}(\mathrm{sk}_A)$ and $\mathrm{pk}(\mathrm{sk}_B)$ on the network models the fact that the intruder has the public keys of A and of B. Inputting x and x' follows our definition of strong secrecy (Definition 5.14). The notation $a.P$ where a is an action and P is a process denotes the process $\{a.T \mid T \in \mathcal{P} \text{ is a trace of } P\}$.

We have checked the above equivalence using AKiSs and we have proved that it holds. We conclude that this protocols satisfies strong secrecy of x (for one session). We have also checked strong secrecy for the different variations of this protocol described in [26]. The variations introduce violations of strong secrecy and fixes for these violations in order to re-obtain strong secrecy. We obtained using AKiSs the expected results (but for only one session of the protocol).

5.9.5 Checking for Guessing Attacks

We can distinguish secrets used in cryptographic protocols as being *weak secrets* and *strong secrets*. Strong secrets such as encryption keys have a bit-length which makes it intractable for an attacker to completely enumerate them. In contrast, weak secrets such as PINs or passwords can be enumerated in a more reasonable time-frame (e.g. assuming that a password is an English word, it is sufficient to enumerate all words in an English dictionary). Weak secrets are useful as they are easier to remember by humans. They are used for example to authenticate two participants as the following example protocol shows:

$$\begin{aligned}A \to B:&\quad \mathrm{enc}(n,w)\\ B \to A:&\quad \mathrm{enc}(h(n),w).\end{aligned}$$

In this protocol, the participants A and B share a weak secret w. The goal of the protocol is to authenticate B from A's point of view. To do this, A picks a fresh name n and send a *challenge* $\mathrm{enc}(n,w)$ to B. If B knows w, he can decrypt the challenge, compute the hash $h(n)$ of the name n freshly chosen by A and encrypt the hash $h(n)$ with w to send it to A. The participant A can then check that the decryption of what he received is the hash of the fresh name. In this case, A is convinced to be talking to B, since B has "proved" by decrypting and encrypting with w that we knows the weak secret w.

Suppose that an intruder records all of the messages $\varphi = \{w_1 \mapsto \mathrm{enc}(n,w), w_2 \mapsto \mathrm{enc}(h(w),w)\}$ transmitted in a session between A and B. As w is a weak secret, he can enumerate all of the possible values v of w offline (i.e. after the session has ended). He can then check if a value v is the correct one (i.e. is equal to w) by performing the following test:

$$h(\mathrm{dec}(w_1,v)) \stackrel{?}{=} \mathrm{dec}(w_2,v).$$

When $v = w$, the test will succeed and when $v \neq w$, the test will fail. Therefore we say that the protocol makes w vulnerable to a guessing attack (or equivalently to an off-line dictionary attack). We can model that a private name w appearing in a process P is resistant against an offline dictionary attack by the following equivalence:

$$P.\mathbf{send}(c,w) \approx_t P.\mathbf{send}(c,v).$$

In the above equivalence $c \in \mathcal{C}$ is a public channel, $w, v \in \mathcal{N}$ are private names with w representing the weak secret and v being a fresh name. The notation $P.a$ where P is a process and a is an action denotes the process $\{T.a \mid T \in P \text{ is a trace of } P\}$. The intuition behind the equivalence

is that the attacker first records all messages output by P. Then he tries to guess the value of w (by enumerating all its values). This guessing is modeled by the two actions $\mathbf{send}(c, w)$ and $\mathbf{send}(c, v)$. If the two sides of the equivalence are indistinguishable, it means that the attacker has no test to distinguish between the real value w of the weak secret and another value v.

We now proceed to analyze resistance against guessing attacks for the example protocol that we have described above. We can model the roles of A and of B in the above protocol as the following traces:

$$P_A = \mathbf{send}(c, \mathrm{enc}(n, w))$$
$$P_B = \mathbf{receive}(c, x).\mathbf{send}(c, \mathrm{enc}(h(\mathrm{dec}(x, w)), w)).$$

The participant A therefore simply sends the encryption of a fresh name $n \in \mathcal{N}$ with the weak secret $w \in N$ on the network. The participant B reads x (the message that A supposedly sent) from the network and outputs on the network the solution to A's challenge. We model the underlying rewrite system as

$$\mathsf{R} = \{ \quad \mathrm{dec}(\mathrm{enc}(x, y), y) \to x \\ \qquad\quad \mathrm{enc}(\mathrm{dec}(x, y), y) \to x \\ \}.$$

The first equation is standard and the second equation models that the encryption scheme is not *key revealing* (otherwise there would be a trivial attack: the attacker could test w_1 is an encryption with the value v that he has guessed: $\mathrm{enc}(\mathrm{dec}(w_1, v), v) = w_1$). Resistance against guessing attacks of w then reduces to the equivalence

$$P.\mathbf{send}(c, w) \approx_t P.\mathbf{send}(c, v)$$

where $v \in \mathcal{N}$ is a fresh name. The tool AKISS shows that the two processes are not coarse trace equivalent (and therefore not trace equivalent) and outputs the following test which holds when $X = P.\mathbf{send}(c, w)$ but does not hold when $X = Q.\mathbf{send}(c, v)$:

$$(X, \emptyset) \xrightarrow{\mathbf{send}(c),\mathbf{receive}(c,w_1),\mathbf{test},\mathbf{send}(c),\mathbf{send}(c)} (Y, \varphi) \text{ s.t. } (\mathrm{dec}(w_2, w_3) \stackrel{?}{=} h(\mathrm{dec}(w_1, w_3)))\varphi.$$

Therefore the equivalence does not holds and we conclude that P is not resistant against guessing attacks of w. We now consider a variant of the EKE protocol [23]:

$$A \to B : \quad \mathrm{enc}(\mathrm{pk}(k), w)$$
$$B \to A : \quad \mathrm{enc}(\mathrm{aenc}(r, \mathrm{pk}(k)), w).$$

The goal of the EKE protocol is to establish a session key r between two participants who share a weak secret w. To do so, the initiator A generates a fresh private key k and sends to B the associated public key $\mathrm{pk}(k)$ encrypted under w. The participant B then creates the fresh session key r and send this key to A wrapped by asymmetrically encrypting it with $\mathrm{pk}(k)$ and then symmetrically with w. In the EKE protocol a challenge-response phase meant to prevent replay attacks follows but we do not model it since we concentrate on resistance against guessing attacks on w. We model the two participants as the following traces

$$P_A = \mathbf{send}(c, \mathrm{enc}(\mathrm{pk}(k), w))$$
$$P_B = \mathbf{receive}(c, x).\mathbf{send}(c, \mathrm{enc}(\mathrm{aenc}(r, \mathrm{dec}(x, w)), w)),$$

where $c \in \mathcal{C}$ is a public channel and $r, k, w \in \mathcal{N}$ are private names. Then an interleaving of P_A and P_B represents one session of the EKE protocol. We consider two more traces:

$$P'_A = \mathbf{send}(c, \mathrm{enc}(\mathrm{pk}(k'), w))$$
$$P'_B = \mathbf{receive}(c, x').\mathbf{send}(c, \mathrm{enc}(\mathrm{aenc}(r', \mathrm{dec}(x', w)), w)),$$

with $r', k' \in \mathcal{N}$ modeling a second session of the same protocol. Then the process P consisting of all interleavings of P_A, P_B, P'_A, P'_B model two concurrent sessions of the EKE protocol between

participants A and B. We consider the term rewriting system

$$\begin{array}{ll}\mathsf{R} = \{ & \mathrm{dec}(\mathrm{enc}(x,y),y) \to x \\ & \mathrm{enc}(\mathrm{dec}(x,y),y) \to x \\ & \mathrm{adec}(\mathrm{aenc}(x,\mathrm{pk}(y)),y) \to x \\ \}. & \end{array}$$

The first and the third rewrite rules are standard. The second rewrite rule models the fact that the symmetric encryption scheme is *key-concealing* to prevent an attacker from checking if a ciphertext has been encrypted using a known key. We have shown using AKISs that

$$P.\mathbf{send}(c,w) \approx_t P.\mathbf{send}(c,v)$$

for a fresh private name $v \in \mathcal{N}$ and we can therefore conclude that EKE satisfies resistance against guessing attacks on w for two sessions.

5.10 Conclusion and Further Work

This chapter of the thesis is concerned with the verification of trace equivalence for security protocols. We obtain a procedure for verifying trace equivalence for finite, determinate processes without else branches under convergent optimally reducing rewrite systems. To handle non-determinate processes, we present an effective proof method for trace equivalence based on the procedure for determinate processes. We conjecture that the procedure always terminates for subterm convergent term rewriting systems, but unfortunately, due to the highly complex form of Horn clauses obtained by the procedure, we have not been able to prove this.

We have implemented this procedure in the tool AKISs and we have used the procedure to run several experiments. Although the implementation works quite fast for checking if one symbolic trace is trace-included in a determinate process, the number of symbolic traces of a process can be exponential and therefore an exponential number of such trace-inclusions need to be verified in order to check that two processes are trace-equivalent. We have used our tool to provide the first provably sound and automatic [1] proof of vote privacy in the FOO e-voting protocol (Section 5.9.3). As future work, there are several tracks which deserve to be investigated:

1. Prove that the saturation procedure presented in the second part of the chapter terminates. This would provide a very nice theoretical result that is also easy to implement in practice.

2. Get rid of the explicit enumeration of symbolic traces. The definition of a process as a set of all its symbolic traces is fine from a theoretical point of view, as long as complexity does not enter the picture. However, from a practical point of view, the unavoidable exponential blowup is very bad for the running time and the explicit enumeration of symbolic traces is very error-prone.

3. Get rid of the hypothesis of determinate processes. E-voting protocols, which represent the most important application for checking equivalence properties, are naturally not determinate. This means that approximations must be used in order to prove equivalences, such as the approximation that we used when verifying the FOO e-voting protocol.

4. Get rid of the "no else branch" hypothesis. In our current work, we only allow processes that perform tests and, if the test succeeds, they continue; otherwise they block. In practice, there are protocols which may do arbitrary actions in the "otherwise" branch. Verifying such protocols is a worthwhile goal.

[1] The work in Chapter 5 of [119] also allows automatic verification of FOO by using an add-on to ProVerif called ProSwapper; however the soundness of the add-on itself was never proven and therefore there is not formal guarantee of the result of the verification.

5. Perform sound abstractions. Instead of having fully-abstract models of traces, construct sound models that allow proving equivalence properties. Not having to be fully-abstract may allow for better efficiency and could allow a tool to make security proofs for an unbounded number of sessions.

6. Work modulo AC or other equational theories. There are a number of cryptographic primitives such as Diffie-Hellman or XOR which cannot be oriented as a convergent term rewriting system. Working modulo AC (associative-commutative) operators could allow a tool to prove security properties under such cryptographic primitives.

Chapter 6

Composability

6.1 Introduction

Most of existing techniques for proving protocols secure are dedicated to the analysis of a protocol, without taking into account other protocols which may be used at the same time. However, most cryptographic protocols do not run in isolation. Instead, several protocols can run on the same network and there might be undesirable interactions between protocols which cause security problems.

Even apparently isolated protocols might interact in unexpected ways. For example, a user might choose the same password for two different network services, or a server might use the same key for different protocols. Even if the network services (or the different protocols) were proven secure in isolation, there is no security guarantee which carries over when they share keys or passwords.

Furthermore, a number of protocols are verified under the assumption that agents share some pre-distributed keys (e.g. public keys or symmetric keys between agents and servers). But these keys might have been established by some other sub-protocols. There is no guarantee that a protocol remains secure if a specific key-exchange protocol is used to establish the keys, even if both protocols have been proven secure in isolation.

Even assuming that we could produce a global model of all protocols which are used in a certain setting, it might be unrealistic to formally verify such a collection of protocols in its entirety due to computational constraints.

Therefore more modular reasoning about security is desirable, where we can infer security guarantees for the composition of protocols from the security guarantees of the individual protocols. The goal of this chapter is to study the composition of protocols. Which security properties should P_1 and P_2 have such that running P_1 and P_2 on the same network (with possible information being passed between P_1 and P_2) is secure. In particular, given a protocol P_1 that has been proven secure assuming pre-established keys or assuming some secure channel, under which conditions does P_1 remain secure if it uses P_2 as a sub-protocol to establish some of keys?

6.1.1 Related Work

One of the first papers studying the composition of protocols in the Dolev-Yao model is [93]. In the formalism of strand spaces [85], Guttman and Thayer show that two protocols which make use of concatenation and encryption can be safely executed together without damaging interactions, as soon as the protocols are "independent". Also, an assumption is made that all keys are atomic and not generated for example by hashing some message. The independence hypothesis requires in particular that the sets of encrypted messages handled by the two protocols be disjoint. This is a semantic hypothesis on all possible executions of the two protocols which needs to be checked by hand.

In [57], Cortier *et al* show that tagging is sufficient to avoid collusion between protocols sharing common keys and making use of standard cryptographic primitives: concatenation, signature, hash functions and encryption. This framework allows to compose processes symmetrically; however, it does not allow to securely compose e.g. a key exchange protocol with another protocol which makes use of the shared key. In particular, this is because the shared keys should never appear as payloads.

In [92], Guttman provides a non-trivial characterization which ensures that two protocols can run securely together when sharing some data such as keys or payloads. The main improvement over [93] is that keys are allowed to be non-atomic. The characterization is syntactic but has to be computed for each pair of protocols. As cryptographic primitives, the protocols are allowed to contain encryptions and concatenations. The proof method in our result is roughly similar to the proof methods described here: an attack against the composition is transformed into an attack against one of the two protocols.

In [71], Delaune *et al* use a derivative of the applied-π calculus to model off-line guessing attacks. They show that in the passive case resistance against guessing attacks is preserved by the composition of two protocols which share the weak secret against which the attack is mounted. This result (in the passive case) holds for arbitrary equational theories. However, for the active case this is no longer the case: it is however proven that *tagging* the weak secret enforces secure composition (in the sense of guessing attacks). Again, this framework applies to parallel composition only.

Mödersheim and Viganò [109] have proposed a framework for composing protocols sequentially. They propose a criterion for a protocol P_1 to safely use P_2 as a sub-protocol (for implementing a secure channel). However, their criterion is a semantic criterion, for which no decision procedure has been provided yet. In [91], Groß and Mödersheim introduce *vertical* protocol compositions, where a key-exchange protocol is coupled with a protocol (called application protocol) which uses the exchanged key. Vertical compositions are similar to the sequential compositions we have used in this work. They identify a number of preconditions which, if satisfied by a set $\mathcal{P}$ of key-exchange protocols and application protocols, allow arbitrary vertical compositions among the protocol in $\mathcal{P}$. Their work also allows for self-composition, in the sense that an application protocol can serve as a key-exchange protocol for yet another application protocol.

In [70], Delaune *et al* use a simulation based approach inspired by work in the computational model [33] to provide a framework for securely composing protocols in the applied-π calculus. This involves defining for each sub-protocol an *ideal functionality* and then showing that a certain implementation securely emulates the ideal functionality. Another line of work is represented by the Protocol Composition Logic [67], which can be used to modularly prove security properties of protocols using a fixed set of primitives. However, the proofs may be rather involved and are not automatic.

6.1.2 Contribution

In this chapter, we propose a generic composition result for arbitrary cryptographic primitives which can be modeled by equational theories. More precisely, we show that an attack trace against the composition of two protocols can be transformed into an attack trace against one of the two protocols.

The composition theorem holds for any cryptographic primitives which can be modeled by equational theories, provided that the signatures of the two composed protocols are disjoint. This allows us to handle many cryptographic primitives: symmetric and asymmetric encryption, hash functions, messages authentication codes, signatures, blind signatures, re-encryption, zero-knowledge proofs and others [18, 72]. We can also allow some common primitives between the two protocols, such as encryption and hash, provided that they are tagged. In order to formally describe and reason about protocols and their compositions, we introduce in Section 6.2 a slightly different process algebra than in the previous chapter. This (new) process algebra contains actions such as νn and $|$ and is more adapted to describing compositions of protocols than the one used in the previous chapter. We concentrate on proving secrecy properties of compositions although we believe that our result carries over to other trace properties such as authentication.

Our main theorem is generic in the sense that the composition can be any interleaving of actions from the two protocols: for example, the composition can be parallel or sequential, possibly with nested replication. In particular, we capture the case where a protocol uses a sub-protocol to e.g. establish keys. As a consequence, we can for example easily compose a protocol using Diffie-Hellman exponentiation for establishing symmetric keys, together with any protocol making use of pre-established keys.

Applications. Our main composition result can be used in different contexts. As an application, we study the case of key-exchange protocols. We first consider the case where a key-exchange protocol is used to establish shared long-term keys.

Assume that $P = \nu n.(P_1 \mid P_2)$ is a protocol that establishes a key between two participants. The process P denotes the parallel execution of a participant modeled by P_1 and another participant modeled by P_2. The two agents P_1 and P_2 share the previously distributed key n. The goal of the protocol P is to establish a fresh shared session key between P_1 and P_2. Assume that the key will be stored in the variable y_1 for P_1 and the variable y_2 for P_2.

We identify which properties should be satisfied by P in order to be safely used within any other protocol. As expected, we retrieve the fact that the established key (stored in y_1 for P_1 and in y_2 for P_2) should remain secret to an attacker, but we also point out two other important properties which are not always checked when verifying key-exchange protocols.

We show that whenever P satisfies our identified properties and if a protocol Q is secure assuming pre-established keys (e.g. if Q preserves the secrecy of some data s):

$$Q = \nu k.((y_1 := k).Q_1 \mid (y_2 := k).Q_2) \models \mathsf{Secret}(s)$$

then Q remains secure when running P as subprotocol for establishing the secret key (in y_1 for the first participant and in y_2 for the second participant):

$$\nu n \cdot (P_1 \cdot Q_1 \mid P_2 \cdot Q_2) \models \mathsf{Secret}(s).$$

We also consider the case where a key-exchange protocol is used within each session for establishing a secure channel. We show that if a protocol Q' is secure assuming a secure channel:

$$Q' = !(\nu k.((y_1 := k).Q_1 \mid (y_2 := k).Q_2)) \models \mathsf{Secret}(s)$$

then Q' remains secure when running P as subprotocol:

$$!(\nu n.(P_1.Q_1 \mid P_2.Q_2)) \models \mathsf{Secret}(s).$$

Our approach has several advantages over prior work. As opposed to [93, 57, 92, 109, 91], our result allows not only the standard cryptographic primitives like encryption and hash functions, but arbitrary primitives expressible as equational theories. Furthermore, unlike [57, 71], our result allows to compose protocols asymmetrically (i.e. not just in parallel). A difference between our approach and [70] is that we do not need to prove anything about the protocols we are trying to compose except standard reachability properties. In particular, we do not have to provide a key exchange functionality and prove that an implementation satisfies this functionality. However, [70] can be used to reason about protocols which do share primitives.

This chapter is based on work [49] that has been published at CSF 2010.

Plan of the Chapter. In Section 6.2 we introduce a process algebra which is suitable for defining composition of protocols. In Section 6.3 we identify difficulties in composing protocols and we give examples of compositions that "go wrong". Once we rule out the problematic cases that we identified, we propose a generic composition theorem for protocols employing disjoint cryptographic primitives in Section 6.4. We present applications of this theorem to key establishment protocols in Section 6.5. We show that a form of *tagging* is sufficient to enforce disjointness of cryptographic protocols making use of symmetric encryption and hash functions in Section 6.6. We conclude our work in Section 6.7 with perspectives for further work.

6.2 Process Algebra

We now introduce the process algebra that we use to describe protocols and their composition. As opposed to the process algebra that we have used in the previous chapter, this process algebra is tailored to easily describe protocol compositions and reason about their security. The process algebra closest to ours is the applied π-calculus [3]. However, the applied π-calculus is not adequate in our case since it makes formulating our main theorem unnecessarily cumbersome. The main differences between our calculus and the applied π-calculus are the following ones:

- we add a *synchronization phase*, so that we can write $P.Q$ for arbitrary processes P and Q. This is important to express the fact that a protocol first runs P before continuing with Q,
- we consider only one public channel,
- only positive tests are allowed (no else branches).

Processes are defined inductively in Figure 6.1.

$P, Q \ldots$	::=		processes	
	|	0		*(null process)*
	|	νx	for $x \in \mathcal{X}$	*(fresh name)*
	|	$\mathbf{receive}(x)$	for $x \in \mathcal{X}$	*(input)*
	|	$(x := t)$	for $x \in \mathcal{X}, x \notin \mathcal{V}ar(t), t \in T(\mathcal{F}, \mathcal{X})$	*(assignment)*
	|	$\mathbf{send}(t)$	for $t \in T(\mathcal{F}, \mathcal{M} \cup \mathcal{X})$	*(output)*
	|	$[s = t]$	for $s, t \in T(\mathcal{F}, \mathcal{M} \cup \mathcal{X})$	*(test)*
	|	$(P.Q)$		*(sequential composition)*
	|	$(P \mid Q)$		*(parallel composition)*
	|	$!P$		*(replication)*

Figure 6.1: Process Algebra

The null process (or the empty process) 0 does nothing. The process νx binds x to a fresh private name. The process $\mathbf{receive}(x)$ reads a term t from the public channel, and binds x to t. The assignment process $(x := t)$ instantiates x with t. The process $\mathbf{send}(t)$ outputs the term t on the public channel. The test process $[s = t]$ blocks if $s \neq_{\mathsf{E}} t$ and does nothing otherwise. The sequential composition process $P.Q$ executes P followed by Q. The parallel composition process $(P \mid Q)$ runs P and Q in parallel. The replication process $!P$ will act as an infinite number of Ps in parallel. We may write $\nu\{x_1, \ldots, x_k\}$ instead of $\nu x_1 \ldots . \nu x_k$.

When we write processes, we assume that "!" binds strongest, followed by "|" and then by ".". We assume that the variables introduced by νx, $\mathbf{receive}(x)$ and $(x := t)$ are bound throughout sequential composition (denoted by ".") as far to the right as possible. By $fv(P)$ we denote the set of free variables of P. We identify processes up to α-renaming of bound variables and up to the following structural equivalence rules in Figure 6.2.

$$P \equiv P.0 \equiv 0.P \qquad P \mid 0 \equiv P \qquad (P \mid Q) \mid R \equiv P \mid (Q \mid R)$$
$$!0 \equiv 0 \qquad P \mid Q \equiv Q \mid P \qquad !P \equiv P \mid !P$$

Figure 6.2: Structural Equivalence Rules

Example 6.1. Let $\mathcal{F}_{DH} = \{f, g, \mathrm{mac}\}$ be a signature where g is a function symbols if arity 1 and where f, mac are function symbols of arity 2. Together with the equational theory

$$\mathsf{E}_{DH} = \{f(g(y), x) = f(g(x), y)\}$$

the function symbols f and g model the Diffie-Hellman primitives ($f(x,y) = x^y \mod p$, $g(y) = \alpha^y \mod p$ for a generator α) and mac denotes a keyed hash function.

We have that the process $P_{DH} = \nu x_k.(P_1 \mid P_2)$ models for the Diffie-Hellman protocol if

$$\begin{aligned} P_1 &= \nu x.\mathbf{send}(g(x)).\mathbf{send}(\mathrm{mac}(g(x),x_k)).\mathbf{receive}(z).\mathbf{receive}(z'). \\ &\quad [z' = \mathrm{mac}(z,x_k)].y_1 := f(x,z) \\ &\qquad\qquad \text{and} \\ P_2 &= \nu y.\mathbf{send}(g(y)).\mathbf{send}(\mathrm{mac}(g(y),x_k)).\mathbf{receive}(z).\mathbf{receive}(z'). \\ &\quad [z' = \mathrm{mac}(z,x_k)].y_2 := f(y,z). \end{aligned}$$

The process P_1 models the first participant in an authenticated Diffie-Hellman key exchange while P_2 models the second participant. The free variable x_k should be previously shared by the two participants to ensure authentication.

Processes are executed within an environment formed of a *frame* φ that contains messages sent over the network (as in the applied π-calculus) and a *binding substitution* σ whose domain is a subset of the free variables of the process. The formal operational semantics is given by the transition relations $\rightarrow_{\mathcal{F},\mathsf{E}}$ and $\xrightarrow{t}_{\mathcal{F},\mathsf{E}}$ (where t is a label $\mathbf{receive}(r)$) described in Figure 6.3.

$$\textsc{New}\ \frac{P = \nu x.R \qquad x \notin \mathcal{D}om(\sigma) \qquad n \in \mathcal{N} \text{ fresh}}{(P,\varphi,\sigma) \rightarrow_{\mathcal{F},\mathsf{E}} (R,\varphi,\sigma \cup \{x \mapsto n\})}$$

$$\textsc{Input}\ \frac{P = \mathbf{receive}(x).R \qquad \varphi \vdash^{r}_{\mathcal{F},\mathsf{E}} t \qquad x \notin \mathcal{D}om(\sigma)}{(P,\varphi,\sigma) \xrightarrow{\mathbf{receive}(r)}_{\mathcal{F},\mathsf{E}} (R,\varphi,\sigma \cup \{x \mapsto t\})}$$

$$\textsc{Assgn}\ \frac{P = (x := t).R \qquad x \notin \mathcal{D}om(\sigma) \qquad \mathcal{V}ar(t) \subseteq \mathcal{D}om(\sigma) \qquad t\sigma =_{\mathsf{E}} t'}{(P,\varphi,\sigma) \rightarrow_{\mathcal{F},\mathsf{E}} (R,\varphi,\sigma \cup \{x \mapsto t'\})}$$

$$\textsc{Output}\ \frac{P = \mathbf{send}(t).R \qquad \mathcal{V}ar(t) \subseteq \mathcal{D}om(\sigma) \qquad t' =_{\mathsf{E}} t\sigma}{(P,\varphi,\sigma) \rightarrow_{\mathcal{F},\mathsf{E}} (R,\varphi \cup \{w_{|\mathcal{D}om(\varphi)|+1} \mapsto t'\},\sigma)}$$

$$\textsc{Test}\ \frac{P = [s = t].R \qquad s\sigma =_{\mathsf{E}} t\sigma \qquad \mathcal{V}ar(s),\mathcal{V}ar(t) \subseteq \mathcal{D}om(\sigma)}{(P,\varphi,\sigma) \rightarrow_{\mathcal{F},\mathsf{E}} (R,\varphi,\sigma)}$$

$$\textsc{Parallel}\ \frac{P = (Q_0 \mid Q_1).R \qquad (Q_0,\varphi,\sigma) \xrightarrow{\S}_{\mathcal{F},\mathsf{E}} (Q_0',\varphi',\sigma')}{(P,\varphi,\sigma) \xrightarrow{\S}_{\mathcal{F},\mathsf{E}} ((Q_0' \mid Q_1).R,\varphi',\sigma')}$$

Figure 6.3: Operational Semantics.

The New operation chooses a fresh name n, binds x to n and moves to the next action. The Input operation waits for a term t constructible by the recipe r from the frame φ containing the messages already exchanged on the network, binds x to t and moves to the next action. The Assign operation binds x to $t\sigma$ and continues to the next action. The Output operation adds to the frame the term $t\sigma$ being output. The Test operation moves to the next operation if the two terms being compared are equal and *blocks* otherwise. The Parallel operation states that a parallel composition can proceed as either of the two processes being run in parallel.

In Figure 6.3, § is a meta-variable denoting either the empty string, in which case $\xrightarrow{\S}_{\mathcal{F},\mathsf{E}} = \rightarrow_{\mathcal{F},\mathsf{E}}$, or a label $x = \mathbf{receive}(r)$, in which case $\xrightarrow{\S}_{\mathcal{F},\mathsf{E}} = \xrightarrow{\mathbf{receive}(r)}_{\mathcal{F},\mathsf{E}}$. The relation $\xrightarrow{l_1 \cdot \ldots \cdot l_n}{}^{*}_{\mathcal{F},\mathsf{E}}$ is defined as $\rightarrow^{*}_{\mathcal{F},\mathsf{E}} \xrightarrow{l_1} \rightarrow_{\mathcal{F},\mathsf{E}} \rightarrow^{*}_{\mathcal{F},\mathsf{E}} \xrightarrow{l_2} \rightarrow_{\mathcal{F},\mathsf{E}} \rightarrow^{*}_{\mathcal{F},\mathsf{E}} \ldots \rightarrow^{*}_{\mathcal{F},\mathsf{E}} \xrightarrow{l_n} \rightarrow_{\mathcal{F},\mathsf{E}} \rightarrow^{*}_{\mathcal{F},\mathsf{E}}$. We omit $\mathcal{F}$ and E when they are clear from the context.

Examples of process executions can be found in Section 6.3. We conclude this section by defining secrecy and freshness. Secrecy and freshness are two properties that we require of protocols in order to compose them securely. Intuitively, a variable is secret if in any protocol execution, its instantiation remains not deducible. We will require variables that are shared between two protocols to be kept secret by both protocols in order to securely compose them. Otherwise, if a protocol reveals a shared variable, it might create a dangerous situation for the other protocol as exemplified in Section 6.3.1.

Definition 6.1 (secrecy)**.** We say that a process P *preserves the secrecy* of $x \in \mathcal{V}ar(P)$ in the equational theory E, and we denote it by $P \models_{\mathsf{E}} \mathsf{Secret}(x)$, if whenever

$$(P, \emptyset, \emptyset) \overset{l_1,\ldots,l_n}{\rightarrow^*} (Q, \varphi, \sigma)$$

we have that $\varphi \not\vdash_{\mathsf{E}} x\sigma$.

Freshness of a variable with respect to a set of variables is another security property that we will demand of the two protocols in order to secure compose them. Intuitively, freshness of a variable means that protocols do not *reuse* one of the older secrets to instantiate this variable. The reuse by a protocol of some older secret in shared variable might create an equality unexpected by the second protocol and might lead to an insecure situation as exemplified in Section 6.3.3.

Definition 6.2 (freshness)**.** We say that a process P *guarantees the freshness* of $x \in \mathcal{V}ar(P)$ w.r.t. $\{y_1, \ldots, y_k\} \subseteq \mathcal{V}ar(P)$ in the equational theory E if whenever

$$(P, \emptyset, \emptyset) \xrightarrow{l_1,\ldots,l_n} (Q, \varphi, \sigma)$$

we have that $x\sigma \neq_{\mathsf{E}} y_i\sigma$ for all $1 \leq i \leq k$.

In the above definitions we assume that the variables in the processes have been conveniently α-renamed before application of the definition. If the above definitions concern variables appearing under replications, we assume that the conditions hold for any of the variables denoted by x (and resp. $y_1, \ldots, y_k$). This can be achieved formally by coloring all bound variables with different colors; whenever a variable is α-renamed it preserves its color. We would then say that a *certain color c is secret* when all variables colored with c remain secret. As this technicality is not essential in our approach, we prefer to use the less formal version.

6.3 Difficulties

Of course, two protocols P, Q cannot in general be (securely) composed in arbitrary ways. We illustrate several cases where composing two secure protocols yields an attack. For readability, in this section we use the notation $\{s\}_t$ for the term $\mathrm{enc}(s, t)$.

6.3.1 Revealing Shared Keys

If a protocol P is establishing a (secret) key k, then a protocol Q using the key k should not reveal it. This would clearly compromise the security of P but it could also compromise the security of Q. Assume for example that a protocol P_1 (playing the role of P above) generates two fresh (secret) data x and y and reveals the encryption of x under y:

$$P_1 = \nu x.\nu y.\mathbf{send}(\{x\}_y)$$

Note that P_1 may compute x and y in a more complicated way, but we consider just the rather trivial case where x and y are instantiated to fresh nonces for the sake of clarity. Then P_1 preserves the secrecy of both x and y. Assume now that a process Q_1 (playing the role of Q above) reveals y and uses x for encrypting a secret z:

$$Q_1 = \nu z.\mathbf{send}(y).\mathbf{send}(\{z\}_x).$$

Then $\nu x'.\nu y'.(x := x').(y := y').Q_1$ preserves the secrecy of z, while the composition $P_1.Q_1$ of both processes does not preserve the secrecy of z. Indeed

$$(P_1.Q_1, \emptyset, \emptyset) \rightarrow^* (0, \varphi, \sigma = \{x \mapsto k_1, y \mapsto k_2, z \mapsto k_3\})$$

for fresh private names $k_1, k_2, k_3 \in \mathcal{N}$, where $\varphi = \{w_1 \mapsto \{k_1\}_{k_2}, w_2 \mapsto k_2, w_3 \mapsto \{k_3\}_{k_1}\}$. We have $\varphi \vdash^{\mathrm{dec}(w_3,\mathrm{dec}(w_1,w_2))}_{\mathsf{E}_{\mathrm{enc}}} z\sigma$ and thus $P_1.Q_1 \not\models_{\mathsf{E}_{\mathrm{enc}}} \mathsf{Secret}(z)$, where $\mathsf{E}_{\mathrm{enc}} = \{\mathrm{dec}(\mathrm{enc}(x, y), y) = x\}$ models symmetric encryption.

Note that this attack works even if P_1 and Q_1 actually use different encryption symbols (thus even if they use disjoint signatures).

6.3.2 Sharing Primitives

The interaction of two protocols using common primitives may yield an attack, even if each of the protocols is secure when executed in isolation. Indeed, consider again the process P_1 described above and let Q_2 be a process that uses x for encrypting a secret z and outputs m for any m received under the encryption of y.

$$Q_2 = \nu z.\mathbf{send}(\{z\}_x).\mathbf{receive}(z').\mathbf{send}(\mathrm{dec}(z', y)).$$

Then $\nu x'.\nu y'.(x := x').(y := y').Q_2$ preserves the secrecy of z, while the composition $P_1.Q_2$ of both processes does not preserve the secrecy of z. Indeed

$$(P_1.Q_2, \emptyset, \emptyset) \xrightarrow{\mathbf{receive}(w_1)}{}^* (0, \varphi', \sigma \cup \{z' \mapsto \{k_1\}_{k_2}\})$$

where σ has been defined above and $\varphi' = \{w_1 \mapsto \{k_1\}_{k_2}, w_2 \mapsto \{k_3\}_{k_1}, w_3 \mapsto k_1\}$. We have $\varphi' \vdash^{\mathrm{dec}(w_2,w_3)}_{\mathsf{E}_{\mathrm{enc}}} z\sigma$, thus $P_1.Q_2 \not\models_{\mathsf{E}_{\mathrm{enc}}} \mathsf{Secret}(z)$.

So, in what follows, we will assume that the composed protocols use disjoint primitives. In Section 6.6, we extend our result to the case where the protocols may share some primitives such as encryption and hash, provided they are tagged.

6.3.3 Key Freshness

It is important that shared variables (that are assumed to be fresh) are indeed instantiated by fresh values. Assume for example that a protocol $R = R_1.R_2.R_3$ is composed of three phases:

- it first generates a fresh key x: let $R_1 = \nu x$;
- it then runs a sub-protocol R_2 to establish some secret y;
- it outputs a fresh value z if $x = y$: let $R_3 = \nu z.[x = y].\mathbf{send}(z)$.

Then R preserves the secrecy of z when $R_2 = \nu k.(y := k)$ chooses y to be a fresh key:

$$R_1.\nu k.(y := k).R_3 \models \mathsf{Secret}(z)$$

while it is not secure for all sub-protocols R_2 establishing a secret key y. Indeed, for $R_2' = (y := x)$, we have that $\nu k.(x := k).R_2'$ preserves the secrecy of the shared key y but, because y is not fresh,

$$R_1.R_2'.R_3 \not\models \mathsf{Secret}(z).$$

In what follows, we will see that the three situations that we have described in Section 6.3.1, Section 6.3.2 and respectively Section 6.3.3 are actually the only problematic cases and we will therefore require in our composition theorem that sub-protocols do not introduce such equalities.

6.4 Composing Disjoint Protocols

In our composition theorem, we show that under appropriate conditions, protocols that use disjoint cryptographic primitives compose well. We formalize the fact that two protocols use disjoint cryptographic primitives by ensuring that the protocols run over two distinct signatures, each equipped with its own equational theory.

6.4.1 Combination of Equational Theories

We will therefore consider that the signature $\mathcal{F}$ consists of the union of two disjoint signatures $\mathcal{F}_a$ and $\mathcal{F}_b$, i.e. $\mathcal{F} = \mathcal{F}_a \cup \mathcal{F}_b$ where $\mathcal{F}_a \cap \mathcal{F}_b = \emptyset$. We also consider an equational theory E_a over $\mathcal{F}_a$ and an equational theory E_b over $\mathcal{F}_b$. We define the equational theory $\mathsf{E} = \mathsf{E}_a \cup \mathsf{E}_b$ (over $\mathcal{F}$) to be the union of the equational theories E_a and E_b. Note that the relation $=_E$ (as defined in Chapter 2) is not in general the union of $=_{\mathsf{E}_a}$ and $=_{\mathsf{E}_b}$. We will assume w.l.o.g. that the two equational theories E_a and E_b are not trivial:

Definition 6.3. An equational theory E_0 is trivial if $s =_{\mathsf{E}_0} t$ for any terms s, t.

Trivial equational theories are not very interesting since any two terms are rendered equal by the theory. If one of E_a or E_b is trivial, then so is $\mathsf{E} = \mathsf{E}_a \cup \mathsf{E}_b$. Under a trivial equational theory, the intruder trivially knows all terms and implicitly all "secrets".

We also consider $\mathcal{N}_a \uplus \mathcal{N}_b \uplus \mathcal{N}_c = \mathcal{N}$ to be a partition of the set of private names and $\mathcal{M}_a \uplus \mathcal{M}_b \uplus \mathcal{M}_c = \mathcal{M}$ to be a partition of the set of public names. We let $D_a = \mathcal{F}_a \cup \mathcal{N}_a \cup \mathcal{M}_a$ be the *a-domain*, $D_b = \mathcal{F}_b \cup \mathcal{N}_b \cup \mathcal{M}_b$ be the *b-domain* and $D_c = \mathcal{M}_c \cup \mathcal{N}_c \cup \mathcal{W} \cup \mathcal{X}$ be the *c*-domain. Note that while the *a*-domain D_a and the *b*-domain D_b contain function symbols, the *c*-domain D_c does not. However, the *c* domain D_c contains all parameters $w_i \in \mathcal{W}$, while D_a and D_b do not contain any parameter. For symmetry with the other two domains, we define $\mathcal{F}_c$ to be the empty signature and $\mathsf{E}_c = \{\}$ to be the empty equational theory.

For every $d \in \{a, b, c\}$, we define the set of *pure d-terms* to be $T(\mathcal{F}_d, D_d \setminus \mathcal{F}_d \cup \mathcal{X})$. A term is *pure* if it is a pure *d*-term for some $d \in \{a, b, c\}$. We say that a process is over the *d*-domain (with $d \in \{a, b\}$) if all terms appearing in the process are pure *d*-terms.

6.4.2 Composition Theorem

Since a run of the protocol involves only a finite number of replications and determines the scheduling in parallel composition, we simply need to state our main result on *traces*, that is processes that contain neither parallel composition nor replication. In order to state our composition theorem in a general way, we simply need to show that any execution trace on two combined processes can be transformed into an execution trace on one of the two processes. Then a trace of the composition leading to an attack can be transformed into a trace of one of the individual protocols leading to an attack. The fact of stating the main composition theorem on traces is not a limitation, since our theorem can be used to securely compose processes with arbitrary replications and parallel compositions, as illustrated in Section 6.5.

We say that a process is *atomic* if it is one of $\mathbf{send}(t)$, $\mathbf{receive}(x)$, $[s = t]$, $x := t$, νx for some terms s, t and some variable x.

Let $P = P_1.\ldots.P_n$ be a trace over the *a*-domain with free variables $\{x_1, \ldots, x_p\}$ where P_i is an atomic process $(1 \leq i \leq n)$. Let $Q = Q_1.\ldots.Q_m$ be a trace over the *b*-domain with free variables $\{y_1, \ldots, y_q\}$ where Q_i is an atomic process $(1 \leq i \leq n)$. Intuitively, the free variables of Q are established by P and conversely, the free variables of P are established by Q.

Let $R = R_1.\ldots.R_{n+m}$ be a *ground interleaving* of $P_1, \ldots, P_n, Q_1, \ldots, Q_m$, that is $fv(R) = \emptyset$ and $\{R_1, \ldots, R_{n+m}\} = \{P_1, \ldots, P_n, Q_1, \ldots, Q_m\}$ (as multi-set equality). We consider R' to be a copy of R where the shared variables of P and Q are duplicated. More precisely, let $R' = R'_1.\ldots.R'_{n+m}$ be such that

1. $R'_i = P_j\{x \mapsto x^a\}$ if R_i is P_j for some j and where x ranges over all variables in P_j

2. $R'_i = Q_j\{x \mapsto x^b\}$ if R_i is Q_j for some j and where x ranges over all variables in Q_j

Example 6.2. We consider for the roles of P and Q the processes P_1 and Q_1 in Section 6.3.1:

$$\begin{array}{lll} P = \nu x.\nu y. & \mathbf{send}(\{x\}_y) & \\ Q = \quad \nu z. & & .\mathbf{send}(y).\mathbf{send}([z]_x), \end{array}$$

where $\{\cdot\}_\cdot$ and $[\cdot]_\cdot$ denote distinct encryption function symbols. We have that $fv(P) = \emptyset$ and $fv(Q) = \{x, y\}$. We also consider the following ground interleaving of P and Q:

$$R = \nu x.\nu y.\nu z.\mathbf{send}(\{x\}_y).\mathbf{send}(y).\mathbf{send}([z]_x).$$

We have then that

$$R' = \nu x^a.\nu y^a.\nu z^b.\mathbf{send}(\{x^a\}_{y^a}).\mathbf{send}(y^b).\mathbf{send}([z^b]_{x^b}).$$

We consider a run of the composition of P and Q.

$$(R, \emptyset, \emptyset) \xrightarrow{l_1,\ldots,l_k}{}^{*} (S_0, \varphi_0, \sigma_0) \xrightarrow{\S} (S, \varphi, \sigma) \tag{6.1}$$

where $\S$ is $\mathbf{receive}(r_{k+1})$ for some recipe r_{k+1} if the last action was an input and then empty string otherwise.

Assume w.l.o.g. that $x_1, \ldots, x_{p'}$ are the variables from $\{x_1, \ldots, x_p\}$ which appear in $\mathcal{D}om(\sigma_0)$ and that $y_1, \ldots, y_{q'}$ are the variables from $\{y_1, \ldots, y_q\}$ which appear in $\mathcal{D}om(\sigma_0)$. This means that $x_1, \ldots, x_{p'}, y_1, \ldots, y_{q'}$ are the shared variables that were instantiated before the last action of the execution.

Let $\{z_i\}_{1 \le i \le p'}$ and $\{z_i\}_{p+1 \le i \le p+q'}$ be fresh variables such that

$$\begin{array}{ll} z_i = z_j \text{ iff } x_i\sigma_0 =_{\mathsf{E}} y_j\sigma_0 & \text{for all } 1 \le i \le p' \text{ and all } p+1 \le j \le p+q' \\ z_i = z_j \text{ iff } x_i\sigma_0 =_{\mathsf{E}} x_j\sigma_0 & \text{for all } 1 \le i, j \le p' \\ z_i = z_j \text{ iff } y_i\sigma_0 =_{\mathsf{E}} y_j\sigma_0 & \text{for all } p+1 \le i, j \le p+q' \end{array}$$

and let

$$\begin{array}{rcl} R'' & = & \nu\{z_i\}_{1 \le i \le p', p+1 \le i \le p+q'}. \\ & & x_1^a := z_1. \ldots . x_{p'}^a := z_{p'}. \\ & & y_1^b := z_{p+1}. \ldots . y_{q'}^b := z_{p+q'}. \\ & & R'. \end{array}$$

Example 6.3. Continuing the previous example, we consider the run

$$\begin{array}{rcl} (R, \emptyset, \emptyset) \to^4 & (\mathbf{send}(y).\mathbf{send}([z]_x), \{w_1 \mapsto \{k_1\}_{k_2}\}, \{x \mapsto k_1, y \mapsto k_2, z \mapsto k_3\}) \\ \to & (\mathbf{send}([z]_x), \{w_1 \mapsto \{k_1\}_{k_2}, w_2 \mapsto k_2\}, \{x \mapsto k_1, y \mapsto k_2, z \mapsto k_3\}), \end{array}$$

where k_1, k_2, k_3 are fresh names, $\sigma_0 = \{x \mapsto k_1, y \mapsto k_2, z \mapsto k_3\}$ and $\varphi = \{w_1 \mapsto \{k_1\}_{k_2}, w_2 \mapsto k_2\}$. As $x\sigma_0 = k_1 \neq k_2 = y\sigma_0$, we obtain that

$$\begin{array}{rcl} R'' = & \nu\{z_1, z_2\}.x^b := z_1.y^b := z_2.R' \\ = & \nu\{z_1, z_2\}.x^b := z_1.y^b := z_2.\nu x^a.\nu y^a.\nu z^b.\mathbf{send}(\{x^a\}_{y^a}).\mathbf{send}(y^b).\mathbf{send}([z^b]_{x^b}). \end{array}$$

R'' corresponds to an interleaving of P and Q where P and Q do not share any variable anymore (since they are duplicated) and where the previously shared variables are instantiated by fresh distinct names. Note that whenever the execution of R instantiates two shared variables by the same value (e.g. $x_i\sigma_0 =_{\mathsf{E}} x_j\sigma_0$) then the (duplicated version of) x_i and x_j are instantiated in R'' by the same fresh name. This corresponds e.g. to the case where the same key is distributed among several participants (thus is assigned to distinct variables).

We are now ready to state our main theorem which says that we can mimic on R'' the execution trace of R (as in Equation 6.2) unless P or Q do not preserve the secrecy or the freshness of the shared variables.

Theorem 6.1. *Assume $\varphi_0 \not\vdash t\sigma$ for any $t \in \{x_1, \ldots, x_{p'}, y_1, \ldots, y_{q'}\}$ and that $x_i\sigma_0 \neq_{\mathsf{E}} y_j\sigma_0$ for all $1 \leq i \leq p', 1 \leq j \leq q'$. Then there exist S', φ', σ' such that*

$$(R'', \emptyset, \emptyset) \xrightarrow{l_1,\ldots,l_k,\S}{}^{*} (S', \varphi', \sigma') \tag{6.2}$$

and

1. *if r is a minimal recipe such that $\varphi \vdash^r x\sigma_0$ for some $x \in \{x_1, \ldots, x_{p'}, y_1, \ldots, y_{q'}\}$, then $\varphi' \vdash^r x^d\sigma'_0$ for some $d \in \{a, b\}$.*
2. *otherwise, if $\varphi \vdash^r x\sigma$ for $x \in \mathcal{V}ar(P) \cup \mathcal{V}ar(Q) \setminus \{x_1, \ldots, x_{p'}, y_1, \ldots, y_{q'}\}$ then $\varphi' \vdash^r x^d\sigma'$ for some $d \in \{a, b\}$.*

Furthermore, if $x, y \in \mathcal{D}om(\sigma)$ are such that $x\sigma =_{\mathsf{E}} y\sigma$ and $x^d, y^d \in \mathcal{D}om(\sigma')$ for some $d \in \{a, b\}$, then $x^d\sigma' =_{\mathsf{E}} y^d\sigma'$.

Intuitively, R'' is a composition of P and Q where the two processes do not share variables anymore. Theorem 6.1 says that we can mimic on R'' the execution trace of R. Furthermore, if R reveals a shared variable, R'' reveals some (duplicated) shared variable as well. It will be used in the next section to conclude that one of the two processes P or Q (executed alone) reveals one of its (shared) variables thus is not secure. Indeed, assume w.l.o.g. that R'' reveals the variable x_i^a. Since R'' corresponds to P executed in parallel (and independently) of Q, the process Q can be entirely simulated by the adversary; thus P reveals x_i, that is $P \not\models_E \mathsf{Secret}(x_i)$.

Example 6.4. Continuing our previous example, we had that

$$(R, \emptyset, \emptyset) \rightarrow^5 (\mathbf{send}([z]_x), \varphi, \sigma_0),$$

and that

$$\begin{aligned} R'' = &\ \nu\{z_1, z_2\}.x^b := z_1.y^b := z_2.R' \\ = &\ \nu\{z_1, z_2\}.x^b := z_1.y^b := z_2.\nu x^a.\nu y^a.\nu z^b.\mathbf{send}(\{x^a\}_{y^a}).\mathbf{send}(y^b).\mathbf{send}([z^b]_{x^b}). \end{aligned}$$

By Theorem 6.1, we obtain that there exist a process S', a frame φ' and a substitution σ' such that

$$(R'', \emptyset, \emptyset) \rightarrow^5 (S', \varphi', \sigma').$$

Let $r = w_2$ be a minimal for $y\sigma_0$ in the frame φ. By Item 1, we have that w_2 is recipe in φ' of $y^b\sigma'$.

The final remark in Theorem 6.1 (that $x\sigma =_{\mathsf{E}} y\sigma$ implies $x^d\sigma' =_{\mathsf{E}} y^d\sigma'$) is important to be able to completely separate the processes P and Q in R'': as the variable x_i^a $(1 \leq i \leq p')$ and y_i^b $(1 \leq i \leq q')$ are instantiated in the process R'' by fresh names such that x_i^a and y_j^b receive the same name if $x_i\sigma =_{\mathsf{E}} y_j\sigma$, it is important that such an equality does not happen. Otherwise, the two processes still share data and therefore we cannot conclude about either of them individually.

6.4.3 Proof of Theorem 6.1

This subsection consists of the proof of Theorem 6.1, which requires several non-trivial steps:

1. we define (Definition 6.5) a canonical form for each term that is called the *collapsed* form and which is obtained by replacing each term with an alien subterm to which it is equal, if such an alien subterm exists. An alien subterm is a subterm whose root symbol comes from a different signature than the root symbol of the term itself.

   The collapsed form of a term enjoys the property that each subterm has a canonical domain by avoiding spurious patterns such as $h^{-1}(h(\cdot))$ as shown in Example 6.5. Furthermore, in two collapsed terms which are equal modulo $=_{\mathsf{E}}$, alien subterms can be abstracted by names and the equality modulo $=_{\mathsf{E}}$ still holds (Lemma 6.6).

2. having terms in collapsed form, we can intuitively decide depending on the signature of the root symbol which of the two protocols created the term (even if subterms were created by the other protocol). Whenever a signature change occurs in a term, it means that either the subterm where the signature change occurs is the instantiation of one of the variables shared by the two protocols or that it was transmitted by the intruder to the protocol.

   Since our goal is to show that from the run of the composition, we can produce a run where the shared variables are duplicated and the free variables instantiated by fresh names, we define (Definition 6.4) a function $R^{\tilde{n}}_{d,\tilde{s}}$ which replaces the occurrences of the instantiations $\tilde{s}$ of the shared variables by the fresh names $\tilde{n}$ for the shared variables instantiated by the protocol from signature $d \in \{a, b\}$.

3. using the previously defined function $R^{\tilde{n}}_{d,\tilde{s}}$, we transform the *collapsed* run of the interleaving of the two traces into a run on the same interleaving, but where all shared variables are duplicated and where free variables are instantiated with fresh names.

   Using the properties of col and of $R^{\tilde{n}}_{d,\tilde{s}}$ proved in Lemma 6.7, Lemma 6.8, Lemma 6.9 and Lemma 6.10, we can prove that the new run *succeeds* (i.e. it obeys the operational semantics) and that if the initial run ended with an *attack*, the newly defined run has a similar *attack*.

Disjoint Equational Theories

Note that we have assumed that the two equational theories E_a and E_b are not trivial. Non-trivial equational theories enjoy the following property:

Lemma 6.1. *If E_0 is not trivial then $a \neq_{\mathsf{E}_0} b$ for any distinct atoms $a \neq b$.*

Proof. By contradiction, we assume $a \neq b$ but $a =_{\mathsf{E}_0} b$. We consider two arbitrary terms s and t and the replacement $\sigma = \{a \mapsto s, b \mapsto t\}$. As E_0 is stable by replacement of atoms by terms, we have that $s = a\sigma =_{\mathsf{E}_0} b\sigma = t$. Therefore $s =_{\mathsf{E}_0} t$ for arbitrary terms s and t. Therefore E_0 is trivial, which contradicts the hypothesis. Therefore our assumption was false and we have $a \neq_{\mathsf{E}_0} b$. □

If t is a term, we let $\mathsf{root}(t)$ denote the topmost symbol of the term:

$$\mathsf{root}(t) = \begin{cases} f & \text{if } t = f(t_1, \ldots, t_n) \\ a & \text{if } t = a \in \mathcal{A} \end{cases}$$

If t is a term, we define the *domain* of t to be the domain of its topmost symbol:

$$\mathsf{domain}(t) = d \in \{a, b, c\} \text{ iff } \mathsf{root}(t) \in D_d.$$

If $d \in \{a, b\}$, then a context $C \in T(\mathcal{F}_d, D_d \cup \{\_^1, \ldots, \_^k\})$ is called a *pure d-context*. If $C \neq \_^1$ is not the empty context, then we define $\mathsf{domain}(C)$ to be the domain of the topmost element in C.

We write $C[[s_1, \ldots, s_k]]$ for the term $C[s_1, \ldots, s_k]$ if C is a pure non-empty context and if $\mathsf{domain}(C) \neq \mathsf{domain}(s_i)$ for all $1 \leq i \leq k$. The terms $s_1, \ldots, s_k$ are then called the *alien subterms* of $C[s_1, \ldots, s_k]$. Note that in order to write $C[[s_1, \ldots, s_k]]$ for the term $C[s_1, \ldots, s_k]$, C cannot be the empty context. Furthermore, for each term t there is a unique (up to renaming of holes) context C such that $t = C[[s_1, \ldots, s_k]]$ for some terms $s_1, \ldots, s_k$.

Replacing Terms

In the proof of the theorem, we will be "abstracting away" the keys generated by a protocol and we will replace them with fresh names. This replacement is captured by the function $R^{\tilde{n}}_{d,\tilde{s}}$, defined below.

Let $e \in \{a, b\}$ be a domain, let $\tilde{s} = s_1, \ldots, s_k$ be terms, let $\tilde{n} = n^a_1, \ldots, n^a_k, n^b_1, \ldots, n^b_k \in \mathcal{N}$ be fresh private names such that $n^d_i = n^d_j$ iff $s_i =_{\mathsf{E}} s_j$ for all $d \in \{a, b\}, 1 \leq i, j \leq k$.

We define the function $R^{\tilde{n}}_{e,\tilde{s}}$ as follows:

Definition 6.4.

$$R^{\tilde{n}}_{e,\tilde{s}}(t) = \begin{cases} n^e_i & \text{if } \mathsf{root}(t) \notin D_e \text{ and } t =_{\mathsf{E}} s_i \text{ for some } 1 \leq i \leq k \\ f(R^{\tilde{n}}_{d,\tilde{s}}(t_1), \ldots, R^{\tilde{n}}_{d,\tilde{s}}(t_l)) & \text{otherwise, when } t = f(t_1, \ldots, t_l) \text{ with } f \in \mathcal{F}_d \\ t & \text{otherwise (when } t \in \mathcal{A} \text{ is an atom)} \end{cases}$$

The function $\mathsf{R}^{\tilde{n}}_{e,\tilde{s}}$ is designed to replace the terms $s_1, \ldots, s_k$ occurring at places where the signature changes from $\mathcal{F}_a$ to $\mathcal{F}_b$ (or vice-versa). The term s_i will be replaced by either n^a_i or by n^b_i, depending on what signature appears directly above s_i. The argument e is used to "initialize" the signature, i.e. define what signature is "above" the entire term.

We extend $\mathsf{R}^{\tilde{n}}_{e,\tilde{s}}$ to frames as expected:

$$R^{\tilde{n}}_{e,\tilde{s}}(\{w_1 \mapsto t_1, \ldots, w_n \mapsto t_n\}) = \{w_1 \mapsto R^{\tilde{n}}_{e,\tilde{s}}(t_1), \ldots, w_n \mapsto R^{\tilde{n}}_{e,\tilde{s}}(t_n)\}.$$

Collapsed Form

Since we do not make any assumption on E_a and E_b, there is no normal form. Following Chapter 9 of [14], we define a canonical form $\mathsf{col}(t)$ of a term t modulo E, called the *collapsed* form of t.

The idea of the collapsed form is to assign a canonical domain to the topmost symbol of a term. As shown in the following example, rewriting modulo E might change the signature of the topmost symbol, as the following examples shows:

> *Example* 6.5. Let $\mathsf{E}_a = \{h^{-1}(h(x)) = x\}$ and $\mathsf{E}_b = \{\mathrm{dec}(\mathrm{enc}(x, y), y) = x\}$. We have that $h^{-1}(h(\mathrm{enc}(s, k))) =_{\mathsf{E}} \mathrm{enc}(s, k)$. The topmost symbol $h^{-1} \in \mathcal{F}_a$ changed to $\mathrm{enc} \in \mathcal{F}_b$ during the rewrite step.
>
> The idea of the collapsed form is to simplify the term by removing spurious parts such as $h^{-1}(h(\cdot))$.

Formally, to obtain the collapsed form of a term, the term is (recursively) replaced with any of its alien subterms to which it is equal modulo E as follows,

Definition 6.5.

$$\mathsf{col}(t) = \begin{cases} t & \text{if } t \in \mathcal{A} \text{ is an atom} \\ s_i & \text{if } t = f(t_1, \ldots, t_l), \\ & \quad f(\mathsf{col}(t_1), \ldots, \mathsf{col}(t_l)) = C[[s_1, \ldots, s_k]] \text{ and} \\ & \quad C[[n_1, \ldots, n_k]] =_{\mathsf{E}_d} n_i \text{ where } n_1, \ldots, n_k \text{ are fresh names} \\ & \quad \text{such that } n_i = n_j \text{ iff } s_i =_{\mathsf{E}} s_j \text{ for all } 1 \leq i, j \leq l \\ & \quad \text{and } d = \mathsf{domain}(C) \\ f(\mathsf{col}(t_1), \ldots, \mathsf{col}(t_l)) & \text{if } t = f(t_1, \ldots, t_l) \text{ but the above condition does not hold.} \end{cases}$$

We can prove that the collapsed form of a term is equal modulo the equational theory to the term itself:

Lemma 6.2. *For any term t, we have that $\mathsf{col}(t) =_{\mathsf{E}} t$.*

Proof. By induction on the size of t.

1. For the base case, if t is an atom, then $\mathsf{col}(t) = t$ by the definition of col and therefore $\mathsf{col}(t) =_{\mathsf{E}} t$.
2. For the inductive case, if $t = f(t_1, \ldots, t_l)$, we have that $t_1 =_{\mathsf{E}} \mathsf{col}(t_1), \ldots, t_l =_{\mathsf{E}} \mathsf{col}(t_l))$ by the induction hypothesis. Therefore $t = f(t_1, \ldots, t_l) =_{\mathsf{E}} f(\mathsf{col}(t_1)), \ldots, \mathsf{col}(t_l))$.

   We distinguish two cases:

   (a) if $\mathsf{col}(t) = f(\mathsf{col}(t_1), \ldots, \mathsf{col}(t_l))$ (i.e. the third case of the definition of col) we have already shown that $t =_{\mathsf{E}} \mathsf{col}(t)$.

(b) otherwise, $f(\mathsf{col}(t_1),\ldots,\mathsf{col}(t_l)) = C[[s_1,\ldots,s_k]]$ for some context C and some terms $s_1,\ldots,s_k$ and there exists an index $1 \leq i \leq k$ such that $C[n_1,\ldots,n_k] =_{\mathsf{E}_d} n_i$ for some fresh names $n_1,\ldots,n_k$ such that $n_x = n_y$ iff $s_x =_{\mathsf{E}} s_y$ for all $1 \leq x,y \leq k$ and $\mathsf{col}(t) = s_i$, where $d = \mathsf{domain}(C)$. As $C[n_1,\ldots,n_k] =_{\mathsf{E}_d} n_i$ and $\mathsf{E}_d \subseteq \mathsf{E}$, it follows that $C[n_1,\ldots,n_k] =_{\mathsf{E}} n_i$.

For all $1 \leq x \leq k$, let $[x]$ denote the smallest index $1 \leq y \leq k$ such that $n_x = n_y$. As the equational theory E is stable by replacement of names by arbitrary terms, we have that $C[n_1,\ldots,n_k] =_{\mathsf{E}} n_i$ implies $C[n_1,\ldots,n_k]\{n_1 \mapsto s_{[1]},\ldots,n_k \mapsto s_{[k]}\} =_{\mathsf{E}} n_i\{n_1 \mapsto s_{[1]},\ldots,n_k \mapsto s_{[k]}\}$, which is equivalent to $C[s_{[1]},\ldots,s_{[k]}] =_{\mathsf{E}} s_{[i]}$. But by choice of the names $n_1,\ldots,n_k$ and the definition of $[x]$, we have that $s_{[x]} =_{\mathsf{E}} s_x$ for all $1 \leq x \leq k$. Therefore we obtain $C[s_1,\ldots,s_k] =_{\mathsf{E}} s_i$.

We have that $t =_{\mathsf{E}} f(\mathsf{col}(t_1),\ldots,\mathsf{col}(t_l))$, that $f(\mathsf{col}(t_1),\ldots,\mathsf{col}(t_l)) = C[[s_1,\ldots,s_k]]$, that $C[s_1,\ldots,s_k] =_{\mathsf{E}} s_i$ and that $\mathsf{col}(t) = s_i$ for some index $1 \leq i \leq k$.

By transitivity of $=_{\mathsf{E}}$, we immediately obtain that $t =_{\mathsf{E}} \mathsf{col}(t)$.

□

We also have that the size of term does not increase by collapsing.

Lemma 6.3. *For any term t, $|\mathsf{col}(t)| \leq |t|$.*

Proof. By well-founded induction on the size of t. We distinguish among three cases:

1. if $t \in \mathcal{A}$, then $\mathsf{col}(t) = t$ and therefore $|\mathsf{col}(t)| = |t|$.
2. if $t = f(t_1,\ldots,t_l)$ and $\mathsf{col}(t) = f(\mathsf{col}(t_1),\ldots,\mathsf{col}(t_l))$ we have that $|\mathsf{col}(t_i)| \leq |t_i|$ for all $1 \leq i \leq l$ by the induction hypothesis and therefore
$$|\mathsf{col}(t)| = 1 + \sum_{1\leq i\leq l} |\mathsf{col}(t_i)| \leq 1 + \sum_{1\leq i\leq l} |t_i| = |t|.$$
3. otherwise $t = f(t_1,\ldots,t_l)$, $f(\mathsf{col}(t_1),\ldots,\mathsf{col}(t_l)) = C[[s_1,\ldots,s_k]]$, $C[[n_1,\ldots,n_k]] =_{\mathsf{E}_d} n_m$ where $n_1,\ldots,n_k$ are fresh names such that $n_i = n_j$ iff $s_i =_{\mathsf{E}} s_j$ for all $1 \leq i,j \leq l$, $\mathsf{col}(t) = s_m$ and $c = \mathsf{domain}(C)$.

   By the definition of $C[[\ldots]]$, C cannot be the trivial context and therefore s_m is a subterm of $\mathsf{col}(t_x)$ for some $t_x \in \{t_1,\ldots,t_l\}$. Therefore $|s_m| \leq |\mathsf{col}(t_x)|$. By the induction hypothesis, we also have that $|\mathsf{col}(t_x)| \leq |t_x|$ and, as t_x is a subterm of t, we have that $|t_x| \leq |t|$. By transitivity, we obtain that $|s_m| \leq |t|$ or equivalently $|\mathsf{col}(t)| \leq |t|$.

□

We also have that any subterm of a collapsed term is collapsed:

Lemma 6.4. *Any subterm $t \in \mathsf{st}(\mathsf{col}(s))$ of $\mathsf{col}(s)$ is such that $t = \mathsf{col}(t)$.*

Proof. By induction on the size of s. We distinguish among the following cases:

1. if s is an atom, then $\mathsf{col}(s) = s \in \mathcal{A}$ and its only subterm $t \in \mathsf{st}(\mathsf{col}(s))$ is $t = \mathsf{col}(s)$. Therefore $t = s$ and $\mathsf{col}(t) = t$.
2. if $s = f(t_1,\ldots,t_l)$ and $\mathsf{col}(s) = f(\mathsf{col}(t_1),\ldots,\mathsf{col}(t_l))$, then any subterm $t \in \mathsf{st}(\mathsf{col}(s))$ is either:
   (a) a subterm $t \in \mathsf{st}(\mathsf{col}(t_i))$ of the term $\mathsf{col}(t_i)$ for some i, in which case $t = \mathsf{col}(t)$ by the induction hypothesis,
   (b) or $t = \mathsf{col}(s)$ is the entire term, in which case $\mathsf{col}(t) = \mathsf{col}(f(\mathsf{col}(t_1),\ldots,\mathsf{col}(t_l))$ and it follows by the definition of col that $\mathsf{col}(t) = t$.

3. otherwise it must be that $s = f(t_1, \ldots, t_l)$, $f(\mathsf{col}(t_1), \ldots, \mathsf{col}(t_l)) = C[[s_1, \ldots, s_k]]$, $\mathsf{col}(s) = s_x$ and $C[n_1, \ldots, n_k] =_{\mathsf{E}_d} n_x$ for some $x \in \{1, \ldots, k\}$ where $n_1, \ldots, n_k$ are fresh names such that $n_i = n_j$ iff $s_i =_{\mathsf{E}} s_j$ for all $1 \leq i, j \leq k$ and $c = \mathsf{domain}(C)$.

   As C is not the empty context, we have that s_x is a subterm of $\mathsf{col}(t_y)$ for some y. We conclude by the induction hypothesis, since any subterm t of $\mathsf{col}(t_y)$ (and therefore any subterm t of $s_x = \mathsf{col}(s)$) is such that $t = \mathsf{col}(t)$.

□

We also trivially have that first collapsing the direct subterms does not change the final result of collapse.

Lemma 6.5. *For any function symbol $f \in \mathcal{F}_d$ and for any terms $s_1, \ldots, s_k$ we have that*

$$\mathsf{col}(f(s_1, \ldots, s_k)) = \mathsf{col}(f(\mathsf{col}(s_1), \ldots, \mathsf{col}(s_k)).$$

Proof. Let $C, t_1, \ldots, t_l$ be such that $f(\mathsf{col}(s_1), \ldots, \mathsf{col}(s_k)) = C[[t_1, \ldots, t_l]]$. By Lemma 6.4 we have that $\mathsf{col}(\mathsf{col}(s_i)) = \mathsf{col}(s_i)$ for all $1 \leq i \leq k$ and therefore $f(\mathsf{col}(\mathsf{col}(s_1)), \ldots, \mathsf{col}(\mathsf{col}(s_k))) = f(\mathsf{col}(s_1), \ldots, \mathsf{col}(s_k)) = C[[t_1, \ldots, t_l]]$.

Let $n_1, \ldots, n_l$ be fresh names such that $n_x = n_y$ iff $t_x =_{\mathsf{E}} t_y$ for all $1 \leq x, y \leq l$. We distinguish two cases:

1. if $C[n_1, \ldots, n_l] =_{\mathsf{E}_d} n_x$ for some $1 \leq x \leq l$, we have that

$$\mathsf{col}(f(s_1, \ldots, s_k)) = \mathsf{col}(f(\mathsf{col}(s_1), \ldots, \mathsf{col}(s_k))) = s_x.$$

2. if $C[n_1, \ldots, n_l] \neq_{\mathsf{E}_d} n_x$ for any $1 \leq x \leq l$, we have that

$$\mathsf{col}(f(s_1, \ldots, s_k)) = \mathsf{col}(f(\mathsf{col}(s_1), \ldots, \mathsf{col}(s_k))) = f(\mathsf{col}(s_1), \ldots, \mathsf{col}(s_k)).$$

□

Furthermore, two terms which are equal modulo E have collapsed forms that enjoy the following property:

Lemma 6.6 (Fundamental Collapse Lemma)**.** *If $s =_{\mathsf{E}} t$, then $\mathsf{col}(s) = C[[s_1, \ldots, s_k]]$, $\mathsf{col}(t) = D[[s_{k+1}, \ldots, s_{k+l}]]$ such that $\mathsf{domain}(C) = \mathsf{domain}(D)$ and $C[n_1, \ldots, n_k] =_{\mathsf{E}_d} D[n_{k+1}, \ldots, n_{k+l}]$ where $d = \mathsf{domain}(C)$ and $n_1, \ldots, n_{k+l}$ are fresh names such that $n_i = n_j$ iff $s_i =_{\mathsf{E}} s_j$ for all $1 \leq i, j \leq k + l$.*

The full proof of Lemma 6.6 can be found in Appendix A.1.

We extend col to frames as expected: if $\varphi = \{w_1 \mapsto t_1, \ldots, w_n \mapsto t_n\}$, then $\mathsf{col}(\varphi) = \{w_1 \mapsto \mathsf{col}(t_1), \ldots, w_n \mapsto \mathsf{col}(t_n)\}$.

We next show that $R^{\tilde{n}}_{d,\tilde{s}}$ preserves equality modulo E when applied on collapsed terms.

Lemma 6.7. *If $s =_{\mathsf{E}} t$, then $R^{\tilde{n}}_{d,\tilde{s}}(\mathsf{col}(s)) =_{\mathsf{E}} R^{\tilde{n}}_{d,\tilde{s}}(\mathsf{col}(t))$ for any $d \in \{a, b\}$.*

Proof. By well-founded induction on $max(|s|, |t|)$. As $s =_{\mathsf{E}} t$, we have by Lemma 6.6 that

1. either $\mathsf{col}(s) = \mathsf{col}(t) \in \mathcal{A}$ and in this case we can immediately conclude: $\mathsf{col}(s) = \mathsf{col}(t)$ implies $R^{\tilde{n}}_{d,\tilde{s}}(\mathsf{col}(s)) = R^{\tilde{n}}_{d,\tilde{s}}(\mathsf{col}(t))$ which implies $R^{\tilde{n}}_{d,\tilde{s}}(\mathsf{col}(s)) =_{\mathsf{E}} R^{\tilde{n}}_{d,\tilde{s}}(\mathsf{col}(t))$,

2. or there exists $c \in \{a, b\}$ and two pure c-contexts C, D such that

$$\mathsf{col}(s) = C[[t_1, \ldots, t_k]]$$
$$\mathsf{col}(t) = D[[t_{k+1}, \ldots, t_{k+l}]]$$

   and such that $C[[n_1, \ldots, n_k]] =_{\mathsf{E}_c} D[[n_{k+1}, \ldots, n_{k+l}]]$ where $n_1, \ldots, n_{k+l}$ are fresh names such that $n_i = n_j$ iff $t_i =_{\mathsf{E}} t_j$ for all $1 \leq i, j \leq k + l$.

   We further distinguish two cases:

(a) if $c \neq d$ and there exists $s_i \in \tilde{s}$ such that $s =_{\mathsf{E}} t =_{\mathsf{E}} s_i \in \tilde{s}$, we have by the definition of $R^{\tilde{n}}_{d,\tilde{s}}$ that $R^{\tilde{n}}_{d,\tilde{s}}(\mathsf{col}(s)) = n^d_i = R^{\tilde{n}}_{d,\tilde{s}}(\mathsf{col}(t))$ and we are done;

(b) otherwise, $R^{\tilde{n}}_{d,\tilde{s}}(\mathsf{col}(s)) = R^{\tilde{n}}_{d,\tilde{s}}(C[[t_1,\ldots,t_k]]) = C[R^{\tilde{n}}_{c,\tilde{s}}(t_1),\ldots,R^{\tilde{n}}_{c,\tilde{s}}(t_k)]$ (by the definition of $R^{\tilde{n}}_{d,\tilde{s}}$) and similarly $R^{\tilde{n}}_{d,\tilde{s}}(\mathsf{col}(t)) = D[R^{\tilde{n}}_{c,\tilde{s}}(t_{k+1}),\ldots,R^{\tilde{n}}_{c,\tilde{s}}(t_{k+l})]$.
By the definition of $C[[t_1,\ldots,t_k]]$ and $D[[t_{k+1},\ldots,t_{k+l}]]$, we have that neither C nor D are the empty context. Therefore t_i is a strict subterm of $\mathsf{col}(s) = C[[t_1,\ldots,t_k]]$ for all $1 \leq i \leq k$ and that t_{k+i} is a strict subterm of $\mathsf{col}(t) = D[[t_{k+1},\ldots,t_{k+l}]]$ for all $1 \leq i \leq l$.
Therefore $|t_i| < |\mathsf{col}(s)|$ for all $1 \leq i \leq k$ and $|t_{k+i}| < |\mathsf{col}(t)|$ for all $1 \leq i \leq l$. By Lemma 6.3, we have that $|\mathsf{col}(s)| \leq |s|$ and $|\mathsf{col}(t)| \leq |t|$. Therefore $|t_i| < |s|$ for all $1 \leq i \leq k$ and $|t_{k+i}| < |t|$ for all $1 \leq i \leq l$.
We can therefore apply the induction hypothesis on every pair t_i, t_j $1 \leq i,j \leq k+l$ to obtain that if $t_i =_{\mathsf{E}} t_j$ then $R^{\tilde{n}}_{c,\tilde{s}}(\mathsf{col}(t_i)) =_{\mathsf{E}} R^{\tilde{n}}_{c,\tilde{s}}(\mathsf{col}(t_j))$. Furthermore, by Lemma 6.4, we have that $\mathsf{col}(t_i) = t_i$ for all $1 \leq i \leq k+l$. Therefore $t_i =_{\mathsf{E}} t_j$ implies $R^{\tilde{n}}_{c,\tilde{s}}(t_i) =_{\mathsf{E}} R^{\tilde{n}}_{c,\tilde{s}}(t_j)$ for all $1 \leq i,j \leq k+l$.
Let $[i]$ denote the smallest index j such that $R^{\tilde{n}}_{c,\tilde{s}}(t_i)$ and $R^{\tilde{n}}_{c,\tilde{s}}(t_j)$ are equal modulo E: $[i] = min\{j \mid R^{\tilde{n}}_{c,\tilde{s}}(t_j) =_{\mathsf{E}} R^{\tilde{n}}_{c,\tilde{s}}(t_j)\}$.
As $C[[n_1,\ldots,n_k]] =_{\mathsf{E}_c} D[[n_{k+1},\ldots,n_{k+l}]]$, we have by enlarging the equational theory that $C[[n_1,\ldots,n_k]] =_{\mathsf{E}} D[[n_{k+1},\ldots,n_{k+l}]]$ and, as E is stable by replacement of names with arbitrary terms, we obtain

$$C[[n_1,\ldots,n_k]]\sigma =_{\mathsf{E}} D[[n_{k+1},\ldots,n_{k+l}]]\sigma$$

where $\sigma = \{n_i \mapsto R^{\tilde{n}}_{c,\tilde{s}}(t_{[i]})\}_{1\leq i\leq k+l}$.
As $R^{\tilde{n}}_{c,\tilde{s}}(t_i) =_{\mathsf{E}} R^{\tilde{n}}_{c,\tilde{s}}(t_{[i]})$ by the definition of $[i]$ for all $1 \leq i \leq k+l$, we have that

$$C[[R^{\tilde{n}}_{c,\tilde{s}}(t_1),\ldots,R^{\tilde{n}}_{c,\tilde{s}}(t_k)]] =_{\mathsf{E}} D[[R^{\tilde{n}}_{c,\tilde{s}}(t_{k+1}),\ldots,R^{\tilde{n}}_{c,\tilde{s}}(t_{k+l})]]$$

as well. Equivalently, $R^{\tilde{n}}_{d,\tilde{s}}(\mathsf{col}(s)) =_{\mathsf{E}} R^{\tilde{n}}_{d,\tilde{s}}(\mathsf{col}(t))$, which is what we had to prove.

□

We also need the following easy technical result.

Lemma 6.8. *Let t be a term such that $t \neq_{\mathsf{E}} s_i$ for any $s_i \in \tilde{s}$. Then $R^{\tilde{n}}_{a,\tilde{s}}(t) = R^{\tilde{n}}_{b,\tilde{s}}(t)$.*

Proof. We distinguish among the three cases of the definition of $R^{\tilde{n}}_{c,\tilde{s}}$ (with $c \in \{a,b\}$). As $t \neq_{\mathsf{E}} s_i$ for any $s_i \in \tilde{s}$, we cannot be in the first cases.
If $t = f(t_1,\ldots,t_l)$, we have that $R^{\tilde{n}}_{a,\tilde{s}}(t) = f(R^{\tilde{n}}_{d,\tilde{s}}(t_1),\ldots,R^{\tilde{n}}_{d,\tilde{s}}(t_l)) = R^{\tilde{n}}_{b,\tilde{s}}(t)$ where $d \in \{a,b\}$ is such that $f \in \mathcal{F}_d$.
Otherwise, $R^{\tilde{n}}_{a,\tilde{s}}(t) = t = R^{\tilde{n}}_{b,\tilde{s}}(t)$.

□

Lemma 6.9. *Let $\tilde{s} = s_1,\ldots,s_n$ be terms, let φ be a frame and let r be a recipe over φ such that $\varphi \not\vdash^{r'}_{\mathsf{E}} s_i$ for any $1 \leq i \leq n$ and for any subrecipe $r' \sqsubseteq r$ of r. Then for any $c \in \{a,b\}$ we have that*

$$rR^{\tilde{n}}_{c,\tilde{s}}(\mathsf{col}(\varphi)) =_{\mathsf{E}} R^{\tilde{n}}_{c,\tilde{s}}(\mathsf{col}(r\varphi)).$$

Proof. We make the proof by induction on the size of r.

1. if $r = w_i \in \mathcal{D}om(\varphi)$, we immediately have that

$$rR^{\tilde{n}}_{c,\tilde{s}}(\mathsf{col}(\varphi)) = w_iR^{\tilde{n}}_{c,\tilde{s}}(\mathsf{col}(\varphi)) = R^{\tilde{n}}_{c,\tilde{s}}(\mathsf{col}(w_i\varphi)) = R^{\tilde{n}}_{c,\tilde{s}}(\mathsf{col}(r\varphi)).$$

2. otherwise, if $r = a \in \mathcal{A}$ is an atom other than a parameter $w_i \in \mathcal{D}om(\varphi)$, we have that $a \notin \tilde{s}$ (because $\varphi \vdash^r a$). Therefore $R^{\tilde{n}}_{c,\tilde{s}}(\mathsf{col}(a)) = R^{\tilde{n}}_{c,\tilde{s}}(a) = a$.

   We have $rR^{\tilde{n}}_{c,\tilde{s}}(\mathsf{col}(\varphi)) = aR^{\tilde{n}}_{c,\tilde{s}}(\mathsf{col}(\varphi)) = a = R^{\tilde{n}}_{c,\tilde{s}}(\mathsf{col}(a)) = R^{\tilde{n}}_{c,\tilde{s}}(\mathsf{col}(a\varphi)) = R^{\tilde{n}}_{c,\tilde{s}}(\mathsf{col}(r\varphi))$.

3. otherwise, $r = f(r_1, \ldots, r_k)$ for some recipes $r_1, \ldots, r_k$. By the definition of col, we have that:

   (a) either $\mathsf{col}(r\varphi) = f(\mathsf{col}(r_1\varphi), \ldots, \mathsf{col}(r_k\varphi))$, in which case by the definition of $R^{\tilde{n}}_{c,\tilde{s}}$ and because $r\varphi \neq_{\mathsf{E}} s_i$ for any $1 \leq i \leq n$, we obtain

   $$R^{\tilde{n}}_{c,\tilde{s}}(\mathsf{col}(r\varphi)) = f(R^{\tilde{n}}_{d,\tilde{s}}(\mathsf{col}(r_1\varphi)), \ldots, R^{\tilde{n}}_{d,\tilde{s}}(\mathsf{col}(r_k\varphi))),$$

   where $d \in \{a, b\}$ is such that $f \in D_d$.

   By the induction hypothesis, we have that $r_i R^{\tilde{n}}_{d,\tilde{s}}(\mathsf{col}(\varphi)) =_{\mathsf{E}} R^{\tilde{n}}_{d,\tilde{s}}(\mathsf{col}(r_i\varphi))$ for all $1 \leq i \leq l$.

   Then $R^{\tilde{n}}_{c,\tilde{s}}(\mathsf{col}(r\varphi)) =_{\mathsf{E}} f(r_1 R^{\tilde{n}}_{d,\tilde{s}}(\mathsf{col}(\varphi)), \ldots, r_k R^{\tilde{n}}_{d,\tilde{s}}(\mathsf{col}(\varphi))) = f(r_1, \ldots, r_k) R^{\tilde{n}}_{d,\tilde{s}}(\mathsf{col}(\varphi))$, which is equal, by Lemma 6.8, to $rR^{\tilde{n}}_{c,\tilde{s}}(\mathsf{col}(\varphi))$.

   (b) or $f(\mathsf{col}(r_1\varphi), \ldots, \mathsf{col}(r_k\varphi)) = C[[t_1, \ldots, t_m]]$, $C[n_1, \ldots, n_m] =_{\mathsf{E}_d} n_x$ for some $1 \leq x \leq m$ where $n_1, \ldots, n_m$ are fresh names such that $n_i = n_j$ iff $t_i =_{\mathsf{E}} t_j$ (for all $1 \leq i \leq m$) where $d \in \{a, b\}$ is such that $f \in \mathcal{F}_d$, and $\mathsf{col}(r\varphi) = t_x$. As $C[n_1, \ldots, n_m] =_{\mathsf{E}_d} n_x$ and $\mathsf{E}_d \subseteq \mathsf{E}$, we have that $C[n_1, \ldots, n_m] =_{\mathsf{E}} n_x$.

   Note that, as C is not the empty context, we have that for any $1 \leq i \leq m$, t_i is a subterm of some $\mathsf{col}(r_j\varphi)$ $(1 \leq j \leq k)$. Therefore, by Lemma 6.4, we have that $\mathsf{col}(t_i) = t_i$ for all $1 \leq i \leq m$.

   We have that the right-hand side of the equality we are trying to prove is $R^{\tilde{n}}_{c,\tilde{s}}(\mathsf{col}(r\varphi)) = R^{\tilde{n}}_{c,\tilde{s}}(t_x)$.

   The left-hand side (from hereon, lhs) of the equality is

   $$r(R^{\tilde{n}}_{c,\tilde{s}}(\mathsf{col}(\varphi))) = f(r_1, \ldots, r_k) R^{\tilde{n}}_{c,\tilde{s}}(\mathsf{col}(\varphi)) = f(r_1 R^{\tilde{n}}_{d,\tilde{s}}(\mathsf{col}(\varphi)), \ldots, r_k R^{\tilde{n}}_{d,\tilde{s}}(\mathsf{col}(\varphi)))$$

   where $d \in \{a, b\}$ is such that $f \in \mathcal{F}_d$. By the induction hypothesis, $r_i R^{\tilde{n}}_{d,\tilde{s}}(\mathsf{col}(\varphi)) =_{\mathsf{E}} R^{\tilde{n}}_{d,\tilde{s}}(\mathsf{col}(r_i\varphi))$ for all $1 \leq i \leq l$.

   Therefore the lhs is $r(R^{\tilde{n}}_{c,\tilde{s}}(\mathsf{col}(\varphi))) =_{\mathsf{E}} f(R^{\tilde{n}}_{d,\tilde{s}}(\mathsf{col}(r_1\varphi)), \ldots, R^{\tilde{n}}_{d,\tilde{s}}(\mathsf{col}(r_k\varphi)))$, which, as $f(\mathsf{col}(r_1\varphi), \ldots, \mathsf{col}(r_k\varphi)) =_{\mathsf{E}} r\varphi \neq_{\mathsf{E}} s_i$ for any $1 \leq i \leq n$, is equal by the definition of $R^{\tilde{n}}_{c,\tilde{s}}$, to $R^{\tilde{n}}_{c,\tilde{s}}(f(\mathsf{col}(r_1\varphi), \ldots, \mathsf{col}(r_k\varphi)))$. But by assumption $f(\mathsf{col}(r_1\varphi), \ldots, \mathsf{col}(r_k\varphi)) = C[[t_1, \ldots, t_m]]$. As C starts with $f \in \mathcal{F}_d$, we obtain

   $$R^{\tilde{n}}_{c,\tilde{s}}(f(\mathsf{col}(r_1\varphi), \ldots, \mathsf{col}(r_k\varphi))) = R^{\tilde{n}}_{c,\tilde{s}}(C[t_1, \ldots, t_m]) = C[R^{\tilde{n}}_{d,\tilde{s}}(t_1), \ldots, R^{\tilde{n}}_{d,\tilde{s}}(t_m)].$$

   We have shown that the lhs of the equality is $r(R^{\tilde{n}}_{c,\tilde{s}}(\mathsf{col}(\varphi)) =_{\mathsf{E}} C[R^{\tilde{n}}_{d,\tilde{s}}(t_1), \ldots, R^{\tilde{n}}_{d,\tilde{s}}(t_m)]$. But $C[n_1, \ldots, n_m] =_{\mathsf{E}} n_x$. As $n_i = n_j$ implies $t_i =_{\mathsf{E}} t_j$ (by choice of n_i) and $t_i =_{\mathsf{E}} t_j$ implies $R^{\tilde{n}}_{d,\tilde{s}}(\mathsf{col}(t_i)) =_{\mathsf{E}} R^{\tilde{n}}_{d,\tilde{s}}(\mathsf{col}(t_j))$ by Lemma 6.7. But $t_i = \mathsf{col}(t_i)$ for all $1 \leq i \leq m$ and therefore $n_i = n_j$ implies $R^{\tilde{n}}_{d,\tilde{s}}(t_i) =_{\mathsf{E}} R^{\tilde{n}}_{d,\tilde{s}}(t_j)$ for all $1 \leq i, j \leq m$. Therefore, as $C[n_1, \ldots, n_m] =_{\mathsf{E}} n_x$ and as the equational theory E is stable by replacement of names by arbitrary terms, we obtain that $C[R^{\tilde{n}}_{d,\tilde{s}}(t_1), \ldots, R^{\tilde{n}}_{d,\tilde{s}}(t_m)] =_{\mathsf{E}} R^{\tilde{n}}_{c,\tilde{s}}(t_x)$. Therefore the lhs of the equality is $r(R^{\tilde{n}}_{c,\tilde{s}}(\mathsf{col}(\varphi))) =_{\mathsf{E}} R^{\tilde{n}}_{c,\tilde{s}}(t_x)$.

   We have shown that the right-hand side of the equality is $R^{\tilde{n}}_{c,\tilde{s}}(\mathsf{col}(r\varphi)) = R^{\tilde{n}}_{c,\tilde{s}}(t_x)$ and that the left-hand side of the equality is $r(R^{\tilde{n}}_{c,\tilde{s}}(\mathsf{col}(\varphi))) =_{\mathsf{E}} R^{\tilde{n}}_{c,\tilde{s}}(t_x)$. As both the rhs and the lhs are equal modulo E to the same term, we conclude by transitivity that $r(R^{\tilde{n}}_{c,\tilde{s}}(\mathsf{col}(\varphi))) =_{\mathsf{E}} R^{\tilde{n}}_{c,\tilde{s}}(\mathsf{col}(r\varphi))$.

□

Lemma 6.10. *Let $c \in \{a, b\}$ be a domain, let φ be a frame, let $s_1, \ldots, s_k$ be ground terms and let $r_1, \ldots, r_l$ be recipes over φ such that $\varphi \not\vdash^{r'_i}_{\mathsf{E}} s_i$ for any $1 \leq i \leq k$ and for any subrecipe $r'_i \sqsubseteq r_i$ of r_i. Let $\sigma = \{x_i \mapsto s_i\}_{1 \leq i \leq k}$ and let $\tau = \{y_i \mapsto r_i\varphi\}_{1 \leq i \leq l}$. If s is a pure c-term, then*

$$R^{\tilde{n}}_{c,\tilde{s}}(\mathsf{col}(s\sigma\tau)) =_{\mathsf{E}} s\sigma'\tau',$$

where $\sigma' = \{x_i \mapsto R^{\tilde{n}}_{c,\tilde{s}}(\mathsf{col}(s_i))\}_{1 \leq i \leq k}$ and $\tau' = \{y_i \mapsto R^{\tilde{n}}_{c,\tilde{s}}(\mathsf{col}(r_i\varphi))\}_{1 \leq i \leq l}$.

Proof. By induction on the size of s:

1. if $s = x_i \in \mathcal{D}om(\sigma)$, then $s\sigma\tau = s_i$ and $R^{\tilde{n}}_{c,\tilde{s}}(\mathsf{col}(s\sigma\tau)) = R^{\tilde{n}}_{c,\tilde{s}}(\mathsf{col}(s_i))$.

   Similarly, $s\sigma'\tau' = \sigma'(x_i) = R^{\tilde{n}}_{c,\tilde{s}}(\mathsf{col}(s_i))$ and therefore $R^{\tilde{n}}_{c,\tilde{s}}(\mathsf{col}(s\sigma\tau)) = s\sigma'\tau'$.

2. if $s = y_i \in \mathcal{D}om(\tau)$, then $s\sigma\tau = r_i\varphi$ and $R^{\tilde{n}}_{c,\tilde{s}}(\mathsf{col}(s\sigma\tau)) = R^{\tilde{n}}_{c,\tilde{s}}(\mathsf{col}(r_i\varphi))$.

   But $s\sigma'\tau' = y_i\sigma'\tau' = y_i\tau' = r_i R^{\tilde{n}}_{c,\tilde{s}}(\mathsf{col}(\varphi))$. By Lemma 6.9, $r_i R^{\tilde{n}}_{c,\tilde{s}}(\mathsf{col}(\varphi)) =_{\mathsf{E}} R^{\tilde{n}}_{c,\tilde{s}}(\mathsf{col}(r_i\varphi))$. Therefore $s\sigma'\tau' =_{\mathsf{E}} R^{\tilde{n}}_{c,\tilde{s}}(\mathsf{col}(r_i\varphi))$.

   Therefore $R^{\tilde{n}}_{c,\tilde{s}}(\mathsf{col}(s\sigma\tau)) =_{\mathsf{E}} s\sigma'\tau'$.

3. if $s = a \in \mathcal{A} \setminus (\mathcal{D}om(\sigma) \cup \mathcal{D}om(\tau))$ is an atom not in the domains of σ and τ, we have that $s\sigma\tau = a$ and $s\sigma'\tau' = a$.

   By the definition of col, we have that $\mathsf{col}(s\sigma\tau) = \mathsf{col}(a) = a$. As s is a pure c-term, we have that $a \in D_c$ and therefore $R^{\tilde{n}}_{c,\tilde{s}}(a) = a$. Therefore $R^{\tilde{n}}_{c,\tilde{s}}(\mathsf{col}(s\sigma\tau)) = R^{\tilde{n}}_{c,\tilde{s}}(\mathsf{col}(a)) = R^{\tilde{n}}_{c,\tilde{s}}(a) = a$.

   Therefore $R^{\tilde{n}}_{c,\tilde{s}}(\mathsf{col}(s\sigma\tau)) = a = s\sigma'\tau'$.

4. otherwise $s = f(t_1, \ldots, t_n)$ for some pure c-terms $t_1, \ldots, t_n$ and we distinguish between two cases:

   (a) either $\mathsf{col}(s\sigma\tau) = f(\mathsf{col}(t_1\sigma\tau), \ldots, \mathsf{col}(t_n\sigma\tau))$, in which case $R^{\tilde{n}}_{c,\tilde{s}}(\mathsf{col}(s\sigma\tau))$ is by definition $f(R^{\tilde{n}}_{c,\tilde{s}}(\mathsf{col}(t_1\sigma\tau)), \ldots, R^{\tilde{n}}_{c,\tilde{s}}(\mathsf{col}(t_n\sigma\tau)))$. By the induction hypothesis,

   $$R^{\tilde{n}}_{c,\tilde{s}}(\mathsf{col}(t_i\sigma\tau)) =_{\mathsf{E}} t_i\sigma'\tau' \text{ for all } 1 \leq i \leq n$$

   and therefore $R^{\tilde{n}}_{c,\tilde{s}}(\mathsf{col}(s\sigma\tau)) =_{\mathsf{E}} f(t_1\sigma'\tau', \ldots, t_n\sigma'\tau') = s\sigma'\tau'$.

   (b) or $f(\mathsf{col}(t_1\sigma\tau), \ldots, \mathsf{col}(t_n\sigma\tau)) = C[[u_1, \ldots, u_m]], C[n_1, \ldots, n_m] =_{\mathsf{E}_d} n_x$ where $n_1, \ldots, n_m$ are fresh names such that $n_i = n_j$ iff $u_i =_{\mathsf{E}} u_j$ $(1 \leq i, j \leq m)$, where $c \in \{a, b\}$ is such that $f \in \mathcal{F}_d$ and $\mathsf{col}(s\sigma\tau) = u_x$. As $C[n_1, \ldots, n_m] =_{\mathsf{E}_d} n_x$ and $\mathsf{E}_d \subseteq \mathsf{E}$, we have that $C[n_1, \ldots, n_m] =_{\mathsf{E}} n_x$.
   As C is not the empty context, we have that each u_i $(1 \leq i \leq m$ is a subterm of some term $\mathsf{col}(t_j\sigma\tau)$ (for $1 \leq j \leq n$). Therefore, by Lemma 6.4, we have that $\mathsf{col}(u_i) = u_i$.
   Therefore the left-hand side of the equality we are trying to prove is $R^{\tilde{n}}_{c,\tilde{s}}(\mathsf{col}(s\sigma\tau)) = R^{\tilde{n}}_{c,\tilde{s}}(u_x)$.
   But the right-hand side (from hereon rhs) of the equality is $s\sigma'\tau' = f(t_1\sigma'\tau', \ldots, t_n\sigma'\tau')$. By the induction hypothesis $t_i\sigma'\tau' =_{\mathsf{E}} R^{\tilde{n}}_{c,\tilde{s}}(\mathsf{col}(t_i\sigma\tau))$ for all $1 \leq i \leq n$.
   Therefore the rhs is $s\sigma'\tau' =_{\mathsf{E}} f(R^{\tilde{n}}_{c,\tilde{s}}(\mathsf{col}(t_1\sigma\tau)), \ldots, R^{\tilde{n}}_{c,\tilde{s}}(\mathsf{col}(t_n\sigma\tau)))$, which is equal, by the definition of $R^{\tilde{n}}_{c,\tilde{s}}$, to $R^{\tilde{n}}_{c,\tilde{s}}(f(\mathsf{col}(t_1\sigma\tau), \ldots, \mathsf{col}(t_n\sigma\tau)))$.
   But $f(\mathsf{col}(t_1\sigma\tau), \ldots, \mathsf{col}(t_n\sigma\tau)) = C[[u_1, \ldots, u_m]]$ and therefore the rhs is $s\sigma'\tau' =_{\mathsf{E}} R^{\tilde{n}}_{c,\tilde{s}}(C[u_1, \ldots, u_m])$. As C is a pure c-context, it follows that $R^{\tilde{n}}_{c,\tilde{s}}(C[u_1, \ldots, u_m]) = C[R^{\tilde{n}}_{c,\tilde{s}}(u_1), \ldots, R^{\tilde{n}}_{c,\tilde{s}}(u_m)]$.
   By the choice of $n_1, \ldots, n_m$, we have that $n_i = n_j$ implies $u_i =_{\mathsf{E}} u_j$, which implies, by Lemma 6.7, $R^{\tilde{n}}_{c,\tilde{s}}(\mathsf{col}(u_i)) =_{\mathsf{E}} R^{\tilde{n}}_{c,\tilde{s}}(\mathsf{col}(u_j))$. As $\mathsf{col}(u_i) = u_i$ for all $1 \leq i \leq m$, we obtain that $n_i = n_j$ implies $R^{\tilde{n}}_{c,\tilde{s}}(u_i) =_{\mathsf{E}} R^{\tilde{n}}_{c,\tilde{s}}(u_j)$. As $C[n_1, \ldots, n_m] =_{\mathsf{E}} n_x$ and as the

equational theory E is stable by replacement of names by arbitrary terms, we obtain that $C[R^{\tilde{n}}_{c,\tilde{s}}(u_1),\ldots,R^{\tilde{n}}_{c,\tilde{s}}(u_m)] =_{\mathsf{E}} R^{\tilde{n}}_{c,\tilde{s}}(u_x)$.

Therefore $s\sigma'\tau' =_{\mathsf{E}} C[R^{\tilde{n}}_{c,\tilde{s}}(u_1),\ldots,R^{\tilde{n}}_{c,\tilde{s}}(u_m)] =_{\mathsf{E}} R^{\tilde{n}}_{c,\tilde{s}}(u_x)$.

We have shown that both the lhs $R^{\tilde{n}}_{c,\tilde{s}}(\mathsf{col}(s\sigma\tau)) = R^{\tilde{n}}_{c,\tilde{s}}(u_x)$ and the rhs $s\sigma'\tau' =_{\mathsf{E}} R^{\tilde{n}}_{c,\tilde{s}}(u_x)$ are equal modulo E to the same term and therefore $R^{\tilde{n}}_{c,\tilde{s}}(\mathsf{col}(s\sigma\tau)) =_{\mathsf{E}} s\sigma'\tau'$.

□

We are now ready to prove Theorem 6.1.

Theorem 6.1. *Assume $\varphi_0 \not\vdash t\sigma$ for any $t \in \{x_1,\ldots,x_{p'},y_1,\ldots,y_{q'}\}$ and that $x_i\sigma_0 \neq_{\mathsf{E}} y_j\sigma_0$ for all $1 \le i \le p', 1 \le j \le q'$. Then there exist S',φ',σ' such that*

$$(R'',\emptyset,\emptyset) \xrightarrow{l_1,\ldots,l_k,\S}{}^{*} (S',\varphi',\sigma') \tag{6.2}$$

and

1. *if r is a minimal recipe such that $\varphi \vdash^r x\sigma_0$ for some $x \in \{x_1,\ldots,x_{p'},y_1,\ldots,y_{q'}\}$, then $\varphi' \vdash^r x^d\sigma'_0$ for some $d \in \{a,b\}$.*
2. *otherwise, if $\varphi \vdash^r x\sigma$ for $x \in \mathcal{V}ar(P) \cup \mathcal{V}ar(Q) \setminus \{x_1,\ldots,x_{p'},y_1,\ldots,y_{q'}\}$ then $\varphi' \vdash^r x^d\sigma'$ for some $d \in \{a,b\}$.*

Furthermore, if $x,y \in \mathcal{D}om(\sigma)$ are such that $x\sigma =_{\mathsf{E}} y\sigma$ and $x^d,y^d \in \mathcal{D}om(\sigma')$ for some $d \in \{a,b\}$, then $x^d\sigma' =_{\mathsf{E}} y^d\sigma'$.

Proof.

Assumption 1. We assume w.l.o.g. that fresh names chosen by actions $P_j = \nu x$ $(1 \le j \le n)$ in R'' coming from P are from $\mathcal{N}_a$ and that fresh names chosen by actions $Q_j = \nu x$ $(1 \le j \le m)$ in R'' coming from Q are from $\mathcal{N}_b$.

Let $\{n_i\}_{1\le i\le p'}, \{m_i\}_{1\le i\le q'}$ be fresh names such that:

1. $n_i = n_j$ iff $x_i\sigma_0 =_{\mathsf{E}} x_j\sigma_0$ for all $1 \le i,j \le p'$,
2. $n_i = m_j$ iff $x_i\sigma_0 =_{\mathsf{E}} y_j\sigma_0$ for all $1 \le i \le p', 1 \le j \le q'$ and
3. $m_i = m_j$ iff $y_i\sigma_0 =_{\mathsf{E}} y_j\sigma_0$ for all $1 \le i,j \le q'$.

Let $\tilde{s}$ denote the sequence of secrets shared between the two protocols

$$\tilde{s} = x_1\sigma_0,\ldots,x_{p'}\sigma_0,y_1\sigma_0,\ldots,y_{q'}\sigma_0$$

and $\tilde{n}$ denote the fresh names chosen above associated with these shared secrets:

$$\tilde{n} = n_1,\ldots,n_{p'},m_1,\ldots,m_{q'}.$$

Let $\varphi' = R^{\tilde{n}}_{a,\tilde{s}}(\mathsf{col}(\varphi))$ and let σ' be defined as follows:

1. $\sigma'(x^a) = R^{\tilde{n}}_{a,\tilde{s}}(\mathsf{col}(\sigma(x)))$ for all variables $x \in \mathcal{V}ar(P)$ of P,
2. $\sigma'(x^b) = R^{\tilde{n}}_{b,\tilde{s}}(\mathsf{col}(\sigma(x)))$ for all variables $x \in \mathcal{V}ar(Q)$ of Q,
3. $\sigma'(z_i) = n_i$ for all $1 \le i \le p'$ and
4. $\sigma'(z_i) = m_i$ for all $1 \le i \le q'$.

Let $W_i = R_i.\ldots.R_{n+m}$ and $W'_i = R'_i.\ldots.R'_{n+m}$ denote suffixes of the traces R and R' for all $1 \leq i \leq n+m$. We let $W_{n+m+1} = W'_{n+m+1} = 0$. We have by hypothesis that

$$(R, \emptyset, \emptyset) \xrightarrow{l_1,\ldots,l_k}{}^{*} (S_0, \varphi_0, \sigma_0) \xrightarrow{\S} (S, \varphi, \sigma). \tag{6.3}$$

Therefore there exist $\varphi_1, \ldots, \varphi_l, \sigma_1, \ldots, \sigma_l$ such that:

$$(W_1, \varphi_1, \sigma_1) \xrightarrow{\S_1} (W_2, \varphi_2, \sigma_2) \xrightarrow{\S_2} \ldots \xrightarrow{\S_{l-1}} (W_l, \varphi_l, \sigma_l) \xrightarrow{\S} (S, \varphi, \sigma),$$

where $\varphi_1 = \{\}$ is the empty frame and $\sigma_1 = \{\}$ is the identity substitution. We also have that $\varphi_l = \varphi_0$ and that $\sigma_l = \sigma_0$.

We have that for all $1 \leq i \leq l$, $\varphi_i = \varphi[\mathcal{W}_i]$ for some $\mathcal{W}_i \in \mathcal{W}$ is a restriction of φ and $\sigma_i = \sigma[fv(W_i)]$ is a restriction of σ. For all $1 \leq i \leq l$, let $\varphi'_i = \varphi'[\mathcal{W}_i]$ and $\sigma'_i = \sigma'[fv(W'_i)]$ be the corresponding restriction on φ' and σ'.

Claim 6.1. *Let $T(a) = P$ and $T(b) = Q$. For all $1 \leq i \leq l$, for all $d \in \{a, b\}$, for all pure d-terms $t \in T(\mathcal{F}_d, (fv(W_i) \cap \mathcal{V}ar(T(d))) \cup \mathcal{N}_d \cup \mathcal{M}_d)$, we have that there exist:*

1. *a pure d-term $s^i_d(t)$ with $\mathcal{V}ar(s^i_d(t)) \subseteq \{z_1, \ldots, z_r\} \uplus \{z'_1, \ldots, z'_{r'}\}$ for some set of variables $\{z_1, \ldots, z_r, z'_1, \ldots z'_{r'}\}$,*
2. *a substitution $\pi^i_d(t)$ with $\mathcal{D}om(\pi^i_d(t)) = \{z_1, \ldots, z_r\}$ such that*

$$\pi^i_d(t)(z_j) \in_{\mathsf{E}} \{x_1\sigma_0, \ldots, x_{p'}\sigma_0, y_1\sigma_0, \ldots, y_{q'}\sigma_0\}$$

   for all $1 \leq j \leq r$ where $\in_{\mathsf{E}}$ denotes membership modulo E and

3. *a substitution $\tau^i_d(t)$ with $\mathcal{D}om(\tau^i_d(t)) = \{z'_1, \ldots, z'_{r'}\}$ such that*

$$\tau^i_d(t)(z'_j) = r^d_j \varphi_i$$

   for some recipe r^d_j for all $1 \leq j \leq r'$

such that

$$t\sigma =_{\mathsf{E}} s^i_d(t)\pi^i_d(t)\tau^i_d(t) \tag{6.4}$$

and

$$tR^{\tilde{n}}_{a,\tilde{s}}(\mathsf{col}(\sigma)) =_{\mathsf{E}} s^i_d(t)R^{\tilde{n}}_{a,\tilde{s}}(\mathsf{col}(\pi^i_d(t)))R^{\tilde{n}}_{a,\tilde{s}}(\mathsf{col}(\tau^i_d(t))). \tag{6.5}$$

Proof. We will show that the claim holds by induction on $(i, |t|)$ equipped with the lexicographic order.

1. for $i = 1$, we have that $fv(W_i) = fv(R) = \emptyset$ and therefore $t \in T(\mathcal{F}_d, \mathcal{N}_d \cup \mathcal{M}_d)$. We choose $s^i_d(t) = t$, $\pi^i_d(t) = \{\}$ and $\tau^i_d(t) = \{\}$ to conclude.
2. for $i \geq 2$, we distinguish between the following cases:
   (a) if $t = n \in \mathcal{N}_d \cup \mathcal{M}_d$, we choose $s^i_d(t) = t$, $\pi^i_d(t) = \{\}$ and $\tau^i_d(t) = \{\}$ to conclude.
   (b) if $t = x \in fv(W_{i-1})$, we choose $s^i_d(t) = s^{i-1}_d(t)$, $\pi^i_d(t) = \pi^{i-1}_d(t)$ and $\tau^i_d(t) = \tau^{i-1}_d(t)$ to conclude.
   (c) if $t = x \in fv(W_i) \setminus fv(W_{i-1})$, we have one of the following cases:
      i. either $W_i = \nu x.W_{i+1}$, in which case we distinguish between the following cases:
         A. if $x \in \{x_1, \ldots, x_{p'}, y_1, \ldots, y_{q'}\}$ then we let $s^i_d(x) = z^d$ for a fresh variable $z^d \in \mathcal{X}_d$, $\pi^i_d(x) = \{z^d \mapsto \sigma(x)\}$ and $\tau^i_d(x) = \{\}$ to conclude.
         B. otherwise x appears in only one of $P = T(a)$ or $Q = T(b)$. Assume that it only appears in $T(e)$ for $e \in \{a, b\}$. By Assumption 1, we have that $\sigma(x) = n \in \mathcal{N}_e$. We define $s^i_e(x) = \sigma(x) = n$, $\pi^i_e(x) = \{\}$ and $\tau^i_e(x) = \{\}$ to conclude.

ii. or $W_i = \mathbf{receive}(x).W_{i+1}$, in which case we let $s^i_d(x) = z^d$ for a fresh variable $z^d \in \mathcal{X}_d$, $\pi^i_d(x) = \{\}$ and $\tau^i_d(x) = \{z^d \mapsto r_i\varphi_i\}$ to conclude.

iii. or $W_i = (x := s).W_{i+1}$ for some term s, in which case we distinguish two cases:

A. if $x \in \{x_1, \ldots, x_{p'}, y_1, \ldots, y_{q'}\}$, then we let $s^i_d(x) = z^d$ for a fresh variable $z^d \in \mathcal{X}_d$, $\pi^i_d(x) = \{z^d \mapsto \sigma(x)\}$ and $\tau^i_d(x) = \{\}$ to conclude.

B. otherwise x appears in only one of $P = T(a)$ or $Q = T(b)$. Assume that it only appears in $T(e)$ for $e \in \{a, b\}$. In this case, s is a pure e-term and we choose $s^i_e(x) = s^{i-1}_e(s)$, $\pi^i_e(x) = \pi^{i-1}_e(s)$ and $\tau^i_e(x) = \tau^{i-1}_e(s)$ to conclude.

(d) if $t = f(t_1, \ldots, t_k)$ for a function symbol $f \in \mathcal{F}_a$ of arity $\mathsf{ar}(f) = k$, then we choose $s^i_d(t) = f(s^i_d(t_1), \ldots, s^i_d(t_k))$, $\pi^i_d(t) = \bigcup_{1\leq j\leq k} \pi^i_d(t_j)$ and $\tau^i_d(t) = \bigcup_{1\leq j\leq k} \tau^i_d(t_j)$ to conclude.

In each of the cases, Equation (6.4) and Equation (6.5) follows by syntactic simplifications. □

We next show that:

$$(R'', \emptyset, \emptyset) \rightarrow^* (W'_1, \varphi'_1, \sigma'_1) \xrightarrow{\S_1} (W'_2, \varphi'_2, \sigma'_2) \xrightarrow{\S_2} \ldots \xrightarrow{\S_{l-1}} (W'_l, \varphi'_l, \sigma'_l) \xrightarrow{\S} (S', \varphi', \sigma')$$

We have that $\sigma'_1 = \sigma[fv(R')] = \{z_i \mapsto n_i\}_{1\leq i\leq n} \cup \{z_{p+i} \mapsto m_i\}_{1\leq i\leq m} \cup \{x^a_i \mapsto n_i\}_{1\leq i\leq p'} \cup \{y^b_i \mapsto m_i\}_{1\leq i\leq q'}$ where $fv(R') = \{z_i\}_{1\leq i\leq p+q} \cup \{x^a_i\}_{1\leq i\leq p'} \cup \{y^b_i\}_{1\leq i\leq q'}$. We immediately have that the transitions $(R'', \emptyset, \emptyset) \rightarrow^* (R', \varphi'_1, \sigma'_1)$ succeed by definition of R'' and choice of $n'_1, \ldots, n'_p, m'_1, \ldots, m'_q$. But $R' = W'_1$ by choice of W'_1 and therefore $(R'', \emptyset, \emptyset) \rightarrow^* (W'_1, \varphi'_1, \sigma'_1)$.

For all $1 \leq i \leq l$, we prove that

$$(W_i, \varphi_i, \sigma_i) \xrightarrow{\S_i} (W_{i+1}, \varphi_{i+1}, \sigma_{i+1}) \text{ implies } (W'_i, \varphi'_i, \sigma'_i) \xrightarrow{\S_i} (W'_{i+1}, \varphi'_{i+1}, \sigma'_{i+1}). \tag{6.6}$$

We assume that $W_i = P_j.W_{i+1}$ with P_j being an action in P, since the case where $W_i = Q_j.W_{i+1}$ with Q_j being an action in Q is completely analogous. We distinguish among five cases:

1. if $P_j = \nu x$, then $W_i = \nu x.W_{i+1}$. As $(W_i, \varphi_i, \sigma_i) \xrightarrow{\S_i} (W_{i+1}, \varphi_{i+1}, \sigma_{i+1})$, we have that $\S_i$ is the empty string, $\varphi_{i+1} = \varphi_i$ and $\sigma_{i+1} = \sigma_i \cup \{x \mapsto n\}$ where $n \in \mathcal{N}_a$ is a fresh name.

   By definition, we have that $W'_i = \nu x^a.W'_{i+1}$, $\varphi'_{i+1} = R^{\tilde{n}}_{a,\tilde{s}}(\mathsf{col}(\varphi_{i+1})) = R^{\tilde{n}}_{a,\tilde{s}}(\mathsf{col}(\varphi_i)) = \varphi'_i$ and $\sigma'_{i+1} = \sigma'_i \cup \{x^a \mapsto R^{\tilde{n}}_{a,\tilde{s}}(\mathsf{col}(\sigma(x)))\} = \sigma'_i \cup \{x^a \mapsto n\}$.

   As n was chosen to be fresh and as $x^a \notin \mathcal{D}om(\sigma'_i)$, it follows that $(\nu x^a.W'_{i+1}, \varphi'_i, \sigma'_i) \xrightarrow{\S_i} (W'_{i+1}, \varphi'_i, \sigma'_i \cup \{x^a \mapsto n\})$. But $W'_i = \nu x^a.W'_{i+1}$, $\varphi'_{i+1} = \varphi'_i$ and $\sigma'_{i+1} = \sigma'_i \cup \{x^a \mapsto n\}$ and therefore
   $$(W'_i, \varphi'_i, \sigma'_i) \xrightarrow{\S_i} (W'_{i+1}, \varphi'_{i+1}, \sigma'_{i+1}),$$
   which is what we had to prove.

2. if $P_j = \mathbf{receive}(x)$, then $W_i = \mathbf{receive}(x).W_{i+1}$. As $(W_i, \varphi_i, \sigma_i) \xrightarrow{\S_i} (W_{i+1}, \varphi_{i+1}, \sigma_{i+1})$, we have that $\S_i = \mathbf{receive}(r_i)$ for some recipe r_i, $\varphi_{i+1} = \varphi_i$ and $\sigma_{i+1} = \sigma_i \cup \{x \mapsto r_i\varphi_i\}$.

   By definition, we have that $W'_i = \mathbf{receive}(x^a).W'_{i+1}$, $\varphi'_{i+1} = R^{\tilde{n}}_{a,\tilde{s}}(\mathsf{col}(\varphi_{i+1})) = R^{\tilde{n}}_{a,\tilde{s}}(\mathsf{col}(\varphi_i)) = \varphi'_i$ and $\sigma'_{i+1} = \sigma'_i \cup \{x^a \mapsto R^{\tilde{n}}_{a,\tilde{s}}(\mathsf{col}(\sigma(x)))\} = \sigma'_i \cup \{x^a \mapsto R^{\tilde{n}}_{a,\tilde{s}}(\mathsf{col}(r_i\varphi_i))\}$.

   As φ_i is a restriction of $\varphi_0 = \varphi_l$ to a smaller domain, we have that $\varphi_i \not\vdash t$ for any $t \in \{x_1\sigma_0, \ldots, x_{p'}\sigma_0, y_1\sigma_0, \ldots, y_{q'}\sigma_0\}$. Therefore, by Lemma 6.9, we have that $r_i R^{\tilde{n}}_{a,\tilde{s}}(\mathsf{col}(\varphi_i)) =_{\mathsf{E}} R^{\tilde{n}}_{a,\tilde{s}}(\mathsf{col}(r_i\varphi_i))$. This is equivalent to $r_i\varphi'_i =_{\mathsf{E}} R^{\tilde{n}}_{a,\tilde{s}}(\mathsf{col}(\sigma(x))$, which is equivalent to $\varphi'_i \vdash^{r_i} R^{\tilde{n}}_{a,\tilde{s}}(\mathsf{col}(\sigma(x)))$. Therefore $(\mathbf{receive}(x^a).W'_{i+1}, \varphi'_i, \sigma'_i) \xrightarrow{\S_i} (W'_{i+1}, \varphi'_i, \sigma'_i \cup \{x^a \mapsto R^{\tilde{n}}_{a,\tilde{s}}(\mathsf{col}(\sigma(x)))\})$. But $W'_i = \mathbf{receive}(x^a).W'_{i+1}$, $\varphi'_{i+1} = \varphi'_i$ and $\sigma'_{i+1} = \sigma'_i \cup \{x^a \mapsto R^{\tilde{n}}_{a,\tilde{s}}(\mathsf{col}(\sigma(x)))\}$ and therefore
   $$(W'_i, \varphi'_i, \sigma'_i) \xrightarrow{\S_i} (W'_{i+1}, \varphi'_{i+1}, \sigma'_{i+1}),$$
   which is what we had to prove.

3. if $P_j = (x := t)$, then $W_i = (x := t).W_{i+1}$. As $(W_i, \varphi_i, \sigma_i) \xrightarrow{\S_i} (W_{i+1}, \varphi_{i+1}, \sigma_{i+1})$, we have that $\S_i$ is the empty string, $\varphi_{i+1} = \varphi_i$ and $\sigma_{i+1} = \sigma_i \cup \{x \mapsto t\sigma_i\}$.

   By definition, we have that $W'_i = (x^a := t\{y \mapsto y^A\}_{y \in \mathcal{V}ar(t)}).W'_{i+1}$. Therefore $(W'_i, \varphi'_i, \sigma'_i) \xrightarrow{\S_i} (W'_{i+1}, \varphi'_i, \sigma'_i \cup \{x^a \mapsto t\{y \mapsto y^a\}_{y \in \mathcal{V}ar(t)}\sigma'_i\})$. But we have that:

$$
\begin{array}{lcll}
t\{y \mapsto y^a\}_{y \in \mathcal{V}ar(t)}\sigma'_i & = & tR^{\tilde{n}}_{a,\tilde{s}}(\mathsf{col}(\sigma)) & \text{(by choice of } \sigma') \\
& =_{\mathsf{E}} & s^i_a(t)R^{\tilde{n}}_{a,\tilde{s}}(\mathsf{col}(\pi^i_a(t)))R^{\tilde{n}}_{a,\tilde{s}}(\mathsf{col}(\tau^i_a(t))) & \text{(by Claim 6.1)} \\
& =_{\mathsf{E}} & R^{\tilde{n}}_{a,\tilde{s}}(\mathsf{col}(s^i_a(t)\pi^i_a(t)\tau^i_a(t))) & \text{(by Lemma 6.10)} \\
& =_{\mathsf{E}} & R^{\tilde{n}}_{a,\tilde{s}}(\mathsf{col}(t\sigma)) & \text{(by Claim 6.1, Lemma 6.7)} \\
& =_{\mathsf{E}} & R^{\tilde{n}}_{a,\tilde{s}}(\mathsf{col}(\sigma(x))) & \text{(by choice of } \sigma) \\
& =_{\mathsf{E}} & \sigma'(x^a) & \text{(by definition of } \sigma')
\end{array}
$$

Figure 6.4: Transformation of $t\{y \mapsto y^a\}_{y \in \mathcal{V}ar(t)}\sigma'_i$

   and therefore $\sigma'_{i+1} =_{\mathsf{E}} \sigma_i \cup \{t\{y \mapsto y^a\}_{y \in \mathcal{V}ar(t)}\sigma'_i\}$. We also have that $\varphi'_{i+1} = R^{\tilde{n}}_{a,\tilde{s}}(\mathsf{col}(\varphi_{i+1})) = R^{\tilde{n}}_{a,\tilde{s}}(\mathsf{col}(\varphi_i)) = \varphi'_i$ and therefore

$$(W'_i, \varphi'_i, \sigma'_i) \xrightarrow{\S_i} (W'_{i+1}, \varphi'_{i+1}, \sigma'_{i+1}),$$

   which is what we had to prove.

4. if $P_j = \mathbf{send}(t)$, we have that $W_i = \mathbf{send}(t).W_{i+1}$. As $(W_i, \varphi_i, \sigma_i) \xrightarrow{\S_i} (W_{i+1}, \varphi_{i+1}, \sigma_{i+1})$, we have that $\S_i$ is the empty string, $\varphi_{i+1} = \varphi_i \cup \{w_{|\mathcal{D}om(\varphi_i)|+1} \mapsto t\sigma_i\}$ and $\sigma_{i+1} = \sigma_i$.

   By definition, we have that $W'_i = \mathbf{send}(t\{y \mapsto y^A\}_{y \in \mathcal{V}ar(t)}).W'_{i+1}$. Therefore $(W'_i, \varphi'_i, \sigma'_i) \xrightarrow{\S_i} (W'_{i+1}, \varphi'_i \cup \{w_{|\mathcal{D}om(\varphi'_i)|+1} \mapsto t\{y \mapsto y^a\}_{y \in \mathcal{V}ar(t)}\sigma'_i\}, \sigma'_i)$.

   By a transformation identical to Figure 6.4, we have that $t\{y \mapsto y^a\}_{y \in \mathcal{V}ar(t)}\sigma'_i =_{\mathsf{E}} R^{\tilde{n}}_{a,\tilde{s}}(\mathsf{col}(t\sigma))$. Therefore $\varphi'_{i+1} = \varphi'_i \cup \{w_{|\mathcal{D}om(\varphi'_i)|+1} \mapsto R^{\tilde{n}}_{a,\tilde{s}}(\mathsf{col}(t\sigma))\} =_{\mathsf{E}} t\{y \mapsto y^a\}_{y \in \mathcal{V}ar(t)}\sigma'_i$. Moreover, $\sigma'_{i+1} = \sigma'_i$ and therefore

$$(W'_i, \varphi'_i, \sigma'_i) \xrightarrow{\S_i} (W'_{i+1}, \varphi'_{i+1}, \sigma'_{i+1}),$$

   which is what we had to prove.

5. if $P_j = [s = t]$, we have that $W_i = [s = t].W_{i+1}$. As $(W_i, \varphi_i, \sigma_i) \xrightarrow{\S_i} (W_{i+1}, \varphi_{i+1}, \sigma_{i+1})$, we have that $\S_i$ is the empty string, $\varphi_{i+1} = \varphi_i$, $\sigma_{i+1} = \sigma_i$ and $s\sigma =_{\mathsf{E}} t\sigma$.

   By definition, we have that $W'_i = [s\{y \mapsto y^A\}_{y \in \mathcal{V}ar(s)} = t\{y \mapsto y^A\}_{y \in \mathcal{V}ar(t)}].W'_{i+1}$. By transformations identical to those in Figure 6.4, we have that $t\{y \mapsto y^a\}_{y \in \mathcal{V}ar(t)}\sigma'_i =_{\mathsf{E}} R^{\tilde{n}}_{a,\tilde{s}}(\mathsf{col}(t\sigma))$ and that $s\{y \mapsto y^a\}_{y \in \mathcal{V}ar(s)}\sigma'_i =_{\mathsf{E}} R^{\tilde{n}}_{a,\tilde{s}}(\mathsf{col}(s\sigma))$. But as $s =_{\mathsf{E}} t$, we have that $R^{\tilde{n}}_{a,\tilde{s}}(\mathsf{col}(s\sigma)) =_{\mathsf{E}} R^{\tilde{n}}_{a,\tilde{s}}(\mathsf{col}(t\sigma))$ by Lemma 6.7. Therefore $t\{y \mapsto y^a\}_{y \in \mathcal{V}ar(t)}\sigma'_i =_{\mathsf{E}} s\{y \mapsto y^a\}_{y \in \mathcal{V}ar(s)}\sigma'_i$ and we have that $(W'_i, \varphi'_i, \sigma'_i) \xrightarrow{\S_i} (W'_{i+1}, \varphi'_i, \sigma'_i)$. But $\varphi'_{i+1} = \varphi'_i$ and $\sigma'_{i+1} = \sigma'_i$ and therefore

$$(W'_i, \varphi'_i, \sigma'_i) \xrightarrow{\S_i} (W'_{i+1}, \varphi'_{i+1}, \sigma'_{i+1}),$$

   which is what we had to prove.

We have shown that Equation (6.2) holds. It remains to show that:

1. if r is a minimal recipe such that $\varphi \vdash^r x\sigma_0$ for some $x \in \{x_1, \ldots, x_{p'}, y_1, \ldots, y_{q'}\}$, then $\varphi' \vdash^r x^d\sigma'_0$ for some $d \in \{a, b\}$.

   As $\varphi_0 \not\vdash_{\mathsf{E}} x\sigma_0$ for any variable $x \in \{x_1, \ldots, x_{p'}, y_1, \ldots, y_{q'}\}$ and as $\varphi \vdash^r_{\mathsf{E}} x\sigma_0$ for some variable $x \in \{x_1, \ldots, x_{p'}, y_1, \ldots, y_{q'}\}$, it follows that $\varphi \neq_{\mathsf{E}} \varphi_0$. As $(S_0, \varphi_0, \sigma_0) \xrightarrow{\S} (S, \varphi, \sigma)$,

we have that $S_0 = \mathbf{send}(t).S$ for some action $\mathbf{send}(t) = P_j \in P$ (for some $1 \leq j \leq n$) or $\mathbf{send}(t) = Q_j \in Q$ (for some $1 \leq j \leq m$), $\varphi = \varphi_0 \cup \{w_{|\mathcal{D}om(\varphi_0)|+1} \mapsto t\sigma_0\}$ and that $\sigma = \sigma_0$.

We will show that $\varphi' \vdash^r x^d\sigma'$ for some $d \in \{a, b\}$. Clearly the recipe $r \notin \mathcal{M}$ is not a name, since otherwise $\varphi_0 \vdash^r x\sigma_0$. If $r = w_k$ for some $1 \leq k \leq |\mathcal{D}om(\varphi)|$, then $\varphi' \vdash^r R^{\tilde{n}}_{a,\tilde{s}}(\mathsf{col}(t\sigma_0))$ by the definition of φ'. But as $\varphi \vdash^r t\sigma_0$, we have that $t\sigma_0 =_{\mathsf{E}} x\sigma$ and $x^d\sigma' = R^{\tilde{n}}_{a,\tilde{s}}(\mathsf{col}(x\sigma))$ by definition of σ' and we obtain $x^a\sigma' = R^{\tilde{n}}_{a,\tilde{s}}(\mathsf{col}(t\sigma_0))$ by Lemma 6.7. Therefore $\varphi' \vdash^r x^a\sigma$. Otherwise $r = f(r_1, \ldots, r_l)$ for some recipes $r_1, \ldots, r_l$. By minimality of r, we have that any subrecipe $r' \sqsubseteq r_i$ $(1 \leq i \leq l)$ of $r_1, \ldots, r_l$ is not a recipe for any $y\sigma$ with $y \in \{x_1, \ldots, x_{p'}, y_1, \ldots, y_{q'}\}$. Let $d = \mathsf{domain}(f)$ and let $s = f(z_1, \ldots, z_l)$ be a pure d-term. Let $\pi = \{\}$ and $\tau = \{z_i \mapsto r_i\varphi\}_{1 \leq i \leq l}$. By Lemma 6.8, we have that $R^{\tilde{n}}_{a,\tilde{s}}(\mathsf{col}(\varphi)) = R^{\tilde{n}}_{b,\tilde{s}}(\mathsf{col}(\varphi)) = R^{\tilde{n}}_{d,\tilde{s}}(\mathsf{col}(\varphi))$. Therefore we can apply Lemma 6.10 to obtain that $R^{\tilde{n}}_{d,\tilde{s}}(\mathsf{col}(s\pi\tau)) = sR^{\tilde{n}}_{d,\tilde{s}}(\mathsf{col}(\pi))R^{\tilde{n}}_{d,\tilde{s}}(\mathsf{col}(\tau))$. But $\pi = \{\}$ and therefore $R^{\tilde{n}}_{d,\tilde{s}}(\mathsf{col}(s\tau)) = sR^{\tilde{n}}_{d,\tilde{s}}(\mathsf{col}(\tau))$ which implies

$$R^{\tilde{n}}_{d,\tilde{s}}(\mathsf{col}(r\varphi)) = f(r_1R^{\tilde{n}}_{d,\tilde{s}}(\mathsf{col}(\varphi)), \ldots, r_lR^{\tilde{n}}_{d,\tilde{s}}(\mathsf{col}(\varphi))) = rR^{\tilde{n}}_{d,\tilde{s}}(\mathsf{col}(\varphi)).$$

But $R^{\tilde{n}}_{d,\tilde{s}}(\mathsf{col}(r\varphi)) = R^{\tilde{n}}_{d,\tilde{s}}(\mathsf{col}(x\sigma)) = x^d\sigma'$ and $rR^{\tilde{n}}_{d,\tilde{s}}(\mathsf{col}(\varphi)) = r\varphi'$. Therefore $\varphi' \vdash^r x^d\sigma'$, which is what we had to prove.

2. otherwise, if $\varphi \vdash^r x\sigma$ for $x \in \mathcal{V}ar(P) \cup \mathcal{V}ar(Q) \setminus \{x_1, \ldots, x_{p'}, y_1, \ldots, y_{q'}\}$ then $\varphi' \vdash^r x^d\sigma'$ for all $d \in \{a, b\}$.

   We have that $r\varphi =_{\mathsf{E}} x\sigma$. By Lemma 6.7 we obtain that $R^{\tilde{n}}_{a,\tilde{s}}(\mathsf{col}(r\varphi)) =_{\mathsf{E}} R^{\tilde{n}}_{a,\tilde{s}}(\mathsf{col}(x\sigma))$. We have by Lemma 6.9 that $R^{\tilde{n}}_{a,\tilde{s}}(\mathsf{col}(r\varphi)) =_{\mathsf{E}} rR^{\tilde{n}}_{a,\tilde{s}}(\mathsf{col}(\varphi))$ and therefore we obtain that $rR^{\tilde{n}}_{a,\tilde{s}}(\mathsf{col}(\varphi)) =_{\mathsf{E}} R^{\tilde{n}}_{a,\tilde{s}}(\mathsf{col}(x\sigma))$. But $\varphi' = R^{\tilde{n}}_{a,\tilde{s}}(\mathsf{col}(\varphi))$ and $x^d\sigma' = R^{\tilde{n}}_{a,\tilde{s}}(\mathsf{col}(x\sigma))$ by definition. Therefore $r\varphi' =_{\mathsf{E}} x^d\sigma$ which immediately implies $\varphi' \vdash^r x^d\sigma'$, what we had to show.

It remains to show that if $x, y \in \mathcal{D}om(\sigma)$ are such that $x\sigma =_{\mathsf{E}} y\sigma$ and $x^d, y^d \in \mathcal{D}om(\sigma')$ for some $d \in \{a, b\}$, then $x^d\sigma' =_{\mathsf{E}} y^d\sigma'$. This follows immediately since $x^d\sigma' = R^{\tilde{n}}_{a,\tilde{s}}(\mathsf{col}(x\sigma))$ and $y^d\sigma' = R^{\tilde{n}}_{a,\tilde{s}}(\mathsf{col}(y\sigma))$ by definition. But $R^{\tilde{n}}_{a,\tilde{s}}(\mathsf{col}(x\sigma)) =_{\mathsf{E}} R^{\tilde{n}}_{a,\tilde{s}}(\mathsf{col}(y\sigma))$ (from $x\sigma =_{\mathsf{E}} y\sigma$ by Lemma 6.7), which implies $x^d\sigma' =_{\mathsf{E}} y^d\sigma'$.

□

6.5 Applications

6.5.1 Some Further Useful Lemmas

Theorem 6.1 is our key result for composing processes. We list here some other useful (and rather straightforward) results that we will use in our applications to secure protocol composition.

Theorem 6.1 is stated for *traces*. Given an arbitrary process P, we say that $Q_1 \ldots . Q_n$ is a *trace* of P if $P \overset{Q_1}{\hookrightarrow} P_1 \overset{Q_2}{\hookrightarrow} \ldots \overset{Q_n}{\hookrightarrow} P_n$ where the the transitions $\overset{\S}{\hookrightarrow}$ are defined in Figure 6.5. Intuitively, a process is equivalent to its set of traces.

We denote by $\mathcal{T}races(P)$ the set of traces of P. It intuitively consists of the set of all possible interleaving for the executions of P. We will use this set of reason about protocols containing parallel composition and replications using Theorem 6.1.

$$\textsc{New}\ \frac{P = \nu x.R}{P \overset{\nu x}{\hookrightarrow} R} \qquad \textsc{Input}\ \frac{P = \mathbf{receive}(x).R}{P \overset{\mathbf{receive}(x)}{\hookrightarrow} R} \qquad \textsc{Assgn}\ \frac{P = (x := t).R}{P \overset{x:=t}{\hookrightarrow} R}$$

$$\textsc{Test}\ \frac{P = [s = t].R}{P \overset{[s=t]}{\hookrightarrow} R} \qquad \textsc{Output}\ \frac{P = \mathbf{send}(t).R}{P \overset{\mathbf{send}(t)}{\hookrightarrow} R} \qquad \textsc{Parallel}\ \frac{P = (Q_0 \mid Q_1).R \qquad Q_0 \overset{\S}{\hookrightarrow} Q_0'}{P \overset{\S}{\hookrightarrow} Q_0' \mid Q_1}$$

Figure 6.5: Obtaining the Set of Traces of a Process.

We can immediately obtain the follows useful lemmas. A process P preserves a secret if and only if all its traces preserve the secret.

Lemma 6.11. *Let P be a process. Then for any equational theory E, $P \models_{\mathsf{E}} \mathsf{Secret}(x)$ iff for all $Q \in \mathcal{T}races(P)$ we have that $Q \models_{\mathsf{E}} \mathsf{Secret}(x)$.*

Proof. By induction on the number of transitions and case analysis. □

One can also notice that if a protocol reveals a secret then it *a fortiori* reveals it when projecting two names on a single one.

Lemma 6.12. *For any equational theory E, if $\nu x_1.\nu x_2.P \not\models_{\mathsf{E}} \mathsf{Secret}(x)$ then $\nu x_1.(x_2 := x_1).P \not\models_{\mathsf{E}} \mathsf{Secret}(x)$.*

Proof. By induction on the number of transitions in the trace leading to the revelation of x. □

We also need to show that, when mounting an attack on a process P over the d-domain (for some $d \in \{a, b\}$), the adversary is not more powerful when using the combined theory $\mathsf{E} = \mathsf{E}_a \cup \mathsf{E}_b$. This is captured by the following lemma.

Lemma 6.13. *If P is a trace over the d-domain and*

$$(P, \emptyset, \emptyset) \xrightarrow{\S_1}_{\mathcal{F},\mathsf{E}} (P_1, \varphi_1, \sigma_1) \xrightarrow{\S_2}_{\mathcal{F},\mathsf{E}} \cdots \xrightarrow{\S_n}_{\mathcal{F},\mathsf{E}} (P_n, \varphi_n, \sigma_n)$$

then

$$(P, \emptyset, \emptyset) \xrightarrow{\S_1'}_{\mathcal{F}_d,\mathsf{E}_d} (P_1, \varphi_1', \sigma_1') \xrightarrow{\S_2'}_{\mathcal{F}_d,\mathsf{E}_d} \cdots \xrightarrow{\S_n'}_{\mathcal{F}_d,\mathsf{E}_d} (P_n, \varphi_n', \sigma_n'))$$

for some φ_n', σ_n' and $\S_{\_}'$ such that $\varphi_n \vdash_{\mathsf{E}} x\sigma_n$ implies $\varphi_n' \vdash_{\mathsf{E}_d} x\sigma_n'$ and $x\sigma_n =_{\mathsf{E}} y\sigma_n$ implies $x\sigma_n' =_{\mathsf{E}_d} y\sigma_n'$ for all $x, y \in \mathcal{D}om(\sigma_n)$.

Note that this lemma relies on the assumption we made that E_b is not trivial (it does not equate all terms). Otherwise $\mathsf{E}_a \cup \mathsf{E}_b$ would be trivial and therefore all terms would be deducible. The proof of Lemma 6.13 is technical and can be found in Appendix A.2. From the above lemma, we immediately obtain:

Corollary 6.1. *If P is a process over the d-domain (for some $d \in \{a, b\}$) and $P \models_{\mathcal{F}_d,\mathsf{E}_d} \mathsf{Secret}(x)$, then $P \models_{\mathcal{F},\mathsf{E}} \mathsf{Secret}(x)$.*

6.5.2 Key-exchange Protocol

We are now ready to present some applications of Theorem 6.1. Our first application is the composition of a key-exchange protocol with another protocol that relies on the exchanged key.

It is often the case that a security protocol is verified assuming that some keys are already shared between the principals, abstracting away from the process by which these keys have been established. We can use our result to show that if a key exchange protocol was used to establish the key and if the two protocols use disjoint cryptographic primitives, their composition is secure provided that neither the key exchange protocol nor the main protocol reveal the established keys.

To state our result, we first need the following definition:

Definition 6.6. We say that P binds x if $P = P_1.P_2.\ldots.P_n$ and $P_j \in \{\mathbf{receive}(x), x := t, \nu x\}$ for some $1 \leq j \leq n$ and some term t (note that $P_1, \ldots, P_n$ are not necessarily atomic).

Theorem 6.2. *Let $P = \nu k_1.\ldots.\nu k_n.(P_1 \mid P_2)$ be a process over the a-domain and let $Q = \nu k.(x_k := k.Q_1 \mid y_k := k.Q_2)$ be a process over the b-domain such that:*

- *P_1 binds x_k and P_2 binds y_k*
- *$fv(P) = \emptyset$, $fv(Q) = \emptyset$ and $\mathcal{V}ar(P) \cap \mathcal{V}ar(Q) = \{x_k, y_k\}$*
- *$P \models_{\mathsf{E}_a} \mathsf{Secret}(x_k)$ and $P \models_{\mathsf{E}_a} \mathsf{Secret}(y_k)$*
- *$Q \models_{\mathsf{E}_b} \mathsf{Secret}(x_k)$ and $Q \models_{\mathsf{E}_b} \mathsf{Secret}(y_k)$.*

If $Q \models_{\mathsf{E}_b} \mathsf{Secret}(x_s)$ then $W = \nu k_1.\ldots.\nu k_n.(P_1.Q_1 \mid P_2.Q_2) \models_{\mathsf{E}} \mathsf{Secret}(x_s)$.

Intuitively, the protocol P corresponds to two roles P_1 and P_2 that establish a key k stored respectively in x_k for P_1 and in y_k for P_2. Then each of the two roles Q_1 and Q_2 of Q uses respectively its version of the key. Theorem 6.2 ensures that the protocol P can be safely abstracted by the generation of a single fresh key, distributed among the participants.

This result could easily be extended to an arbitrary number of roles. Note that Q_1 and Q_2 may contain replications thus the key k may be used in several distinct sessions.

Proof. If $d \in \{a, b\}$ and if P is a process, we denote by P^d the process in which any occurrence of a variable $x \in \mathcal{V}ar(P)$ has been replaced by the variable x^d.

We do a proof by contradiction. We assume that $W \not\models_{\mathsf{E}} \mathsf{Secret}(x_s)$. Then, by Lemma 6.11, we have that there exists a trace $R \in \mathcal{T}races(W)$ of W such that $R \not\models_{\mathsf{E}} \mathsf{Secret}(x_s)$.

The trace R is then a ground interleaving of a trace $P_0 \in \mathcal{T}races(P)$ and a trace $Q_0 \in \mathcal{T}races(Q_1 \mid Q_2)$. We denote by R' the same ground interleaving of P_0^a and Q_0^b.

As $R \not\models_{\mathsf{E}} \mathsf{Secret}(x_s)$, there is a run

$$(R, \emptyset, \emptyset) \xrightarrow{\S_1} (R_1, \varphi_1, \sigma_1) \xrightarrow{\S_2} \ldots \xrightarrow{\S_m} (R_m, \varphi_m, \sigma_m)$$

such that $\varphi_m \vdash x_s\sigma_m$.

Let $1 \leq l \leq m$ be the first index such that $\varphi_l \vdash x_k\sigma_l$ or $\varphi_l \vdash y_k\sigma_l$ or $\varphi_l \vdash x_s\sigma_l$.

We can then apply Theorem 6.1 to obtain that the process

$$R'' = \nu\{z_a, z_b\}.(x_k^b := z_a).(y_k^b := z_b).R'$$

reveals x_k^a, y_k^a, x_k^b, y_k^b or x_s^b in the equational theory E. (We do not know if z_a and z_b are the same variable). In either case, by Lemma 6.12, we have that

$$R''' = \nu k^b.(x_k^b := k^b).(y_k^b := k^b).R'$$

reveals x_k^a, y_k^a, x_k^b, y_k^b or x_s^b.

But R''' is an interleaving of some trace of P^a and Q^b. Therefore R''' is a trace of $P^a \mid Q^b$.

Therefore $P^a \mid Q^b$ reveals x_k^a, y_k^a, x_k^b, y_k^b or x_s^b. Assume $P^a \mid Q^b$ reveals x_k^a or y_k^a. Since P^a and Q^b share no data, Q^b can be simulated by the adversary and thus $P^a \not\models_{\mathsf{E}} \mathsf{Secret}(z)$ for some $z \in \{x_k^a, y_k^a\}$. If $P^a \mid Q^b$ reveals x_k^b, y_k^b or x_s^b, we similarly deduce that $Q^b \not\models_{\mathsf{E}} \mathsf{Secret}(z)$ for some $z \in \{x_k^b, y_k^b, x_s^b\}$. By Corollary 6.1, we obtain that $P \not\models_{\mathsf{E}_a} \mathsf{Secret}(z)$ for some $z \in \{x_k, y_k\}$ or that $Q \not\models_{\mathsf{E}_b} \mathsf{Secret}(z)$ for some $z \in \{x_k, y_k, x_s\}$. In both cases, this contradicts the hypotheses. We thus deduce that $W \models_{\mathsf{E}} \mathsf{Secret}(x_s)$.

□

6.5.3 Secure Channels

Another composition scenario is when a protocol is proven secure assuming some secure channels, that is, assuming that some secret key is established on the fly. We show that the secure channel can be implemented by any sub-protocol provided that neither the main protocol nor the sub-protocol reveal the key.

Theorem 6.3. *Let $P = \nu k_1.\ldots.\nu k_n.!(P_1 \mid P_2)$ be a process over the a-domain and let $Q = !(\nu k.(x_k := k.Q_1 \mid y_k := k.Q_2))$ be a process over the b-domain such that:*

1. *P_1 binds x_k and P_2 binds y_k*
2. *$fv(P) = fv(Q) = \emptyset$*
3. *$P \models_{\mathsf{E}_a} \mathsf{Secret}(x_k)$ and $P \models_{\mathsf{E}_a} \mathsf{Secret}(y_k)$*
4. *$Q \models_{\mathsf{E}_b} \mathsf{Secret}(x_k)$ and $Q \models_{\mathsf{E}_b} \mathsf{Secret}(y_k)$ and $Q \models_{\mathsf{E}_b} \mathsf{Secret}(x_s)$*

Then $R = \nu k_1.\ldots.\nu k_n.!(P_1.Q_1 \mid P_2.Q_2) \models_{\mathsf{E}} \mathsf{Secret}(x_s)$.

Compared to Theorem 6.2, the two roles Q_1 and Q_2 now use a different key k in each session.

Proof. We prove the result by contradiction along the same lines as the proof of Theorem 6.2. We assume that $W \not\models_{\mathsf{E}} \mathsf{Secret}(x_s)$. Then, by Lemma 6.11, we have that there exists a trace R of W such that $R \not\models_{\mathsf{E}} \mathsf{Secret}(x_s)$.

R is then a ground interleaving of a trace $P_0 \in \mathcal{T}races(P)$ and a trace $Q_0 \in \mathcal{T}races(!(Q_1 \mid Q_2))$. We denote by R' the same ground interleaving of P_0^a and Q_0^b.

As $R \not\models_{\mathsf{E}} \mathsf{Secret}(x_s)$, there is a trace

$$(R, \emptyset, \emptyset) \xrightarrow{\S_1} (R_1, \varphi_1, \sigma_1) \xrightarrow{\S_2} \ldots \xrightarrow{\S_m} (R_m, \varphi_m, \sigma_m)$$

such that $\varphi_m \vdash x_s\sigma_m$.

Let $1 \leq l \leq m$ be the first index such that $\varphi_l \vdash x_k\sigma_l$ or $\varphi_l \vdash y_k\sigma_l$ or $\varphi_l \vdash x_s\sigma_l$.

We can then apply Theorem 6.1 to obtain that the process

$$R'' = \nu z_a, z_b.(x_k^b := z_a).(y_k^b := z_b).R'$$

reveals x_k^a, y_k^a, x_k^b, y_k^b or x_s^b in the equational theory E. (We do not know if z_a and z_b are the same variable). In either case, by Lemma 6.12, we have that

$$R''' = \nu k^b.(x_k^b := k^b).(y_k^b := k^b).R'$$

reveals x_k^a, y_k^a, x_k^b, y_k^b or x_s^b.

But R''' is an interleaving of some trace of P^a and some trace of Q^b. Therefore R''' is a trace of $P^a \mid Q^b$.

Therefore $P^a \mid Q^b$ reveals x_k^a, y_k^a, x_k^b, y_k^b or x_s^b. Assume $P^a \mid Q^b$ reveals x_k^a or y_k^a. Since P^a and Q^b share no data, Q^b can be simulated by the adversary and thus $P^a \not\models_{\mathsf{E}} \mathsf{Secret}(z)$ for some $z \in \{x_k^a, y_k^a\}$. If $P^a \mid Q^b$ reveals x_k^b, y_k^b or x_s^b, we similarly deduce that $Q^b \not\models_{\mathsf{E}} \mathsf{Secret}(z)$ for some $z \in \{x_k^b, y_k^b, x_s^b\}$. By Corollary 6.1, we obtain that $P \not\models_{\mathsf{E}_a} \mathsf{Secret}(z)$ for some $z \in \{x_k, y_k\}$ or that $Q \not\models_{\mathsf{E}_b} \mathsf{Secret}(z)$ for some $z \in \{x_k, y_k, x_s\}$. In both cases, this contradicts the hypotheses. We thus deduce that $W \models_{\mathsf{E}} \mathsf{Secret}(x_s)$.

□

Example 6.6. Consider the process $\mathcal{P}_{DH}$ defined in Example 6.1, over the signature $\mathcal{F}_{DH}$. Consider any other protocol

$$Q = \nu y_k.(y_1 := y_k.Q_1 \mid y_2 := y_k.Q_2)$$

in which a participant Q_1 sends a secret x_s to the second participant using a shared key y_k. Assume that Q is defined over the signature $\mathcal{F}_{\text{enc}}$ and that $Q \models_{\mathsf{E}_{\text{enc}}} \mathsf{Secret}(x_s)$. Then the sequential composition of $\mathcal{P}_{DH}$ and Q, where $\mathcal{P}_{DH}$ is used to establish the shared key used in Q is defined by

$$W = \nu x_k.(P_1.Q_1 \mid P_2.Q_2)$$

Applying Theorem 6.2, $W \models_{\mathsf{E}_{DH} \cup \mathsf{E}_{\text{enc}}} \mathsf{Secret}(x_s)$, that is W does not leak x_s in the theory $\mathsf{E}_{DH} \cup \mathsf{E}_{\text{enc}}$.

6.6 Tagging

We have shown so far how to compose processes that use disjoint equational theories. However, this hypothesis is not always realistic since protocols often share usual cryptographic primitives such as symmetric encryption (for example, many protocols use AES). In this section, we show that protocols which do share common cryptographic primitives, such as encryption and hash functions, can also be securely composed in the same manner, as long as the two protocols are *tagged* differently.

Tagging is a syntactic transformation of a protocol in order to make it, for example, more resistant against attacks. Many ways to tag protocols have been proposed in different contexts, e.g. for composing protocols [93, 57] as discussed in introduction, to facilitate the analysis [28, 116] or to prevent type-flow attacks [94]. Typically, tagging a security protocol consists in appending a *tag* (e.g. a number, a nonce or a protocol identifier) to each plaintext before encrypting it and removing the tag after decryption. Tagging a protocol does not introduce additional attacks in the protocol, while preserving its communication goals.

We show that if two protocols share cryptographic primitives such as symmetric encryption and hash functions are tagged differently, then their composition is secure as long as the same problems detailed in Section 6.3.1 and Section 6.3.3 are avoided.

Our proof technique relies on our previous theorems, in that we show that an attack against the composition of two differently tagged protocols can be transformed into an attack where the protocols use disjoint encryption and hash functions. Therefore, tagging essentially enforces the disjointness of the two protocols.

6.6.1 Tagging a Protocol

We consider protocols over the signature $\mathcal{F}_{\text{enc},h} = \{\text{enc}, \text{dec}, h\}$ where enc and dec model respectively encryption and decryption and are of arity 2 and h models a hash function and is of arity 1. We also consider the signatures $\mathcal{F}^a_{\text{enc},h} = \{\text{enc}_a, \text{dec}_a, h_a\}$ and $\mathcal{F}^b_{\text{enc},h} = \{\text{enc}_b, \text{dec}_b, h_b\}$. We consider the associated equational theory $\mathsf{E}_{\text{enc}} = \{\text{dec}(\text{enc}(x, y), y) = x\}$ and the equational theories $\mathsf{E}^a_{\text{enc}} = \{\text{dec}_a(\text{enc}_a(x, y), y) = x\}$ and $\mathsf{E}^b_{\text{enc}} = \{\text{dec}_b(\text{enc}_b(x, y), y) = x\}$. The signatures $\mathcal{F}^a_{\text{enc},h}$ and $\mathcal{F}^b_{\text{enc},h}$, together with the associated equational theories $\mathsf{E}^a_{\text{enc}}$ and $\mathsf{E}^b_{\text{enc}}$, can be considered to intuitively model different implementations of the encryption/decryption/hash functions.

In order to define tagging, we first consider the signature renaming transformation $\_^d$ ($d \in \{a, b\}$) which assigns to a protocol P over $\mathcal{F}_{\text{enc},h}$ a protocol P^d ($d \in \{a, b\}$) over the signature $\mathcal{F}^d_{\text{enc},h}$ such that the two protocols are identical modulo bijective renaming of functions symbols (enc, dec and h are transformed into enc_d, dec_d and h_d, respectively, and this transformation is extended homomorphically to the entire protocol).

Example 6.7. If $P = \nu y.\mathbf{receive}(x).\mathbf{send}(\mathrm{dec}(x,y))$, then

$$P^a = \nu(y).\mathbf{receive}(x).\mathbf{send}(\mathrm{dec}_a(x,y)).$$

For every $d \in \{a,b\}$ we consider a *tagging function symbol* tag_d and an *untagging function symbol* untag_d contained in the signature $\mathcal{F}_d = \{\mathrm{tag}_d, \mathrm{untag}_d\}$ (where both function symbols have arity 1). The role of the tag_d function is to *tag* its argument with the tag d. Typically, this means appending d to the argument but the precise implementation of the tagging function does not need to be specified. The role of the untag_d function is to remove the tag. To model this interaction between tag_d and untag_d we consider the equational theories $\mathsf{E}_d = \{\mathrm{untag}_d(\mathrm{tag}_d(x)) = x\}$ (for $d \in \{a,b\}$).

If $A \in \{\mathbf{receive}(x), \mathbf{send}(t), \nu x, x := t, s = t\}$ is an atomic process over $\mathcal{F}^d_{\mathrm{enc},h}$ (with $d \in \{a,b\}$), we let $[|A|]$ be a trace over $\mathcal{F}_{\mathrm{enc},h} \cup \mathcal{F}_d$ denoting the d-tagged version of A, defined as follows:

$$\begin{array}{lcl}
[|\mathbf{receive}(x)|] & = & \mathbf{receive}(x) \\
[|\mathbf{send}(t)|] & = & \mathit{tests}^d(\mathcal{H}(t)) \cdot \mathbf{send}(\mathcal{H}(t)) \\
[|\nu x|] & = & \nu x \\
[|x := t|] & = & \mathit{tests}^d(\mathcal{H}(t)) \cdot x := \mathcal{H}(t) \\
[|s = t|] & = & \mathit{tests}^d(\mathcal{H}(s)) \cdot \mathit{tests}^d(\mathcal{H}(t)) \cdot x := \mathcal{H}(t)
\end{array}$$

where $\mathcal{H}(t)$ *tags* the term t with d as defined below:

$$\begin{array}{lcl}
\mathcal{H}(\mathrm{enc}_d(t_1,t_2)) & = & \mathrm{enc}(\mathrm{tag}_d(\mathcal{H}(t_1)), \mathcal{H}(t_2)) \\
\mathcal{H}(\mathrm{dec}_d(t_1,t_2)) & = & \mathrm{untag}_d(\mathrm{dec}(\mathcal{H}(t_1), \mathcal{H}(t_2))) \\
\mathcal{H}(h_d(t_1)) & = & h(\mathrm{tag}_d(\mathcal{H}(t_1))) \\
\mathcal{H}(a) & = & a \qquad \textit{where } a \textit{ is an atom}
\end{array}$$

and where $\mathit{tests}^d(t)$ is a sequence of tests which ensure that every decryption and every untagging performed by the protocol is successful:

$$\begin{array}{lcl}
\mathit{tests}^d(\mathrm{enc}(t_1,t_2)) & = & \mathit{tests}^d(t_1) \cdot \mathit{tests}^d(t_2) \\
\mathit{tests}^d(h(t_1)) & = & \mathit{tests}^d(t_1) \\
\mathit{tests}^d(\mathrm{tag}_d(t_1)) & = & \mathit{tests}^d(t_1) \\
\mathit{tests}^d(\mathrm{dec}(t_1,t_2)) & = & [\mathrm{enc}(\mathrm{dec}(t_1,t_2),t_2) = t_1] \cdot \\
 & & \mathit{tests}^d(t_1) \cdot \mathit{tests}^d(t_2) \\
\mathit{tests}^d(\mathrm{untag}_d(t_1)) & = & [\mathrm{tag}_d(\mathrm{untag}_d(t_1)) = t_1] \cdot \mathit{tests}^d(t_1) \\
\mathit{tests}^d(a) & = & 0 \qquad \textit{where } a \textit{ is an atom.}
\end{array}$$

The transformation $[|\_|]$ is extended homomorphically from atomic actions to traces.

Example 6.8. Continuing Example 6.7, we have that

$$\mathcal{H}(\mathrm{dec}_a(x,y)) = \mathrm{untag}_a(\mathrm{dec}(x,y)))$$

and that

$$\begin{array}{l}
\mathit{tests}^a(\mathrm{untag}_a(\mathrm{dec}(x,y))) = \\
\quad [\mathrm{tag}_a(\mathrm{untag}_a(\mathrm{dec}(x,y))) = \mathrm{dec}(x,y)]. \\
\quad [\mathrm{enc}(\mathrm{dec}(x,y),y) = x].
\end{array}$$

Finally, we have that

$$\begin{array}{ll}
[|P^a|] = & \nu y.\mathbf{receive}(x).[\mathrm{tag}_a(\mathrm{untag}_a(\mathrm{dec}(x,y))) = \mathrm{dec}(x,y)]. \\
 & [\mathrm{enc}(\mathrm{dec}(x,y),y) = x].\mathbf{send}(\mathrm{untag}_a(\mathrm{dec}(x,y))).
\end{array}$$

Note that before performing the decryption, the process $[|P^a|]$ verifies that it is decrypting a message which has been tagged with a.

6.6.2 Composing Tagged Protocols

Our next lemma shows why tagging two protocols is essentially the same as forcing them to use disjoint equational theories. It allows us to reduce the security problem for differently tagged processes to the security problem for processes which use disjoint equational theories. It states that if there is an attack on a composition of two differently tagged protocols $[|P^a|]$ and $[|Q^b|]$, there is an attack on the composition of the same protocols before tagging (P^a and Q^b), where the encryption and hash functions come from disjoint equational theories.

Lemma 6.14. *Let P and Q be traces over $\mathcal{F}_{\mathrm{enc},h}$. Let W be an arbitrary interleaving of P^a and Q^b and let $R = [|W|]$. If R reveals x then W reveals x.*

Furthermore, we have that if a protocol is secure then its c-tagged version is secure.

Lemma 6.15. *If P is a protocol over $\mathcal{F}_{\mathrm{enc},h}$, then $P \models_{\mathsf{E}_{\mathrm{enc}}} \mathsf{Secret}(x)$ implies $[|P^d|] \models_{\mathsf{E}_{\mathrm{enc}} \cup \mathsf{E}_d} \mathsf{Secret}(x)$.*

We can now state a generic theorem, in the spirit of Theorem 6.1, but for tagged protocols.

Let $P = P_1 \cdot \ldots \cdot P_n$ be a trace over $\mathcal{F}_{\mathrm{enc},h}$ with free variables $\{x_1, \ldots, x_p\}$ where P_i is an atomic process ($1 \leq i \leq n$). Let $Q = Q_1 \cdot \ldots \cdot Q_m$ be a trace over $\mathcal{F}_{\mathrm{enc},h}$ with free variables $\{y_1, \ldots, y_q\}$ where Q_i is an atomic process ($1 \leq i \leq n$).

Let $R = R_1 \ldots . R_{n+m}$ be a ground interleaving of $[|P_1^d|], \ldots, [|P_n^d|], [|Q_1^d|], \ldots, [|Q_m^d|]$. We consider R' a un-tagged copy of R where the shared variables of P and Q are duplicated as in Theorem 6.1 such that P and Q access disjoint variables. More precisely, let $R' = R'_1 \ldots . R'_{n+m}$ be such that:

1. $R'_i = P_j\{x \mapsto x^a\}$ if R_i is $[|P_j^a|]$ for some j and where x ranges over all variables in P_j
2. $R'_i = Q_j\{x \mapsto x^b\}$ if R_i is $[|Q_j^b|]$ for some j and where x ranges over all variables in Q_j

We consider an execution of the composition of $[|P^a|]$ and $[|Q^b|]$ in the equational theory $\mathsf{E}_{\mathrm{enc}} \cup \mathsf{E}_a \cup \mathsf{E}_b$.

$$(R, \emptyset, \emptyset) \xrightarrow{l_1, \ldots, l_k}{}^{*}_{\mathsf{E}_{\mathrm{enc}} \cup \mathsf{E}_a \cup \mathsf{E}_b} (S_0, \varphi_0, \sigma_0) \xrightarrow{\S} (S, \varphi, \sigma) \tag{6.7}$$

where $\S$ is $\mathbf{receive}(r_{k+1})$ for a recipe r_{k+1} if the last action was an input and the empty string otherwise.

Assume w.l.o.g. that $x_1, \ldots, x_{p'}$ are the variables from $\{x_1, \ldots, x_p\}$ which appear in $\mathcal{D}om(\sigma_0)$ and that $y_1, \ldots, y_{q'}$ are the variables from $\{y_1, \ldots, y_p\}$ which appear in $\mathcal{D}om(\sigma_0)$. This means that $x_1, \ldots, x_{p'}, y_1, \ldots, y_{q'}$ are the shared variables which were instantiated before the last action of the execution.

Let $\{z_i\}_{1 \leq i \leq p'}$ and $\{z_i\}_{p+1 \leq i \leq p+q'}$ be fresh variables such that

$$\begin{array}{ll} z_i = z_j \text{ iff } x_i\sigma_0 =_{\mathsf{E}} y_j\sigma_0 & \text{for all } 1 \leq i \leq p' \text{ and all } p+1 \leq j \leq p+q' \\ z_i = z_j \text{ iff } x_i\sigma_0 =_{\mathsf{E}} x_j\sigma_0 & \text{for all } 1 \leq i, j \leq p' \\ z_i = z_j \text{ iff } y_i\sigma_0 =_{\mathsf{E}} y_j\sigma_0 & \text{for all } p+1 \leq i, j \leq p+q' \end{array}$$

and let

$$\begin{array}{rcl} R'' & = & \nu\{z_i\}_{1 \leq i \leq p', p+1 \leq i \leq p+q'} \cdot \\ & & x_1^a := z_1 \ldots . x_{p'}^a := z_{p'} \cdot \\ & & y_1^b := z_{p+1} \ldots . y_{q'}^b := z_{p+q'} \cdot \\ & & R'. \end{array}$$

Then we have:

Theorem 6.4. *Assume $\varphi_0 \not\vdash x\sigma_0$ for any $x \in \{x_1, \ldots, x_{p'}, y_1, \ldots, y_{q'}\}$ and that $x_i\sigma_0 \neq_{\mathsf{E}_{\mathrm{enc}} \cup \mathsf{E}_a \cup \mathsf{E}_b} y_j\sigma_0$ for all $x_i, y_j \in \mathcal{D}om(\sigma_0)$. Then there exist S', φ', σ', $l'_1, \ldots, l'_k, \S'$ such that*

$$(R'', \emptyset, \emptyset) \xrightarrow{l'_1, \ldots, l'_k, \S'}{}^{*} (S', \varphi', \sigma') \tag{6.8}$$

is a run in the equational theory $\mathsf{E}_{\mathrm{enc}}$ and such that:

1. *if r is a minimal recipe such that $\varphi \vdash^r x\sigma_0$ for some $x \in \{x_1, \ldots, x_{p'}, y_1, \ldots, y_{q'}\}$, then $\varphi' \vdash^r x^d\sigma'$ for some $d \in \{a, b\}$.*
2. *otherwise, if $\varphi \vdash^r x\sigma$ for $x \in \mathcal{V}ar(P) \cup \mathcal{V}ar(Q) \setminus \{x_1, \ldots, x_{p'}, y_1, \ldots, y_{q'}\}$ then $\varphi' \vdash^r x^d\sigma'$ for some $d \in \{a, b\}$.*

Furthermore, if $x, y \in \mathcal{D}om(\sigma)$ are such that $x\sigma =_{\mathsf{E}} y\sigma$ and $x^d, y^d \in \mathcal{D}om(\sigma')$ for some $d \in \{a, b\}$, then $x^d\sigma' =_{\mathsf{E}} y^d\sigma'$.

In this tagged setting, the above theorem intuitively states that any trace on the tagged composition of two protocols can be transformed into a trace of the un-tagged composition, but where the two protocols no longer share secrets.

Example 6.9. We illustrate the above theorem with an example where the untagged composition of two protocols is not secure. However, using the theorem, we can conclude that the tagged composition is secure.

We consider the processes

$$P_1 = \nu x.\nu y.\mathbf{send}(\{x\}_y)$$

and

$$Q_2 = \nu z.\mathbf{send}(\{z\}_x).\mathbf{receive}(z').\mathbf{send}(\mathrm{dec}(z', y))$$

previously defined in Section 6.3. We have seen that $\nu x'.\nu y'.(x := x').(y := y').Q_2$ preserves the secrecy of z, while the sequential composition $P_1.Q_2$ (where P_1 is used to create the keys x and y) does not preserve the secrecy of z.

However, the sequential composition $[|P_1^a|].[|Q_2^b|]$ does preserve the secrecy of z by Theorem 6.4.

We can also use Theorem 6.4 to prove tagged variants of Theorem 6.2 and Theorem 6.3:

Theorem 6.5 (Tagged version of key-exchange theorem)**.** *Let $P = \nu k_1. \ldots .\nu k_n.(P_1 \mid P_2)$ and $Q = \nu k.((x_k := k).Q_1 \mid (y_k := k).Q_2)$ be processes over $\mathcal{F}_{\mathrm{enc},h}$ such that:*

1. *P_1 binds x_k and P_2 binds y_k*
2. *$fv(P) = \emptyset$, $fv(Q) = \emptyset$ and $\mathcal{V}ar(P) \cap \mathcal{V}ar(Q) = \{x_k, y_k\}$*
3. *$P \models_{\mathsf{E}_{\mathrm{enc}}} \mathsf{Secret}(x_k)$ and $P \models_{\mathsf{E}_{\mathrm{enc}}} \mathsf{Secret}(y_k)$*
4. *$Q \models_{\mathsf{E}_{\mathrm{enc}}} \mathsf{Secret}(x_k)$ and $Q \models_{\mathsf{E}_{\mathrm{enc}}} \mathsf{Secret}(y_k)$ and $Q \models_{\mathsf{E}_{\mathrm{enc}}} \mathsf{Secret}(x_s)$*

Then $W = \nu k_1. \ldots .\nu k_n.([|P_1^a|].[|Q_1^b|] \mid [|P_2^a|].[|Q_2^b|]) \models_{\mathsf{E}_{\mathrm{enc}} \cup \mathsf{E}_a \cup \mathsf{E}_b} \mathsf{Secret}(x_s)$.

Theorem 6.6 (Tagged version of secure channel theorem)**.** *Let $P = \nu k_1. \ldots .\nu k_n.!(P_1 \mid P_2)$ be a process over $\mathcal{F}_{\mathrm{enc},h}$ and let $Q =!(\nu k.(x_k := k.Q_1 \mid y_k := k.Q_2))$ be a process over $\mathcal{F}_{\mathrm{enc},h}$ such that:*

1. *P_1 binds x_k and P_2 binds y_k*
2. *$fv(P) = fv(Q) = \emptyset$*
3. *$P \models_{\mathsf{E}_a} \mathsf{Secret}(x_k)$ and $P \models_{\mathsf{E}_a} \mathsf{Secret}(y_k)$*
4. *$Q \models_{\mathsf{E}_b} \mathsf{Secret}(x_k)$ and $Q \models_{\mathsf{E}_b} \mathsf{Secret}(y_k)$ and $Q \models_{\mathsf{E}_b} \mathsf{Secret}(x_s)$*

Then $R = \nu k_1. \ldots .\nu k_n.!([|P_1^a|].[|Q_1^b|] \mid [|P_2^a|].[|Q_2^b|]) \models_{\mathsf{E}_{\mathrm{enc}} \cup \mathsf{E}_a \mathsf{E}_b} \mathsf{Secret}(x_s)$.

Theorems 6.5 and 6.6 allow us to securely compose key-exchange protocols which make use of symmetric encryption with protocols which use the exchanged keys, as long as the two protocols are tagged differently and if they obey the security requirements detailed above.

We have seen that Example 6.9 explains the need to tag the encryptions in order to obtain secure composition. One might think that tagging encryptions is sufficient to ensure the security of the composition and that it is not necessary to tag the hash function as well. Unfortunately, this is not true. We end this section on tagging by an example which illustrates why tagging is necessary for the hash function as well.

Example 6.10. ex:hashneedtag

We consider the processes

$$P = \nu x.\mathbf{send}(h(x))$$

and

$$Q = \nu z.\mathbf{receive}(y).[y = h(x)].\mathbf{send}(z).$$

We have that the protocol P does not reveal x. The protocol $\nu x.Q$ reveals neither z nor x. However, if P is used to instantiate the variable x for Q, we have that

$$P.Q \not\models \mathsf{Secret}(z).$$

By Theorem 6.4, we have however that the tagged composition does satisfy the secret of z:

$$[|P^a|].[|Q^b|] \models \mathsf{Secret}(z).$$

6.7 Conclusion and Future Work

6.7.1 Conclusion

We have shown in this chapter that protocols can be securely composed provided that they use primitives modeled by disjoint equational theories or provided that their only primitives are tagged encryptions or tagged hash functions.

Our result is a generic composition result: any trace leading to an attack on the composition of the two protocols is transformed into a trace that leads to an attack in one of the individual protocols, even if the two protocols share secrets such as keys. This allows us to securely perform several kinds of compositions. We can have secure parallel composition under shared secrets or we can have an asymmetric composition, where one of the protocols is used as a sub-protocol. As a matter of fact, our combination theorem could actually be used in any context where two protocols are arbitrarily interleaved and use shared data.

As an application, we have shown how our main composition theorem can be used in order to securely refine a protocol that uses pre-established keys or secure channels.

For the sake of simplicity, the only security property that we have considered is secrecy. We believe however that our result extends to general trace properties (e.g. authentication). This is because our trace transformation proof technique transforms any trace of the composition of two protocols under shared secrets (as long as a shared key is not revealed) into a trace on the composition under no shared secrets. This means that any violation of authentication in the composed protocol would be transformed into a violation of authentication on one of the individual protocols.

We have proven that primitives can be shared between the protocols provided they are tagged, in the case of symmetric encryption and hash. Even though we prove this only for symmetric encryption and hash functions, we believe that our technique can be extended to other usual cryptographic primitives such as asymmetric encryption and digital signature. Other cryptographic primitives can pose more problems (for example it is not obvious if/how the *eXclusive OR* can be tagged). It is an interesting open problem to give a generic definition of tagging and characterize which cryptographic primitives can be tagged.

6.7.2 Dishonest Participants

As we have shown in Section 6.3.1, one of the hypothesis that we use in order for our composition theorem to work is that the two processes involved do not leak shared keys. Unfortunately, this makes our composition theorem unable to cope with dishonest participants because dishonest participants leak *everything* to the intruder (in some sense, they are the intruder). Let

$$P = \nu k.(P_1 \mid P_2)$$

is a key-exchange protocol with two participants P_1 and P_2, with P_1 storing the exchanged key in variable x_k and P_2 storing the exchanged key in variable y_k. Let

$$Q = \nu x.((x_k := x).Q_1 \mid (y_k := y).Q_2)$$

be a protocol which uses the exchanged keys x_k, y_k (assumed to be freshly chosen in Q). We have shown that the sequential composition of P and Q is secure as long as P and Q are over disjoint equational theories and do not leak the shared keys. However, assuming for example that the participant playing the role P_2 and Q_2 is dishonest, all of the keys involved are leaked and therefore we cannot apply our composition theorem.

This is the main limitation of our work. Therefore the most important direction for future work is to investigate which additional conditions are needed to handle composition in the presence of dishonest participants. It is clear that the assumption that shared keys are not revealed needs to be weakened, but there needs to be a condition that replaces it.

One of the main ingredients of our composition theorem is that a sequence $l_1, \ldots, l_k$ of intruder actions leading to an attack in an interleaving of two traces leads to the same attack when the traces no longer share anything. In the case of dishonest participants, this is no longer the case and the sequences of intruder actions needs to be slightly altered as shown in the following example:

Example 6.11. Let $P = \nu z.\mathbf{send}(z)$ and $Q = \mathbf{send}(z).\mathbf{receive}(x_1).\mathbf{receive}(x_2).[x1 = x2].\mathbf{send}(s)$. The variable z is supposed to be a key generated by P for Q and s is a private name. However a "participant" detaining the key z is dishonest and leaks it as soon as possible on the network. We have that

$$(P.Q, \emptyset, \emptyset) \xrightarrow{\mathbf{receive}(w_1),\mathbf{receive}(w_2)}{}^{*} (0, \varphi, \sigma)$$

for some substitution σ and with $\varphi = \{w_1 \mapsto k_1, w_2 \mapsto k_1, w_3 \mapsto s\}$ where k_1 is a fresh name. In the spirit of our composition theorem, we would like that the same intruder actions $\mathbf{receive}(w_1), \mathbf{receive}(w_2)$ to work in the case where P and Q used disjoint variables. However, we have that

$$P.\nu z'.(Q\{z \mapsto z'\}) = \nu z.\mathbf{send}(z).\nu z'.\mathbf{send}(z').\mathbf{receive}(x_1).\mathbf{receive}(x_2).[x1 = x2].\mathbf{send}(s)$$

blocks at the test $[x_1 = x_2]$ since x_1 will be bound to k_1 and x_2 bound to k_2 for fresh names k_1, k_2. Therefore the same intruder actions that were used to get $P.Q$ to output s will not work to get $P.\nu z'.(Q\{z \mapsto z'\})$ to output s. In this case, it is easy to see that it is possible to easily tweak the intruder actions (i.e. change $\xrightarrow{\mathbf{receive}(w_1),\mathbf{receive}(w_2)}{}^{*}$ to $\xrightarrow{\mathbf{receive}(w_1),\mathbf{receive}(w_1)}{}^{*}$) in order to get the disjoint protocol

$$P.\nu z'.(Q\{z \mapsto z'\}) \xrightarrow{\mathbf{receive}(w_1),\mathbf{receive}(w_1)}{}^{*} (0, \{w_1 \mapsto k_1, w_2 \mapsto k_2, w_3 \mapsto s\}, \sigma')$$

to output s. However, finding a generic way to change the intruder actions in order to obtain the same attack is an open problem and would allow to have a theorem for composition which is sound even in the presence of dishonest participants.

6.7.3 Directions for Future Work

A direction for future work is to investigate if composition with disjoint equational theories preserves *trace equivalence*, as defined e.g. in Chapter 5 and more generally other behavioral equivalences which can be used to reason about security properties such as anonymity. As in our composition theorem we reason about recipes, there is hope that the our result carries over to equivalence properties.

We have shown that protocols which have disjoint equational theories compose well and protocols that are tagged compose well. A possible direction for future work is a straightforward generalization of our composition result which combines the equational composition result and the tagging composition result as follows: show that the composition of two protocols which make use of some common cryptographic primitives (and possibly other primitives which are individual to each protocol) compose well as longs as the common primitives are tagged (and the primitives which are particular to each protocol are kept intact).

Also, certain primitives which seem harmless enough that they may be shared without tagging them. For example, the concatenation defined through the equational theory:

$$\mathrm{fst}(\mathrm{pair}(x,y)) = x \qquad \mathrm{snd}(\mathrm{pair}(x,y)) = y.$$

is a candidate. However, let us consider the processes

$$P = \nu x.\nu y.z := \mathrm{pair}(x,y)$$

$$Q = \nu k.\mathbf{send}(\mathrm{enc}(z,k)).\mathbf{receive}(y).\mathbf{send}(\mathrm{pair}(\mathrm{fst}(\mathrm{dec}(y,k)),\mathrm{snd}(\mathrm{dec}(y,k)))).$$

The process $\nu z.Q$ does not reveal z. However, if the generation of z is handled by P, we have that $P \cdot Q$ does reveal z. This is because Q was only verified secure when z is instantiated to a name. We are trying to prove that the equational theory of pair can be safely shared between two protocols as long as neither of the protocols instantiates a shared key to a pair.

Finally, many relevant equational theories are not so easy to tag. In particular, tagging *exclusive or* is particularly difficult. Finding a way to securely compose two protocols which both make use of this primitive (the *exclusive or*) is a challenging open problem.

Chapter 7

Conclusion and Perspectives

We have presented methods that allow us to increase our confidence in the security of cryptographic protocols with applications to e-voting protocols. We have seen that e-voting protocols are different from other previously analyzed protocols in two aspects:

1. they employ novel cryptographic primitives such as blind signatures, trapdoor commitment and homomorphic encryption and
2. the security properties desirable of them are *fundamentally different* from confidentiality and authentication, which were the most studied security properties.

In this thesis, we show that automated verification of security protocols with respect to equivalence properties is feasible and that protocols making use of arbitrary disjoint cryptographic primitives compose well with respect to confidentiality. Chapters 3 to 5 consist of techniques for the automated verification of equivalence properties and Chapter 6 consists of the composition result. We believe that both directions (automated verification and compositionality) provide fruitful ground for further research.

In Chapter 3 we have introduced and motivated the notion of strongly complete set of variants. This notion is used as a base brick in Chapter 5 for the automated procedure for proving equivalence properties. Strongly complete sets of variants allow to get rid of the equational theory modeling the properties of the cryptographic primitives and solve the resulting problem in the free term algebra.

Therefore, in order to be able to handle a greater variety of cryptographic primitives, a first step is to prove that the equational theory modeling the new primitives still has the strong finite variant property. However, for cryptographic primitives such as XOR, this cannot be the case because they contain associative-commutative (AC) function symbols. Therefore no convergent term rewriting system can model XOR. Instead, the notion of strongly complete set of variants should be extended to a convergent rewrite system R *modulo an equational theory* E. The primitive XOR could then be split into two parts: E would model the AC property of XOR while R would model that 0 is a neutral element and that by XORing a value with itself we obtain the neutral element 0.

As it might be cumbersome to establish the strong finite variant property, it would be nice to obtain a combination result: assuming two disjoint rewrite systems $(\mathcal{F}_a, \mathsf{R}_a)$ and $(\mathcal{F}_b, \mathsf{R}_b)$ have the strong finite variant property (eventually modulo an equational theory), does $(\mathcal{F}_a \cup \mathcal{F}_b, \mathsf{R}_a \cup \mathsf{R}_b)$ also have the SFVP? Are strongly complete sets of variants for mixed terms (with function symbols from $\mathcal{F}_a$ and from $\mathcal{F}_b$) effectively constructible from strongly complete sets of variants of pure terms (with function symbols only from $\mathcal{F}_a$ or only from $\mathcal{F}_b$)?

In Chapter 4 we have presented a decision procedure for static equivalence. The procedure is sound and complete for any convergent equational theory, but termination currently needs to be established on a case-by-case basis. We have shown termination for several (classes of) equational

theories, but it would be interesting to obtain an *exact* characterization of the class of equational theories for which the saturation procedure terminates.

The procedure we present does not terminate for an equational theory modeling re-encryption, a cryptographic primitives which can be used in designated verifier proofs of re-encryption as in [72]. The reason for non-termination is illustrated in Example 4.11 using a toy equational theory which presents the same difficulties as re-encryption. Extending the procedure to handle this equational theory in a generic fashion is an interesting open problem and it would allow (partial) automation of security proof for electronic voting protocols which make use of designated verifier proofs.

Another direction for future work would be to extend the procedure to handle cryptographic primitives such as XOR which cannot be modeled by a convergent rewrite system. To handle such theories, an idea is to keep the current structure of the saturation procedure but to perform matching and unification modulo AC. Soundness, completeness and termination for this approach should be investigated.

In Chapter 5 we have presented a procedure for verifying trace equivalence for determinate processes. This procedure has allowed us to obtain the first automated proof of vote privacy for the electronic voting protocol FOO. However, the procedure presents several limitations which should be lifted:

1. We have conjectured that the saturation procedure always terminates for subterm convergent rewrite systems. Due to the highly complex form of Horn clauses generated during saturation, we have not been able to prove this. However, this should be further investigated. Proving termination would imply a strict theoretical extension of previous results [44, 19] from traces to determinate processes.

2. Currently, our procedure only handles *bounded* processes. This restriction can be lifted by allowing sound abstractions in the style of [25]. Abstractions would allow to obtain proofs of equivalence for unbounded protocols, but with the possibility of false attacks.

3. Currently in our process algebra only tests of the form $[s = t]$ are permitted. If a test fails, the process blocks. However, some protocols need *else branches* in order to perform a operation if a test *does not succeed*. An idea for handling *else* branches is to use strongly complete set of variants in order to build complete sets of disunifiers for some notion of completeness. This complete sets of disunifiers would then allow to describe exactly for which instantiation of the terms the *else* branch would be taken.

   Handling *else* branches would allow to obtain security proofs for protocols for the private authentication protocol in [4] or the e-passport protocol in [9].

4. Even assuming that the saturation procedure terminates, our procedure can establish a proof of equivalence or provide a counter-example for determinate processes only. For processes which are not known to be determinate, we provide a sound proof method to establish equivalence; however, if the proof method fails, we do not know if the processes are equivalent or not.

   These situations are caused by negative tests as illustrated bellow. Consider the following traces:

   $$T_1 = \mathbf{send}(c, n).\mathbf{send}(c, m)$$
   $$T_2 = \mathbf{send}(c, n).\mathbf{send}(c, n)$$
   $$S = \mathbf{send}(c, k).\mathbf{send}(c, k),$$

   where $n, m, k \in \mathcal{N}$ are private names. The process $\{T_1, T_2\}$ is not trace-equivalent to the process S (but we cannot prove this using our procedure) intuitively because the intruder can check if the first output is different from the second output and in this case conclude that he is dealing with $\{T_1, T_2\}$ instead of S. This "negative test" is exactly what the current procedure does not handle; instead it only captures positive tests using the predicate i. One idea is to extend the saturation procedure to include a predicate ni which states that two

recipes are not the same. This would allow the procedure to directly handle non-determinate processes as well.

5. As we have already pointed out, our procedure works only for convergent rewrite systems but there are cryptographic primitives such as XOR which cannot be modeled by such rewrite systems. Therefore it is necessary to work with a convergent rewrite system modulo a theory of associativity-commutativity (AC). It would be interesting to extend the procedure to handle strongly complete sets of variants, matching and unification modulo AC. The result should be again proven sound and complete.

6. As the equational theory gets more complex, the chances that a given method automatically proves an equivalence between processes diminish. Therefore it is desirable to obtain combination results: if the equational theory is split into two parts and for each part a procedure for proving equivalence is known, can the two procedures be combined into a procedure for verifying equivalence for the larger equational theory?

7. The equivalences that our procedure can prove are in the Dolev-Yao model as opposed to the more realistic computational model. However, there is a result [52] which links equivalence in the Dolev-Yao model to equivalence in the computational model. As a more distant perspective, bridging the gap between the equivalences that our procedure proves and the equivalences of the computational model is a desirable goal. This could be achieved for example either by a computational soundness result or by modifying the tool so that to fit the computational model better or both.

8. Other perspectives for further research are to extend our procedure to check if a protocol implements an ideal functionality as in [70] or to check more complex properties such as individual or universal verifiability of voting protocols as formalized in [99].

In Chapter 6, we have shown that protocols which use disjoint cryptographic primitives can be securely composed with respect to confidentiality of data if they do not reveal or reuse shared secrets.

We have seen that due to our hypothesis (that shared secrets are not revealed), the main limitation is that the composition theorem cannot be applied for dishonest participants (because they reveal secrets to the adversary as soon as possible). Therefore the most important extension of the composition result in Chapter 6 is to allow composition even in the presence of dishonest users. We have shown (Section 6.7.2) that our theorem does not hold for dishonest participants. Therefore, in order to obtain a composition result in this context, the hypothesis needs to be altered.

Another hypothesis that we make is that the protocols use disjoint primitives. This is not always the case, as both might use, for example, the same encryption algorithms. We mitigate this by proving (for a limited set of primitives) that composition still holds as long as the primitives are *tagged*. Enlarging the set of primitives for which this is shown is the subject of future work. Furthermore, it would be nice to obtain a composition result in which the primitives which are shared between the two protocols are tagged, while all the other primitives (which are used by one of the two protocols) remain as they are.

Obtaining composition results for indistinguishability properties such as trace equivalence is another natural extension. This would be particularly useful in the case of e-voting protocols, where it could be used to show that vote privacy is preserved even in the presence of other protocols. Other minor extension is to prove the same composition result in the presence of *else* branches.

More generally, we would like to apply the techniques that we have developed in this thesis to other contexts. We would like to prove other equivalence-based security properties of electronic voting protocols such as coercion-resistance or receipt-freeness [72]. Furthermore, the same techniques for verifying trace equivalence could be used in a different context to prove *unlinkability* [9].

Unlinkability is a privacy property which intuitively means that an attacker cannot *trace* a participant by *linking together* several uses of a protocol such as the protocol run by the participant during the checking of his electronic passport. Equivalence of processes seems to be well-suited for the verification of privacy properties like anonymity, unlinkability and vote privacy and we would like to investigate other potential applications of trace equivalence such as for verifying privacy in web services or anonymity in routing protocols.

Bibliography

[1] M. Abadi and V. Cortier. Deciding knowledge in security protocols under equational theories. In *ICALP*, pages 46–58, 2004.

[2] M. Abadi and V. Cortier. Deciding knowledge in security protocols under equational theories. *Theoretical Computer Science*, 387(1-2):2–32, 2006.

[3] M. Abadi and C. Fournet. Mobile values, new names, and secure communication. In *Proc. 28th ACM Symposium on Principles of Programming Languages (POPL'01)*. ACM, 2001.

[4] M. Abadi and C. Fournet. Private authentication. *Theor. Comput. Sci.*, 322(3):427–476, 2004.

[5] M. Abadi and A. D. Gordon. A calculus for cryptographic protocols: The spi calculus. *Information and Computation*, 148:36–47, 1999.

[6] R. Affeldt and H. Comon-Lundh. Verification of security protocols with a bounded number of sessions based on resolution for rigid variables. In *Formal to Practical Security*, volume 5458 of *LNCS, State-of-the-Art Survey*, pages 1–20. Springer, 2009.

[7] 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.

[8] S. Anantharaman, P. Narendran, and M. Rusinowitch. Intruders with caps. In *Proc. 18th International Conference on Term Rewriting and Applications (RTA'07)*, volume 4533 of *LNCS*. Springer, 2007.

[9] M. Arapinis, T. Chothia, E. Ritter, and M. D. Ryan. Analysing unlinkability and anonymity using the applied pi calculus. In *Proc. 23rd Computer Security Foundations Symposium (CSF'10)*, pages 107–121. IEEE Comp. Soc. Press, 2010.

[10] M. Arapinis, S. Delaune, and S. Kremer. From one session to many: Dynamic tags for security protocols. In I. Cervesato, H. Veith, and A. Voronkov, editors, *Proceedings of the 15th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'08)*, volume 5330 of *Lecture Notes in Artificial Intelligence*, pages 128–142, Doha, Qatar, Nov. 2008. Springer.

[11] A. Armando, R. Carbone, L. Compagna, J. Cuellar, and L. Tobarra. Formal analysis of saml 2.0 web browser single sign-on: breaking the saml-based single sign-on for google apps. In *Proceedings of the 6th ACM workshop on Formal methods in security engineering*, FMSE '08, pages 1–10, New York, NY, USA, 2008. ACM.

[12] A. Armando et al. The AVISPA Tool for the automated validation of internet security protocols and applications. In *Proc. 17th Int. Conference on Computer Aided Verification (CAV'05)*, volume 3576 of *LNCS*, pages 281–285. Springer, 2005.

[13] M. Arnaud, V. Cortier, and S. Delaune. Combining algorithms for deciding knowledge in security protocols. In F. Wolter, editor, *Proceedings of the 6th International Symposium on Frontiers of Combining Systems (FroCoS'07)*, volume 4720 of *Lecture Notes in Artificial Intelligence*, pages 103–117, Liverpool, UK, Sept. 2007. Springer.

[14] F. Baader and T. Nipkow. *Term rewriting and all that.* Cambridge University Press, New York, NY, USA, 1998.

[15] F. Baader and W. Snyder. Unification theory. In *Handbook of Automated Reasoning*, pages 445–532. Elsevier and MIT Press, 2001.

[16] M. Backes, C. Hritcu, and M. Maffei. Automated verification of remote electronic voting protocols in the applied pi-calculus. In *Proceedings of the 2008 21st IEEE Computer Security Foundations Symposium*, pages 195–209, Washington, DC, USA, 2008. IEEE Computer Society.

[17] M. Backes, C. Hritcu, and M. Maffei. Automated verification of remote electronic voting protocols in the applied pi-calculus. In *Proc. 21st IEEE Computer Security Foundations Symposium (CSF'08)*, 2008.

[18] M. Backes, M. Maffei, and D. Unruh. Zero-knowledge in the applied pi-calculus and automated verification of the direct anonymous attestation protocol. In *IEEE Symposium on Security and Privacy*, pages 202–215, 2008.

[19] M. Baudet. Deciding security of protocols against off-line guessing attacks. In *12th ACM Conference on Computer and Communications Security (CCS'05)*, 2005.

[20] M. Baudet. YAPA (Yet Another Protocol Analyzer), 2008. `http://www.lsv.ens-cachan.fr/~baudet/yapa/index.html`.

[21] M. Baudet, V. Cortier, and S. Delaune. YAPA: A generic tool for computing intruder knowledge. In R. Treinen, editor, *Proceedings of the 20th International Conference on Rewriting Techniques and Applications (RTA'09)*, volume 5595 of *Lecture Notes in Computer Science*, pages 148–163, Brasília, Brazil, June-July 2009. Springer.

[22] M. Baudet, V. Cortier, and S. Delaune. YAPA: A generic tool for computing intruder knowledge. Research Report LSV-09-03, Laboratoire Spécification et Vérification, ENS Cachan, France, Feb. 2009. 26 pages.

[23] S. M. Bellovin and M. Merritt. Encrypted key exchange: Password-based protocols secure against dictionary attacks. In *Proc. IEEE Computer Society Symposium on Research in Security and Privacy*, pages 72–84, Oakland, CA, May 1992.

[24] M. Berrima, N. Ben Rajeb, and V. Cortier. Deciding knowledge in security protocols under some e-voting theories. Research Report RR-6903, INRIA, April 2009.

[25] B. Blanchet. An Efficient Cryptographic Protocol Verifier Based on Prolog Rules. In *14th Computer Security Foundations Workshop (CSFW'01)*, pages 82–96. IEEE Comp. Soc. Press, 2001.

[26] B. Blanchet. Automatic proof of strong secrecy for security protocols. In *IEEE Symposium on Security and Privacy*, pages 86–100, 2004.

[27] B. Blanchet, M. Abadi, and C. Fournet. Automated Verification of Selected Equivalences for Security Protocols. In *Symposium on Logic in Computer Science*, pages 331–340. IEEE Comp. Soc. Press, 2005.

[28] B. Blanchet and A. Podelski. Verification of cryptographic protocols: Tagging enforces termination. In A. Gordon, editor, *Foundations of Software Science and Computation Structures (FoSSaCS'03)*, volume 2620 of *LNCS*, April 2003.

[29] J. Borgström. *Equivalences and Calculi for Formal Verifiation of Cryptographic Protocols*. Phd thesis, EPFL, Switzerland, 2008.

[30] J. Borgström, S. Briais, and U. Nestmann. Symbolic bisimulation in the spi calculus. In *Proc. 15th Int. Conference on Concurrency Theory*, volume 3170 of *LNCS*, pages 161–176. Springer, 2004.

[31] M. Bruso, K. Chatzikokolakis, and J. den Hartog. Analysing unlinkability and anonymity using the applied pi calculus. In *Proc. 23rd Computer Security Foundations Symposium (CSF'10)*, pages 107–121. IEEE Comp. Soc. Press, 2010.

[32] S. Bursuc and H. Comon-Lundh. Protocol security and algebraic properties: Decision results for a bounded number of sessions. In *RTA*, pages 133–147, 2009.

[33] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *IEEE Symposium on Foundations of Computer Science*, pages 136–145, 2001.

[34] R. Canetti and T. Rabin. Universal composition with joint state. In *CRYPTO*, pages 265–281, 2003.

[35] R. Chadha, Ş. Ciobâcă, and S. Kremer. Automated verification of equivalence properties of cryptographic protocols. In *21st European Symposium on Programming (ESOP)*, 2012.

[36] D. Chaum. Blind signatures for untraceable payments. In *Advances in Cryptology: Proceedings of CRYPTO '82*, pages 199–203, 1982.

[37] V. Cheval, H. Comon-Lundh, and S. Delaune. Automating security analysis: symbolic equivalence of constraint systems. In J. Giesl and R. Haehnle, editors, *Proceedings of the 5th International Joint Conference on Automated Reasoning (IJCAR'10)*, volume 6173 of *Lecture Notes in Artificial Intelligence*, pages 412–426, Edinburgh, Scotland, UK, July 2010. Springer-Verlag.

[38] V. Cheval, H. Comon-Lundh, and S. Delaune. Trace equivalence decision: Negative tests and non-determinism. In *Proceedings of the 18th ACM Conference on Computer and Communications Security (CCS'11)*, Chicago, Illinois, USA, Oct. 2011. ACM Press.

[39] Y. Chevalier. *Résolution de problèmes d' accessibilité pour la compilation et la validation de protocoles cryptographiques*. PhD thesis, Université Henri Poincaré, Nancy (France), 2003.

[40] Y. Chevalier and M. Kourjieh. Key substitution in the symbolic analysis of cryptographic protocols. In *Proc. 27th International Conference on Foundations of Software Technology and Theoretical Computer Science (FST&TCS'07)*, pages 121–132, 2007.

[41] Y. Chevalier, R. Küsters, M. Rusinowitch, and M. Turuani. Deciding the security of protocols with Diffie-Hellman exponentiation and product in exponents. In *Proceedings of the 23rd Conference on Foundations of Software Technology and Theoretical Computer Science (FST&TCS'03)*, volume 2914 of *Lecture Notes in Computer Science*, pages 124–135, Mumbai (India), 2003. Springer-Verlag.

[42] Y. Chevalier, R. Küsters, M. Rusinowitch, and M. Turuani. An NP decision procedure for protocol insecurity with XOR. In *18th IEEE Symposium on Logic in Computer Science (LICS'03)*. IEEE Comp. Soc. Press, 2003.

[43] Y. Chevalier and M. Rusinowitch. Combining intruder theories. In *32nd International Colloquium on Automata, Languages and Programming (ICALP'05)*, volume 3580 of *LNCS*, pages 639–651. Springer, 2005.

[44] Y. Chevalier and M. Rusinowitch. Decidability of equivalence of symbolic derivations. *Journal of Automated Reasoning*, 2010.

[45] Ş. Ciobâcă. KiSs, 2009. http://www.lsv.ens-cachan.fr/~ciobaca/kiss.

[46] Ş. Ciobâcă. AKiSs, 2011. http://www.lsv.ens-cachan.fr/~ciobaca/akiss.

[47] Ş. Ciobâcă. Computing finite variants for subterm convergent rewrite systems. In F. Baader, editor, *Proceedings of the 25th International Workshop on Unification (UNIF'11)*, Wrocław, Poland, July 2011.

[48] Ş. Ciobâcă. SubVariant, 2011. http://www.lsv.ens-cachan.fr/~ciobaca/subvariant.

[49] Ş. Ciobâcă and V. Cortier. Protocol composition for arbitrary primitives. In *Proceedings of the 23rd IEEE Computer Security Foundations Symposium (CSF'10)*, pages 322–336, Edinburgh, Scotland, UK, July 2010. IEEE Computer Society Press.

[50] Ş. Ciobâcă, S. Delaune, and S. Kremer. Computing knowledge in security protocols under convergent equational theories. In R. Schmidt, editor, *Proceedings of the 22nd International Conference on Automated Deduction (CADE'09)*, Lecture Notes in Artificial Intelligence, pages 355–370, Montreal, Canada, Aug. 2009. Springer.

[51] Ş. Ciobâcă, S. Delaune, and S. Kremer. Computing knowledge in security protocols under convergent equational theories. *Journal of Automated Reasoning*, 2011.

[52] H. Comon-Lundh and V. Cortier. Computational soundness of observational equivalence. In *Proceedings of the 15th ACM conference on Computer and communications security*, CCS '08, pages 109–118, New York, NY, USA, 2008. ACM.

[53] H. Comon-Lundh and S. Delaune. The finite variant property: How to get rid of some algebraic properties. In J. Giesl, editor, *Proceedings of the 16th International Conference on Rewriting Techniques and Applications (RTA'05)*, volume 3467 of *Lecture Notes in Computer Science*, pages 294–307, Nara, Japan, Apr. 2005. Springer.

[54] H. Comon-Lundh and V. Shmatikov. Intruder deductions, constraint solving and insecurity decision in presence of exclusive or. In *18th IEEE Symposium on Logic in Computer Science (LICS'03)*. IEEE Comp. Soc. Press, 2003.

[55] B. Conchinha, D. A. Basin, and C. Caleiro. Fast: An efficient decision procedure for deduction and static equivalence. In *RTA*, pages 11–20, 2011.

[56] R. Corin, J. Doumen, and S. Etalle. Analysing password protocol security against off-line dictionary attacks. In *Proc. 2nd International Workshop on Security Issues with Petri Nets and other Computational Models (WISP'04)*, ENTCS, 2004.

[57] V. Cortier, J. Delaitre, and S. Delaune. Safely composing security protocols. In V. Arvind and S. Prasad, editors, *Proceedings of the 27th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'07)*, volume 4855 of *Lecture Notes in Computer Science*, pages 352–363, New Delhi, India, Dec. 2007. Springer.

[58] V. Cortier and S. Delaune. Deciding knowledge in security protocols for monoidal equational theories. In *Proc. 14th Int. Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'07)*, LNAI. Springer, 2007.

[59] V. Cortier and S. Delaune. A method for proving observational equivalence. In *Proceedings of the 22nd IEEE Computer Security Foundations Symposium (CSF'09)*, pages 266–276, Port Jefferson, NY, USA, July 2009. IEEE Computer Society Press.

[60] V. Cortier and S. Delaune. Safely composing security protocols. *Formal Methods in System Design*, 34(1):1–36, Feb. 2009.

[61] 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.

[62] V. Cortier, B. Warinschi, and E. Zălinescu. Synthesizing secure protocols. In *12th European Symposium On Research In Computer Security (ESORICS'07)*, volume 4734 of *Lecture Notes in Computer Science*, pages 406–421. Springer, 2007.

[63] C. Cremers. The Scyther Tool: Verification, falsification, and analysis of security protocols. In *Computer Aided Verification, 20th International Conference, CAV 2008, Princeton, USA, Proc.*, volume 5123/2008 of *Lecture Notes in Computer Science*, pages 414–418. Springer, 2008.

[64] C. J. Cremers. Unbounded verification, falsification, and characterization of security protocols by pattern refinement. In *CCS '08: Proceedings of the 15th ACM conference on Computer and communications security*, pages 119–128, New York, NY, USA, 2008. ACM.

[65] M. Dahl, S. Delaune, and G. Steel. Formal analysis of privacy for vehicular mix-zones. In *Proc. 15th European Symposium on Research in Computer Security (ESORICS'10)*, volume 6345 of *LNCS*, pages 55–70. Springer, 2010.

[66] M. Dahl, S. Delaune, and G. Steel. Formal analysis of privacy for anonymous location based services. In *Proc. Workshop on Theory of Security and Applications (TOSCA'11)*, 2011.

[67] A. Datta, A. Derek, J. C. Mitchell, and A. Roy. Protocol composition logic (PCL). *Electron. Notes Theor. Comput. Sci.*, 172:311–358, 2007.

[68] S. Delaune. *Vérification des protocoles cryptographiques et propriétés algébriques*. Thèse de doctorat, Laboratoire Spécification et Vérification, ENS Cachan, France, June 2006.

[69] S. Delaune and F. Jacquemard. A decision procedure for the verification of security protocols with explicit destructors. In *Proceedings of the 11th ACM conference on Computer and communications security*, CCS '04, pages 278–287, New York, NY, USA, 2004. ACM.

[70] S. Delaune, S. Kremer, and O. Pereira. Simulation based security in the applied pi calculus. In R. Kannan and K. Narayan Kumar, editors, *Proceedings of the 29th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'09)*, volume 4 of *Leibniz International Proceedings in Informatics*, pages 169–180, Kanpur, India, Dec. 2009. Leibniz-Zentrum für Informatik.

[71] S. Delaune, S. Kremer, and M. D. Ryan. Composition of password-based protocols. In *Proceedings of the 21st IEEE Computer Security Foundations Symposium (CSF'08)*, pages 239–251, Pittsburgh, PA, USA, June 2008. IEEE Computer Society Press.

[72] S. Delaune, S. Kremer, and M. D. Ryan. Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security*, 17(4):435–487, July 2009.

[73] S. Delaune, S. Kremer, and M. D. Ryan. Symbolic bisimulation for the applied pi calculus. *Journal of Computer Security*, 18(2):317–377, Mar. 2010.

[74] S. Delaune, H. Lin, and Ch. Lynch. Protocol verification via rigid/flexible resolution. In N. Dershowitz and A. Voronkov, editors, *Proceedings of the 14th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'07)*, volume 4790 of *Lecture Notes in Artificial Intelligence*, pages 242–256, Yerevan, Armenia, Oct. 2007. Springer.

[75] H. Delfs and H. Knebl. *Introduction to cryptography: principles and applications*. Springer-Verlag New York, Inc., New York, NY, USA, 2001.

[76] D. E. Denning and G. M. Sacco. Timestamps in key distribution protocols. *Commun. ACM*, 24:533–536, August 1981.

[77] W. Diffie and M. Hellman. New directions in cryptography. *Information Theory, IEEE Transactions on*, 22(6):644–654, Jan. 2003.

[78] D. Dolev and A. C. Yao. On the security of public key protocols. *Foundations of Computer Science, Annual IEEE Symposium on*, 0:350–357, 1981.

[79] L. Durante, R. Sisto, and A. Valenzano. Automatic testing equivalence verification of spi calculus specifications. *ACM Transactions on Software Engineering and Methodology*, 12(2):222–284, 2003.

[80] N. A. Durgin, P. Lincoln, and J. C. Mitchell. Multiset rewriting and the complexity of bounded security protocols. *Journal of Computer Security*, 12(2):247–311, 2004.

[81] J. Engelfriet. Determinacy - (observation equivalence = trace equivalence). *Theor. Comput. Sci.*, 36:21–25, 1985.

[82] S. Escobar, C. Meadows, and J. Meseguer. Maude-NPA: Cryptographic protocol analysis modulo equational properties. In *Foundations of Security Analysis and Design V*, volume 5705 of *LNCS*, pages 1–50. Springer, 2009.

[83] S. Escobar, J. Meseguer, and R. Sasse. Effectively checking the finite variant property. In *RTA*, pages 79–93, 2008.

[84] S. Escobar, J. Meseguer, and R. Sasse. Variant narrowing and equational unification. *Electron. Notes Theor. Comput. Sci.*, 238:103–119, June 2009.

[85] F. J. T. Fábrega. Strand spaces: proving security protocols correct. *J. Comput. Secur.*, 7(2-3):191–230, 1999.

[86] FIPS. *Advanced Encryption Standard (AES)*, Nov. 2001.

[87] A. Fujioka, T. Okamoto, and K. Ohta. A practical secret voting scheme for large scale elections. In *Proceedings of the Workshop on the Theory and Application of Cryptographic Techniques: Advances in Cryptology*, ASIACRYPT '92, pages 244–251, London, UK, 1993. Springer-Verlag.

[88] S. Goldwasser and S. Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.

[89] J. Goubault-Larrecq. Deciding $\mathcal{H}_1$ by resolution. *Information Processing Letters*, 95(3):401–408, Aug. 2005.

[90] J. Goubault-Larrecq and K. N. Verma. Alternating two-way AC-tree automata. Research Report LSV-02-11, Laboratoire Spécification et Vérification, ENS Cachan, France, Sept. 2002. 21 pages.

[91] T. Groß and S. Mödersheim. Vertical protocol composition. In *CSF*, pages 235–250, 2011.

[92] J. D. Guttman. Cryptographic protocol composition via the authentication tests. In *Foundations of Software Science and Computation Structures (FOSSACS'09)*, Lecture Notes in Computer Science, March 2009.

[93] J. D. Guttman and F. J. Thayer. Protocol independence through disjoint encryption. In *Proc. 13th Computer Security Foundations Workshop (CSFW'00)*, pages 24–34. IEEE Comp. Soc. Press, 2000.

[94] J. Heather, G. Lowe, and S. Schneider. How to prevent type flaw attacks on security protocols. In *Proc. of the 13th Computer Security Foundations Workshop (CSFW'00)*. IEEE Computer Society Press, 2000.

[95] N. Heintze and J. D. Tygar. A model for secure protocols and their compositions. *IEEE Trans. Softw. Eng.*, 22:16–30, January 1996.

[96] M. Hirt and K. Sako. Efficient receipt-free voting based on homomorphic encryption. In *Proceedings of the 19th international conference on Theory and application of cryptographic techniques*, EUROCRYPT'00, pages 539–556, Berlin, Heidelberg, 2000. Springer-Verlag.

[97] H. Hüttel. Deciding framed bisimilarity. *Electr. Notes Theor. Comput. Sci.*, 68(6):1–18, 2002.

[98] S. Kremer and M. D. Ryan. Analysis of an electronic voting protocol in the applied pi-calculus. In *14th European Symposium on Programming (ESOP'05)*, volume 3444 of *LNCS*, pages 186–200. Springer, 2005.

[99] S. Kremer, M. D. Ryan, and B. Smyth. Election verifiability in electronic voting protocols. In D. Gritzalis and B. Preneel, editors, *Proceedings of the 15th European Symposium on Research in Computer Security (ESORICS'10)*, volume 6345 of *Lecture Notes in Computer Science*, pages 389–404, Athens, Greece, Sept. 2010. Springer.

[100] P. Lafourcade, D. Lugiez, and R. Treinen. Intruder deduction for the equational theory of abelian groups with distributive encryption. *Inf. Comput.*, 205:581–623, April 2007.

[101] Y. Li, W. Yang, and C.-W. Huang. Preventing type flaw attacks on security protocols with a simplified tagging scheme. In *Proceedings of the 2004 international symposium on Information and communication technologies*, ISICT '04, pages 244–249. Trinity College Dublin, 2004.

[102] J. Liu and H. Lin. A complete symbolic bisimulation for full applied pi calculus. In *Proc. 36th Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM'10)*, volume 5901 of *LNCS*, pages 552–563. Springer, 2010.

[103] M. Loukides and J. Gilmore, editors. *Cracking DES: Secrets of Encryption Research, Wiretap Politics and Chip Design*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 1998.

[104] G. Lowe. An attack on the Needham-Schroeder public-key authentication protocol. *Inf. Process. Lett.*, 56:131–133, November 1995.

[105] G. Lowe. A hierarchy of authentication specifications. In *Proceedings of the 10th IEEE workshop on Computer Security Foundations*, CSFW '97, pages 31–, Washington, DC, USA, 1997. IEEE Computer Society.

[106] A. J. Menezes, S. A. Vanstone, and P. C. V. Oorschot. *Handbook of Applied Cryptography*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 1996.

[107] J. Millen and V. Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *Proceedings of the 8th ACM conference on Computer and Communications Security*, CCS '01, pages 166–175, New York, NY, USA, 2001. ACM.

[108] R. Milner. *Communicating and mobile systems: the π-calculus*. Cambridge University Press, New York, NY, USA, 1999.

[109] S. Mödersheim and L. Viganò. Secure pseudonymous channels. In *Proceedings of the 14th European Symposium On Research In Computer Security (ESORICS'09)*, volume 5789 of *Lecture Notes in Computer Science*, pages 337–354. Springer, 2009.

[110] P. Narendran, F. Pfenning, and R. Statman. On the unification problem for cartesian closed categories (extended abstract). In *Proceedings, Eighth Annual Ieee Symposium On Logic In Computer Science*, pages 57–63. IEEE Computer Society Press, 1993.

[111] National Institute of Standards and Technology. *FIPS PUB 46-3: Data Encryption Standard (DES)*. Federal Information Processing Standards, Oct. 1999. supersedes FIPS 46-2.

[112] R. M. Needham and M. D. Schroeder. Using encryption for authentication in large networks of computers. *Commun. ACM*, 21:993–999, December 1978.

[113] T. Okamoto. An electronic voting scheme. In *IFIP World Conference on IT Tools*, pages 21–30, 1996.

[114] T. Okamoto. Receipt-free electronic voting schemes for large scale elections. In *Proc. 5th Int. Security Protocols Workshop*, volume 1361 of *lncs*. Springer, 1997.

[115] O. Pereira, J. j. Quisquater, U. C. Group, B. . A. nbkab, and A. . B. nb-kab. On the perfect encryption assumption. In *Workshop on Issues in the Theory of Security (in conjunction with ICALP00), University of Geneva, Switzerland 7,8 July 2000*, pages 42–45, 2000.

[116] R. Ramanujam and S.P.Suresh. Tagging makes secrecy decidable for unbounded nonces as well. In *Proc. of the 23rd Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'03)*, Mumbai, 2003.

[117] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 26:96–99, January 1983.

[118] M. Rusinowitch and M. Turuani. Protocol insecurity with finite number of sessions and composed keys is NP-complete. *Theoretical Computer Science*, 299:451–475, 2003.

[119] B. Smyth. *Formal verification of cryptographic protocols with automated reasoning*. PhD thesis, University of Birmingham, July 2011.

[120] J. G. Steiner, C. Neuman, and J. I. Schiller. Kerberos: An authentication service for open network systems. In *in Usenix Conference Proceedings*, pages 191–202, 1988.

[121] A. Tiu and J. E. Dawson. Automating open bisimulation checking for the spi calculus. In *CSF*, pages 307–321, 2010.

[122] H. C. A. van Tilborg, editor. *Encyclopedia of Cryptography and Security*. Springer, 2005.

[123] C. Weidenbach. Towards an automatic analysis of security protocols in first-order logic. In *Proc. 16th International Conference on Automated Deduction (CADE'99)*, volume 1632 of *LNCS*, pages 314–328. Springer, 1999.

Appendix A

Technical Proofs from Chapter 6

A.1 Proof of Lemma 6.6

This section consists of the proof of Lemma 6.6, which requires several helper lemmas. The proof makes use of techniques developed in [14]. We first recall the lemma that we are proving.

Lemma 6.6 (Fundamental Collapse Lemma)**.** *If* $s =_{\mathsf{E}} t$*, then* $\mathsf{col}(s) = C[[s_1, \ldots, s_k]]$*,* $\mathsf{col}(t) = D[[s_{k+1}, \ldots, s_{k+l}]]$ *such that* $\mathsf{domain}(C) = \mathsf{domain}(D)$ *and* $C[n_1, \ldots, n_k] =_{\mathsf{E}_d} D[n_{k+1}, \ldots, n_{k+l}]$ *where* $d = \mathsf{domain}(C)$ *and* $n_1, \ldots, n_{k+l}$ *are fresh names such that* $n_i = n_j$ *iff* $s_i =_{\mathsf{E}} s_j$ *for all* $1 \leq i, j \leq k + l$*.*

In Section A.1.1, we introduce several helper concepts, including a $\simeq$ relation between terms which under-approximates $=_{\mathsf{E}}$. In Section A.1.2, we introduce the *canonical form* of a term, similar to the collapsed form that we have already seen. Section A.1.3 includes a few technical results that we use further on. Section A.1.4 contains the main proof of this Appendix, namely that $\simeq$ is *complete* with respect to $=_{\mathsf{E}}$ in the sense that two terms that are equal modulo E have canonical forms that are in the $\simeq$ relation. Section A.1.5 provides the link between the canonical form and the collapsed form and contains the proof of Lemma 6.6.

A.1.1 Preliminaries

Before proceeding to the proof, we need to define a few concepts.

Definition A.1 (Equivalence classes)**.** If $\equiv$ is an equivalence relation on terms, we let

$$[t]_{\equiv} = \{s \mid s \equiv t\}$$

denote the equivalence class of t w.r.t. to $\equiv$. We let $\mathcal{A}_{\equiv} = \{[t]_{\equiv} \mid t \in T(\mathcal{F}, \mathcal{A})\}$ denote all such equivalence classes.

For any equivalence relation $\equiv$, we authorize the elements of $\mathcal{A}_{\equiv}$ (i.e. the equivalence classes) to be used as atoms in any term algebra $T(\mathcal{F}, \mathcal{A})$ where $\mathcal{A}_{\equiv} \subseteq \mathcal{A}$.

Definition A.2 (Abstraction by equivalence classes)**.** If $\equiv$ is an equivalence relation on terms in $T(\mathcal{F}, \mathcal{A}_0)$, we define the abstraction of $s \in T(\mathcal{F}, \mathcal{A}_0)$ w.r.t. the domain $d \in \{a, b, c\}$ to be

$$[s]^d_{\equiv} = C[[s_1]_{\equiv}, \ldots, [s_k]_{\equiv}] \qquad \text{if} \qquad s = C[s_1, \ldots, s_k]$$

is a term such that C is a pure d-context and $s_1, \ldots, s_k$ are terms with $\mathsf{domain}(s_i) \neq d$ for any $1 \leq i \leq k$. We extend abstractions to substitutions as expected:

$$[\sigma]^d_{\equiv} = \{x_1 \mapsto [t_1]^d_{\equiv}, \ldots, x_n \mapsto [t_n]^d_{\equiv}\}$$

if $\sigma = \{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ is a substitution.

Lemma A.1. *For any function symbol f of arity $\mathsf{ar}(f) = k$ and any sequence of terms $t_1, \ldots, t_k$, we have that*

$$[f(t_1, \ldots, t_k)]^d_{\equiv} = \begin{cases} f([t_1]^d_{\equiv}, \ldots, [t_k]^d_{\equiv}) & \text{if } f \in \mathcal{F}_d \\ [f(t_1, \ldots, t_k)]_{\equiv} & \text{if } f \notin \mathcal{F}_d. \end{cases}$$

Proof. Let C be a pure d-context and let $s_1, \ldots, s_l$ be terms with $\mathsf{domain}(i) \neq d$ for any $1 \leq i \leq l$ such that $f(t_1, \ldots, t_k) = C[s_1, \ldots, s_l]$. We distinguish between the two cases above:

1. if $f \in \mathcal{F}_d$, we have that $C = f(C_1, \ldots, C_k)$ for some pure d-contexts $C_1, \ldots, C_k$ such that $t_i = C_i[s_1, \ldots, s_l]$. By definition $[f(t_1, \ldots, t_k)]^d_{\equiv} = C[[s_1]_{\equiv}, \ldots, [s_l]_{\equiv}]$. By the choice of $C_1, \ldots, C_k$, we have that $C[[s_1]_{\equiv}, \ldots, [s_l]_{\equiv}] = f(C_1[[s_1]_{\equiv}, \ldots, [s_l]_{\equiv}], \ldots, C_k[[s_1]_{\equiv}, \ldots, [s_l]_{\equiv}])$. But $[t_i]^d_{\equiv} = C_i[[s_1]^d_{\equiv}, \ldots, [s_l]^d_{\equiv}]$ for $1 \leq i \leq k$. by definition. We obtain therefore that $f(C_1[[s_1]_{\equiv}, \ldots, [s_l]_{\equiv}], \ldots, C_k[[s_1]_{\equiv}, \ldots, [s_l]_{\equiv}]) = f([t_1]^d_{\equiv}, \ldots, [t_k]^d_{\equiv})$. By transitivity we can conclude that $[f(t_1, \ldots, t_k)]^d_{\equiv} = f([t_1]^d_{\equiv}, \ldots, [t_k]^d_{\equiv})$, which is what we had to show.
2. if $f \notin \mathcal{F}_d$, it follows that $C = \_^i$ for some $1 \leq i \leq k$ is a hole and $C[s_1, \ldots, s_l] = s_i = f(t_1, \ldots t_k)$. Therefore $[f(t_1, \ldots, t_k)]^d_{\equiv} = [s_i]_{\equiv} = [f(t_1, \ldots, t_k)]_{\equiv}$, which is what we had to prove.

□

Lemma A.2 (Equivalence implication)**.** *Let $d \in \{a, b, c\}$ be a domain. If $\equiv_1$ and $\equiv_2$ are equivalence classes on terms in $T(\mathcal{F}, \mathcal{A}_0)$ such that $s \equiv_1 t$ implies $s \equiv_2 t$ for all terms $s, t \in T(\mathcal{F}, \mathcal{A}_0)$, then*

$$[s]^d_{\equiv_1} =_{\mathsf{E}_d} [t]^d_{\equiv_1} \text{ implies } [s]^d_{\equiv_2} =_{\mathsf{E}_d} [t]^d_{\equiv_2}$$

for any two terms $s, t \in T(\mathcal{F}, \mathcal{A}_0)$.

Proof. Let $s, t \in T(\mathcal{F}, \mathcal{A}_0)$ be arbitrary terms such that

$$[s]^d_{\equiv_1} =_{\mathsf{E}_d} [t]^d_{\equiv_1}.$$

We will show that $[s]^d_{\equiv_2} =_{\mathsf{E}_d} [t]^d_{\equiv_2}$. Let C and D be pure d-contexts and $s_1, \ldots, s_k, s_{k+1}, \ldots, s_{k+l}$ be terms with $\mathsf{domain}(s_i) \neq d$ (for $1 \leq i \leq k+l$) such that $s = C[s_1, \ldots, s_k]$ and $t = D[s_{k+1}, \ldots, s_{k+l}]$.

By definition, we have $[s]^d_{\equiv_1} = C[[s_1]_{\equiv_1}, \ldots, [s_k]_{\equiv_1}]$ and that $[t]^d_{\equiv_1} = D[[s_{k+1}]_{\equiv_1}, \ldots, [s_{k+l}]_{\equiv_1}]$, which implies

$$C[[s_1]_{\equiv_1}, \ldots, [s_k]_{\equiv_1}] =_{\mathsf{E}_d} D[[s_{k+1}]_{\equiv_1}, \ldots, [s_{k+l}]_{\equiv_1}].$$

By the assumption of $\equiv_1$ and $\equiv_2$, we have that $[s_i]_{\equiv_1} = [s_j]_{\equiv_1}$ implies $[s_i]_{\equiv_2} = [s_j]_{\equiv_2}$ for all $1 \leq i, j \leq k+l$. Therefore the replacement $\sigma = \{[s_i]_{\equiv_1} \mapsto [s_i]_{\equiv_2}\}_{1 \leq i \leq k+l}$ is well-defined. As the equational theory E_d is stable by replacement of atoms by terms, we have that

$$C[[s_1]_{\equiv_1}, \ldots, [s_k]_{\equiv_1}]\sigma =_{\mathsf{E}_d} D[[s_{k+1}]_{\equiv_1}, \ldots, [s_{k+l}]_{\equiv_1}]\sigma.$$

But by definition

$$[s]^d_{\equiv_2} = C[[s_1]_{\equiv_2}, \ldots, [s_k]_{\equiv_2}] = C[[s_1]_{\equiv_1}, \ldots, [s_k]_{\equiv_1}]\sigma \text{ and}$$
$$[t]^d_{\equiv_2} = D[[s_{k+1}]_{\equiv_2}, \ldots, [s_{k+l}]_{\equiv_2}] = D[[s_{k+1}]_{\equiv_1}, \ldots, [s_{k+l}]_{\equiv_1}]\sigma.$$

This implies $[s]^d_{\equiv_2} =_{\mathsf{E}_d} [t]^d_{\equiv_2}$, which is what we had to prove.

□

For every $n \geq 0$, we let $\simeq_n$ to be an equivalence relation on terms defined as follows:

Definition A.3. For two terms s and t, we write $s \simeq_n t$ if s and t have the same domain $d = \mathsf{domain}(s) = \mathsf{domain}(t)$ and

$$\begin{array}{ll} [s]^d_{\simeq_{n-1}} =_{\mathsf{E}_d} [t]^d_{\simeq_{n-1}} & \text{when } n > 0 \\ [s]^d_{=} = s =_{\mathsf{E}_d} t = [t]^d_{=} & \text{when } n = 0. \end{array}$$

We first show that two terms which are in the $\simeq_n$ relation are equivalent modulo E.

Lemma A.3. *For any $n \in \mathbb{N}$, for any terms s, t we have that $s \simeq_n t$ implies $s =_{\mathsf{E}} t$.*

Proof. By well-founded induction on n. If $n = 0$ we have that $s =_{\mathsf{E}_d} t$ which immediately implies $s =_{\mathsf{E}} t$ since $\mathsf{E}_d \subseteq \mathsf{E}$.

If $n > 0$, we have by definition that $\mathsf{domain}(s) = \mathsf{domain}(t) = d$ for some $d \in \{a, b, c\}$ and that $[s]^d_{\simeq_{n-1}} =_{\mathsf{E}_d} [t]^d_{\simeq_{n-1}}$. By the induction hypothesis we have that $u \simeq_{n-1} v$ implies $u =_{\mathsf{E}} v$ for all terms u, v. Therefore, by Lemma A.2, we have that $[s]^d_{\mathsf{E}} =_{\mathsf{E}_d} [t]^d_{\mathsf{E}}$.

Let C, D be pure d-contexts such that $s = C[[s_1, \ldots, s_k]]$ and $t = D[[s_{k+1}, \ldots, s_{k+l}]]$ for some terms $s_1, \ldots, s_k, s_{k+1}, \ldots, s_{k+l}$. From the fact that $[s]^d_{\mathsf{E}} = [t]^d_{\mathsf{E}}$ we obtain that $C[[s_1]_{\mathsf{E}}, \ldots, [s_k]_{\mathsf{E}}] =_{\mathsf{E}_d} D[[s_{k+1}]_{\mathsf{E}}, \ldots, [s_{k+l}]_{\mathsf{E}}]$, which implies $C[[s_1]_{\mathsf{E}}, \ldots, [s_k]_{\mathsf{E}}] =_{\mathsf{E}} D[[s_{k+1}]_{\mathsf{E}}, \ldots, [s_{k+l}]_{\mathsf{E}}]$. Let $\mathit{first}(i) = \min\{j \mid s_j =_{\mathsf{E}} s_i\}$. The replacement $\sigma = \{[s_i]_{\mathsf{E}} \mapsto s_{\mathit{first}(i)}\}$ is then well defined. As E is stable by replacement of atoms by terms, we obtain that $C[[s_1]_{\mathsf{E}}, \ldots, [s_k]_{\mathsf{E}}]\sigma =_{\mathsf{E}} D[[s_{k+1}]_{\mathsf{E}}, \ldots, [s_{k+l}]_{\mathsf{E}}]\sigma$. This is equivalent to $C[s_{\mathit{first}(1)}, \ldots, s_{\mathit{first}(k)}] =_{\mathsf{E}} D[s_{\mathit{first}(k+1)}, \ldots, s_{\mathit{first}(k+l)}]$. But $s_{\mathit{first}(i)} =_{\mathsf{E}} s_i$ for all $1 \leq i \leq k + l$ and therefore $C[s_1, \ldots, s_l] =_{\mathsf{E}} D[s_{k+1}, \ldots, s_{k+l}]$. This is equivalent to $s =_{\mathsf{E}} t$ which is what we wanted to prove.

□

We have that:

Lemma A.4. *Let s, t be two terms such that $s \simeq_n t$. Then $s \simeq_{n+1} t$.*

Proof. By well-founded induction on n.

If $n = 0$, $s \simeq_0 t$ implies $[s]^d_{=} = s =_{\mathsf{E}_d} t = [t]^d_{=}$. As $[s]^d_{=} = s$ and $[t]^d_{=} = t$, it follows that s and t are pure d terms and therefore $[s]^d_{\simeq_0} = s$ and $[t]^d_{\simeq_0} = t$ by definition. But $s =_{\mathsf{E}_d} t$ and we obtain $[s]^d_{\simeq_0} = [t]^d_{\simeq_0}$ which is the definition of $s \simeq_1 t$.

If $n > 0$, as $s \simeq_n t$, we have that the terms s and t have the same domain $d = \mathsf{domain}(s) = \mathsf{domain}(t)$ and that

$$[s]^d_{\simeq_{n-1}} =_{\mathsf{E}_d} [t]^d_{\simeq_{n-1}}.$$

By the induction hypothesis, we have that $u \simeq_{n-1} v$ implies $u \simeq_n v$ for all terms u, v. Therefore by Lemma A.2, we have that

$$[s]^d_{\simeq_n} =_{\mathsf{E}_d} [t]^d_{\simeq_n},$$

which implies $s \simeq_{n+1} t$.

□

We now define $\simeq$ to be a binary relation between terms:

Definition A.4. We write that $s \simeq t$ if there exists n such that $s \simeq_n t$.

It is easy to see that $\simeq$ is an equivalence relation: symmetry and reflexivity are immediate from the definition of $\simeq_n$ and transitivity is easily proved using Lemma A.4.

Lemma A.5. *For any terms s, t, we have that $s \simeq t$ implies $s =_{\mathsf{E}} t$.*

Proof. Immediate by the definition of $\simeq$ and Lemma A.3.

□

We now directly characterize when $s \simeq t$, without going through $\simeq_n$.

Lemma A.6. *We have that $s \simeq t$ iff s and t have the same domain $d = \mathsf{domain}(s) = \mathsf{domain}(t)$ and $[s]^d_{\simeq} =_{\mathsf{E}_d} [t]^d_{\simeq}$.*

Proof. We prove each implication separately:

1. First assume that $s \simeq t$. We will show that s and t have the same domain $d = \mathsf{domain}(s) = \mathsf{domain}(t)$ and that $[s]^d_{\simeq} =_{\mathsf{E}_d} [t]^d_{\simeq}$.

   If follows by the definition of $\simeq$ that there exists an integer $n \in \mathbb{N}$ such that $s \simeq_n t$. By Lemma A.4 we have that $s \simeq_{n+1} t$. By the definition of $\simeq_{n+1}$, we have that s and t have the same domain. Let $d = \mathsf{domain}(s) = \mathsf{domain}(t)$. Furthermore, by the definition of $\simeq_{n+1}$, it follows that

   $$[s]^d_{\simeq_n} =_{\mathsf{E}_d} [t]^d_{\simeq_n}.$$

   By the definition of $\simeq$, we have that $u \simeq_n v$ implies $u \simeq v$ for any two terms u and v. Therefore by Lemma A.2 we have that

   $$[s]^d_{\simeq} =_{\mathsf{E}_d} [t]^d_{\simeq},$$

   which is what we had to show.

2. We now assume that s and t have the same domain $d = \mathsf{domain}(s) = \mathsf{domain}(t)$ and that $[s]^d_{\simeq} =_{\mathsf{E}_d} [t]^d_{\simeq}$. We will show that $s \simeq t$.

   Let C and D be pure d-contexts and $s_1, \ldots, , s_{k+l}$ be terms with $\mathsf{domain}(s_i) \neq d$ (for $1 \leq i \leq k+l$) such that $s = C[s_1, \ldots, s_k]$ and $t = D[t_{k+1}, \ldots, t_{k+l}]$. We then have by definition that $[s]^d_{\simeq} = C[[s_1]_{\simeq}, \ldots, [s_k]_{\simeq}]$ and that $[t]^d_{\simeq} = D[[s_{k+1}]_{\simeq}, \ldots, [s_{k+l}]_{\simeq}]$. As $[s]^d_{\simeq} =_{\mathsf{E}_d} [t]^d_{\simeq}$, we have that

   $$C[[s_1]_{\simeq}, \ldots, [s_k]_{\simeq}] =_{\mathsf{E}_d} D[[s_{k+1}]_{\simeq}, \ldots, [s_{k+l}]_{\simeq}].$$

   By the definition of $\simeq$, we have that for any $1 \leq i, j \leq k+l$, $[s_i]_{\simeq} = [s_j]_{\simeq}$ iff there exists $n_{i,j} \in \mathbb{N}$ such that $[s_i]_{\simeq_{n_{i,j}}} = [s_j]_{\simeq_{n_{i,j}}}$. Let $n = \max\{n_{i,j} \mid 1 \leq i, j \leq k+l\}$. We have that $[s_i]_{\simeq} = [s_j]_{\simeq}$ implies $[s_i]_{\simeq_{n_{i,j}}} = [s_j]_{\simeq_{n_{i,j}}}$ which implies by Lemma A.4 $[s_i]_{\simeq_n} = [s_j]_{\simeq_n}$ for all $1 \leq i, j \leq k+l$. The replacement

   $$\sigma = \{[s_i]_{\simeq} \mapsto [s_i]_{\simeq_n}\}_{1 \leq i \leq k+l}$$

   is therefore well-defined and, as E_d is stable by replacement of atoms by terms, we have that

   $$C[[s_1]_{\simeq}, \ldots, [s_k]_{\simeq}]\sigma =_{\mathsf{E}_d} D[[s_{k+1}]_{\simeq}, \ldots, [s_{k+l}]_{\simeq}]\sigma.$$

   But by the definitions of $[s]^d_{\simeq_{n+1}}$ and $[t]^d_{\simeq_{n+1}}$ we have

   $$\begin{aligned} [s]^d_{\simeq_{n+1}} &= C[[s_1]_{\simeq_n}, \ldots, [s_k]_{\simeq_n}] = C[[s_1]_{\simeq}, \ldots, [s_k]_{\simeq}]\sigma \text{ and} \\ [t]^d_{\simeq_{n+1}} &= D[[s_{k+1}]_{\simeq_n}, \ldots, [s_{k+l}]_{\simeq_n}] = D[[s_{k+1}]_{\simeq}, \ldots, [s_{k+l}]_{\simeq}]\sigma. \end{aligned}$$

   and therefore $[s]^d_{\simeq_{n+1}} =_{\mathsf{E}_d} [t]^d_{\simeq_{n+1}}$. As $\mathsf{domain}(s) = \mathsf{domain}(t) = d$, we have by definition that $s \simeq_{n+2} t$. But $s \simeq_{n+2} t$ implies by the definition of $\simeq$ that $s \simeq t$, which is what we had to show.

□

Lemma A.7. *If $s \simeq t$ then $[s]^d_{\simeq} =_{\mathsf{E}_d} [t]^d_{\simeq}$ for any domain $d \in \{a, b, c\}$.*

Proof. As $s \simeq t$, it follows by Lemma A.6 that $\mathsf{domain}(s) = \mathsf{domain}(t)$. If $\mathsf{domain}(s) = d$, then we get our conclusion by Lemma A.6. Otherwise, if $\mathsf{domain}(s) \neq d$, we have that $[s]^d_{\simeq} = [s]_{\simeq}$ and $[t]^d_{\simeq} = [t]_{\simeq}$ by the definition of $[\cdot]^d_{\simeq}$. But $s \simeq t$ and therefore $[s]_{\simeq} = [t]_{\simeq}$. We immediately obtain $[s]^d_{\simeq} = [t]^d_{\simeq}$.

□

Lemma A.8. *If $f \in \mathcal{F}$ is a function symbol of arity k and $s_1 \simeq t_1, \ldots, s_k \simeq t_k$ are terms, we have that*

$$f(s_1, \ldots, s_k) \simeq f(t_1, \ldots, t_k).$$

Proof. Let $d = \mathsf{domain}(f)$ be the domain of f. We have that $[f(s_1, \ldots, s_k)]^d_{\simeq} = f([s_1]^d_{\simeq}, \ldots, [s_k]^d_{\simeq})$ and $[f(t_1, \ldots, t_k)]^d_{\simeq} = f([t_1]^d_{\simeq}, \ldots, [t_k]^d_{\simeq})$ by Lemma A.1. As $s_i \simeq t_i$ we have by Lemma A.7 that $[s_i]^d_{\simeq} =_{\mathsf{E}_d} [t_i]^d_{\simeq}$ for all $1 \leq i \leq k$. Therefore $[f(s_1, \ldots, s_k)]^d_{\simeq} =_{\mathsf{E}_d} [f(t_1, \ldots, t_k)]^d_{\simeq}$. Combined with the fact that $\mathsf{domain}(f(s_1, \ldots, s_k)) = \mathsf{domain}(f(t_1, \ldots, t_k)) = \mathsf{domain}(f) = d$, we obtain by Lemma A.6 that $f(s_1, \ldots, s_k) \simeq f(t_1, \ldots, t_k)$, which is what we had to prove.

□

A.1.2 Canonical Form

We now introduce the *canonical form* $\mathsf{can}(t)$ of a term t, defined as follows:

Definition A.5.

$$\mathsf{can}(t) = \begin{cases} t & \text{if } t \text{ is an atom} \\ s_i & \text{if } t = f(t_1, \ldots, t_l), \\ & \quad f(\mathsf{can}(t_1), \ldots, \mathsf{can}(t_l)) = C[[s_1, \ldots, s_k]] \text{ and} \\ & \quad C[[n_1, \ldots, n_k]] =_{\mathsf{E}_d} n_i \text{ where } n_1, \ldots, n_k \text{ are fresh} \\ & \quad \text{names such that } n_i = n_j \text{ iff } s_i \simeq s_j \text{ for all } 1 \leq i, j \leq l \\ & \quad \text{and } d = \mathsf{domain}(C) \\ f(\mathsf{can}(t_1), \ldots, \mathsf{can}(t_l)) & \text{if } t = f(t_1, \ldots, t_l) \text{ but the above does not hold.} \end{cases}$$

We say that a term t is in canonical form if $t = \mathsf{can}(t)$. The difference between the collapsed form of a term and the canonical form is the choice of the fresh names $n_1, \ldots, n_k$ in the second and third cases of the definition. In the case of col, two names n_i and n_j are chosen to be equal ($n_i = n_j$) iff $s_i =_{\mathsf{E}} s_j$, while in the case of the canonical form $n_i = n_j$ iff $s_i \simeq s_j$. As we will see later, $=_{\mathsf{E}}$ and $\simeq$ coincide on canonical forms and therefore the canonical form is identical to the collapsed form. However, this definition of canonical form is required for the proofs. We extend can to substitutions as expected:

$$\mathsf{can}(\sigma) = \{x_1 \mapsto \mathsf{can}(t_1), \ldots, x_k \mapsto \mathsf{can}(t_k)\}$$

whenever $\sigma = \{x_1 \mapsto t_1, \ldots, x_k \mapsto t_k\}$. We will first show that the canonical form of a term is equal modulo E to the term.

Lemma A.9. *For any term t, we have that $\mathsf{can}(t) =_{\mathsf{E}} t$.*

Proof. By induction on the size of t.

1. For the base case, if t is an atom, then $\mathsf{can}(t) = t$ by the definition of can and therefore $\mathsf{can}(t) =_{\mathsf{E}} t$.
2. For the inductive case, if $t = f(t_1, \ldots, t_l)$, we have that $t_1 =_{\mathsf{E}} \mathsf{can}(t_1), \ldots, t_l =_{\mathsf{E}} \mathsf{can}(t_l))$ by the induction hypothesis. Therefore $t = f(t_1, \ldots, t_l) =_{\mathsf{E}} f(\mathsf{can}(t_1)), \ldots, \mathsf{can}(t_l))$.

   We distinguish two cases:

   (a) if $\mathsf{can}(t) = f(\mathsf{can}(t_1), \ldots, \mathsf{can}(t_l))$ (i.e. the third case of the definition of can) we have already shown that $t =_{\mathsf{E}} \mathsf{can}(t)$.

   (b) otherwise, $f(\mathsf{can}(t_1), \ldots, \mathsf{can}(t_l)) = C[[s_1, \ldots, s_k]]$ for some context C and some terms $s_1, \ldots, s_k$ and there exists an index $1 \leq i \leq k$ such that $C[n_1, \ldots, n_k] =_{\mathsf{E}_d} n_i$ for some fresh names $n_1, \ldots, n_k$ such that $n_x = n_y$ iff $s_x \simeq s_y$ for all $1 \leq x, y \leq k$ and $\mathsf{can}(t) = s_i$, where $d = \mathsf{domain}(C)$. As $C[n_1, \ldots, n_k] =_{\mathsf{E}_d} n_i$ and $\mathsf{E}_d \subseteq \mathsf{E}$, it follows that $C[n_1, \ldots, n_k] =_{\mathsf{E}} n_i$.

   For all $1 \leq x \leq k$, let $[x]$ denote the smallest index $1 \leq y \leq k$ such that $n_x = n_y$. As the equational theory E is stable by replacement of names by arbitrary terms, we have that $C[n_1, \ldots, n_k] =_{\mathsf{E}} n_i$ implies $C[n_1, \ldots, n_k]\{n_1 \mapsto s_{[1]}, \ldots, n_k \mapsto s_{[k]}\} =_{\mathsf{E}} n_i\{n_1 \mapsto s_{[1]}, \ldots, n_k \mapsto s_{[k]}\}$, which is equivalent to $C[s_{[1]}, \ldots, s_{[k]}] =_{\mathsf{E}} s_{[i]}$. But by choice of the

names $n_1, \ldots, n_k$ and the definition of $[x]$, we have that $s_{[x]} \simeq s_x$ for all $1 \leq x \leq k$. By Lemma A.5, we have that $s_{[x]} \simeq s_x$ implies $s_{[x]} =_{\mathsf{E}} s_x$ for all $1 \leq x \leq k$ and therefore we obtain $C[s_1, \ldots, s_k] =_{\mathsf{E}} s_i$.
We have that $t =_{\mathsf{E}} f(\mathsf{can}(t_1), \ldots, \mathsf{can}(t_l))$, that $f(\mathsf{can}(t_1), \ldots, \mathsf{can}(t_l)) = C[[s_1, \ldots, s_k]]$, that $C[s_1, \ldots, s_k] =_{\mathsf{E}} s_i$ and that $\mathsf{can}(t) = s_i$ for some index $1 \leq i \leq k$.
By transitivity of $=_{\mathsf{E}}$, we immediately obtain that $t =_{\mathsf{E}} \mathsf{can}(t)$.

□

We now prove some properties of the canonical form. First of all, any subterm of a canonicalized subterm is canonicalized.

Lemma A.10. *Any subterm $t \in \mathsf{st}(\mathsf{can}(s))$ of $\mathsf{can}(s)$ is such that $t = \mathsf{can}(t)$.*

Proof. By induction on the size of s. We distinguish among the following cases:

1. if s is an atom, then $\mathsf{can}(s) = s$ and its only subterm $t \in \mathsf{st}(\mathsf{can}(s))$ is $t = \mathsf{can}(s)$. Therefore $t = s$ and $\mathsf{can}(t) = t$.

2. if $s = f(t_1, \ldots, t_l)$ and $\mathsf{can}(s) = f(\mathsf{can}(t_1), \ldots, \mathsf{can}(t_l))$, then any subterm $t \in \mathsf{st}(\mathsf{can}(s))$ is either:

   (a) a subterm $t \in \mathsf{st}(\mathsf{can}(t_i))$ of the term $\mathsf{can}(t_i)$ for some i, in which case $t = \mathsf{can}(t)$ by the induction hypothesis,

   (b) or $t = \mathsf{can}(s)$ is the entire term, in which case $\mathsf{can}(t) = \mathsf{can}(f(\mathsf{can}(t_1), \ldots, \mathsf{can}(t_l))$ and it is easy to see that $\mathsf{can}(t) = t$.

3. otherwise we have that $s = f(t_1, \ldots, t_l)$, $f(\mathsf{can}(t_1), \ldots, \mathsf{can}(t_l)) = C[[s_1, \ldots, s_k]]$, $\mathsf{can}(s) = s_x$ and $C[n_1, \ldots, n_k] =_{\mathsf{E}_d} n_x$ for some $x \in \{1, \ldots, k\}$ where $n_1, \ldots, n_k$ are fresh names such that $n_i = n_j$ iff $s_i \simeq s_j$ for all $1 \leq i, j \leq k$ and $c = \mathsf{domain}(C)$.

   As C is not the empty context, we have that s_x is a subterm of $\mathsf{can}(t_y)$ for some y. We conclude by the induction hypothesis, since any subterm t of $\mathsf{can}(t_y)$ (and therefore any subterm t of $s_x = \mathsf{can}(s)$) is such that $t = \mathsf{can}(t)$.

□

We will now show that canonicalizing the direct subterms of a term does not change the canonical form of the term.

Lemma A.11. *For any function symbol $f \in \Sigma$ of arity $\mathsf{ar}(f) = k$ and any terms $s_1, \ldots, s_k$, we have that $\mathsf{can}(f(s_1, \ldots, s_k)) = \mathsf{can}(f(\mathsf{can}(s_1), \ldots, \mathsf{can}(s_k))$.*

Proof. By the definition of can,

1. either $\mathsf{can}(f(s_1, \ldots, s_k)) = f(\mathsf{can}(s_1), \ldots, \mathsf{can}(s_k))$, in which case $f(\mathsf{can}(s_1), \ldots, \mathsf{can}(s_k))$ is a canonicalized term (as it is the same term as $\mathsf{can}(f(s_1, \ldots, s_k))$). Therefore

   $$f(\mathsf{can}(s_1), \ldots, \mathsf{can}(s_k)) = \mathsf{can}(f(\mathsf{can}(s_1), \ldots, \mathsf{can}(s_k))).$$

   By transitivity of $=$, we immediately obtain $\mathsf{can}(f(s_1, \ldots, s_k)) = \mathsf{can}(f(\mathsf{can}(s_1), \ldots, \mathsf{can}(s_k)))$.

2. or we have that

   $$\mathsf{can}(f(s_1, \ldots, s_k)) = u_j \tag{A.1}$$

   for some term u_j where $f(\mathsf{can}(s_1), \ldots, \mathsf{can}(s_k)) = C[[u_1, \ldots, u_l]]$ and $C[n_1, \ldots, n_l] =_{\mathsf{E}_d} n_j$, where $d = \mathsf{domain}(C)$ and $n_1, \ldots, n_l$ are fresh names such that $n_x = n_y$ iff $u_x \simeq u_y$ for all $1 \leq x, y \leq l$.

In this case, let us analyze the canonicalized form of the term $t = f(\mathsf{can}(s_1), \ldots, \mathsf{can}(s_k))$. We have that

$$f(\mathsf{can}(\mathsf{can}(s_1)), \ldots, \mathsf{can}(\mathsf{can}(s_k))) \overset{\text{by Lemma A.10}}{=} f(\mathsf{can}(s_1), \ldots, \mathsf{can}(s_k)) = C[[u_1, \ldots, u_l]].$$

As $C[n_1, \ldots, n_l] =_{\mathsf{E}_d} n_j$, we have that the canonicalized form of $t = f(\mathsf{can}(s_1), \ldots, \mathsf{can}(s_k))$ is by definition

$$\mathsf{can}(f(\mathsf{can}(s_1), \ldots, \mathsf{can}(s_k))) = u_j. \tag{A.2}$$

From Equation (A.1) and Equation A.2, we have that

$$\mathsf{can}(f(s_1, \ldots, s_k)) = \mathsf{can}(f(\mathsf{can}(s_1), \ldots, \mathsf{can}(s_k)),$$

which is what we had to prove.

□

We can now show that canonicalization of any set of subterms of a term does not change the canonical form of the term.

Lemma A.12. *For any context C and any terms $s_1, \ldots, s_k$, we have that*

$$\mathsf{can}(C[s_1, \ldots, s_k]) = \mathsf{can}(C[\mathsf{can}(s_1), \ldots, \mathsf{can}(s_k)]).$$

Proof. By induction on C:

1. if C is a hole $\_^i$ for some $1 \leq i \leq k$, then $C[s_1, \ldots, s_k] = s_i$ and $C[\mathsf{can}(s_1), \ldots, \mathsf{can}(s_k)] = \mathsf{can}(s_i)$. Therefore

   $$\mathsf{can}(C[s_1, \ldots, s_k]) = \mathsf{can}(s_i) = C[\mathsf{can}(s_1), \ldots, \mathsf{can}(s_k)]. \tag{A.3}$$

   As $\mathsf{can}(s_i)$ is canonicalized by Lemma A.10, we have that $C[\mathsf{can}(s_1), \ldots, \mathsf{can}(s_k)]$ (which is the same term as $\mathsf{can}(s_i)$) is canonicalized as well. Therefore

   $$C[\mathsf{can}(s_1), \ldots, \mathsf{can}(s_k)] = \mathsf{can}(C[\mathsf{can}(s_1), \ldots, \mathsf{can}(s_k)]). \tag{A.4}$$

   From Equation (A.3) and Equation (A.4), we obtain by the transitivity of $=$ that

   $$\mathsf{can}(C[s_1, \ldots, s_k]) = \mathsf{can}(C[\mathsf{can}(s_1), \ldots, \mathsf{can}(s_k)]),$$

   which is what we had to prove.

2. if $C = a$ is an atom, then $C[s_1, \ldots, s_k] = a$ and $C[\mathsf{can}(s_1), \ldots, \mathsf{can}(s_k)] = a$. As $\mathsf{can}(a) = a$ by the definition of can, we immediately obtain

   $$\mathsf{can}(C[s_1, \ldots, s_k]) = \mathsf{can}(a) = a = \mathsf{can}(a) = \mathsf{can}(C[\mathsf{can}(s_1), \ldots, \mathsf{can}(s_k)]),$$

   which is what we had to prove.

3. otherwise, $C = f(C_1, \ldots, C_l)$ for some function symbol f of arity $\mathsf{ar}(f) = l$ and some contexts $C_1, \ldots, C_l$. Let $\tilde{s}$ denote the sequence of terms $s_1, \ldots, s_k$ and $\mathsf{can}(\tilde{s})$ denote the sequence of terms $\mathsf{can}(s_1), \ldots, \mathsf{can}(s_k)$. By the induction hypothesis we have that

   $$\mathsf{can}(C_i[\tilde{s}]) = \mathsf{can}(C_i[\mathsf{can}(\tilde{s})])) \text{ for all } 1 \leq i \leq l. \tag{A.5}$$

We have that

$$\begin{array}{rcl}
\mathsf{can}(C[\tilde{s}]) & = & \mathsf{can}(f(C_1[\tilde{s}],\ldots,C_l[\tilde{s}])) \\
& & \text{(by choice of } C_1,\ldots,C_l) \\
& = & \mathsf{can}(f(\mathsf{can}(C_1[\tilde{s}]),\ldots,\mathsf{can}(C_l[\tilde{s}]))) \\
& & \text{(by Lemma A.11)} \\
& = & \mathsf{can}(f(\mathsf{can}(C_1[\mathsf{can}(\tilde{s})]),\ldots,\mathsf{can}(C_l[\mathsf{can}(\tilde{s})]))) \\
& & \text{(by Equation (A.5))} \\
& = & \mathsf{can}(f(C_1[\mathsf{can}(\tilde{s})],\ldots,C_l[\mathsf{can}(\tilde{s})])) \\
& & \text{(by Lemma A.11)} \\
& = & \mathsf{can}(C[\mathsf{can}(\tilde{s})]). \\
& & \text{(by choice of } C_1,\ldots,C_l)
\end{array}$$

In summary, we obtained $\mathsf{can}(C[\tilde{s}]) = \mathsf{can}(C[\mathsf{can}(\tilde{s})])$, which is what we had to prove.

□

We immediately obtain as a corollary another way to state the above lemma.

Corollary A.1. *For any term s and any substitution σ, we have that $\mathsf{can}(s\sigma) = \mathsf{can}(s(\mathsf{can}(\sigma)))$.*

Proof. Let C be a context obtained from s by replacing all variables $x \in \mathcal{D}om(\sigma)$ with holes $\_^i$ $(1 \leq i \leq n)$ such that $s\sigma = C[x_1\sigma,\ldots,x_n\sigma]$ and $s(\mathsf{can}(\sigma)) = C[x_1(\mathsf{can}(\sigma)),\ldots,x_n(\mathsf{can}(\sigma))]$ where $\mathcal{D}om(\sigma) = \{x_1,\ldots,x_n\}$.

By Lemma A.12, we have that $\mathsf{can}(C[x_1\sigma,\ldots,x_n\sigma]) = \mathsf{can}(C[\mathsf{can}(x_1\sigma),\ldots,\mathsf{can}(x_n\sigma)]$. This is equivalent to $\mathsf{can}(s\sigma) = \mathsf{can}(s(\mathsf{can}(\sigma)))$, which is what we had to prove.

□

A.1.3 Abstraction and Canonicalization

We now study the interaction between abstraction and canonical forms. We first need a technical lemma:

Lemma A.13. *Let $d \in \{a,b\}$ be a domain and $f \in \mathcal{F}_d$ be a function symbol of arity $\mathsf{ar}(f) = k$. For any terms $s_1,\ldots,s_k$, we have that*

$$[\mathsf{can}(f(s_1,\ldots,s_k))]^d_{\simeq} =_{\mathsf{E}_d} f([\mathsf{can}(s_1)]^d_{\simeq},\ldots,[\mathsf{can}(s_k)]^d_{\simeq}).$$

Proof. By the definition of can, we distinguish between two cases:

1. either $\mathsf{can}(f(s_1,\ldots,s_k)) = f(\mathsf{can}(s_1),\ldots,\mathsf{can}(s_k))$, in which case

   $$[\mathsf{can}(f(s_1,\ldots,s_k))]^d_{\simeq} = [f(\mathsf{can}(s_1),\ldots,\mathsf{can}(s_k))]^d_{\simeq}.$$

   But as $f \in \mathcal{F}_d$, we have by Lemma A.1 that

   $$[f(\mathsf{can}(s_1),\ldots,\mathsf{can}(s_k))]^d_{\simeq} = f([\mathsf{can}(s_1)]^d_{\simeq},\ldots,[\mathsf{can}(s_k)]^d_{\simeq}).$$

   By transitivity of $=$, we obtain that $[\mathsf{can}(f(s_1,\ldots,s_k))]^d_{\simeq} = f([\mathsf{can}(s_1)]^d_{\simeq},\ldots,[\mathsf{can}(s_k)]^d_{\simeq})$, which immediately implies the equality modulo E_d of the two terms:

   $$[\mathsf{can}(f(s_1,\ldots,s_k))]^d_{\simeq} =_{\mathsf{E}_d} f([\mathsf{can}(s_1)]^d_{\simeq},\ldots,[\mathsf{can}(s_k)]^d_{\simeq}).$$

2. or $f(\mathsf{can}(s_1),\ldots,\mathsf{can}(s_k)) = D[[t_1,\ldots,t_l]]$ for some context D and some terms $t_1,\ldots,t_l$ such that $D[n_1,\ldots,n_l] =_{\mathsf{E}_d} n_i$ for some $1 \leq i \leq l$, where $n_1,\ldots,n_l$ are fresh names such that $n_x = n_y$ iff $t_x \simeq t_y$ for all $1 \leq x,y \leq l$. In this case $\mathsf{can}(f(s_1,\ldots,s_k)) = t_i$ and we have by definition that

   $$[\mathsf{can}(f(s_1,\ldots,s_k))]^d_{\simeq} = [t_i]_{\simeq} \tag{A.6}$$

because $\mathsf{domain}(t_i) \neq \mathsf{domain}(D) = d$.

On the other hand, we have that

$$\begin{array}{rcll} f([\mathsf{can}(s_1)]^d_{\simeq}, \ldots, [\mathsf{can}(s_k)]^d_{\simeq}) & = & [f(\mathsf{can}(s_1), \ldots, \mathsf{can}(s_k))]^d_{\simeq} & \text{(by Lemma A.1)} \\ & = & [D[t_1, \ldots, t_l]]^d_{\simeq} & \text{(by choice of } D, t_1, \ldots, t_l) \\ & = & D[[t_1]_{\simeq}, \ldots, [t_l]_{\simeq}]. & \text{(by definition)} \end{array}$$

But $D[n_1, \ldots, n_l] =_{\mathsf{E}_d} n_i$ and, as E_d is stable by replacement of names by terms, we obtain that $D[n_1, \ldots, n_l]\sigma =_{\mathsf{E}_d} n_i\sigma$, where $\sigma = \{n_x \mapsto [t_x]_{\simeq}\}_{1 \leq x \leq l}$. The replacement σ is well-defined since $n_x = n_y$ implies $[t_x]_{\simeq} = [t_y]_{\simeq}$ for all $1 \leq x, y \leq l$. The fact that $D[n_1, \ldots, n_l]\sigma =_{\mathsf{E}_d} n_i\sigma$ is equivalent to $D[[t_1]_{\simeq}, \ldots, [t_l]_{\simeq}] =_{\mathsf{E}_d} [t_i]_{\simeq}$ and therefore we conclude that

$$f([\mathsf{can}(s_1)]^d_{\simeq}, \ldots, [\mathsf{can}(s_k)]^d_{\simeq}) =_{\mathsf{E}_d} [t_i]_{\simeq}. \tag{A.7}$$

From Equation (A.6) and Equation (A.7), we immediately obtain what we had to prove:

$$[\mathsf{can}(f(s_1, \ldots, s_k))]^d_{\simeq} =_{\mathsf{E}_d} f([\mathsf{can}(s_1)]^d_{\simeq}, \ldots, [\mathsf{can}(s_k)]^d_{\simeq}).$$

□

Using the above technical lemma, we can now prove:

Lemma A.14. *If C is a pure d-context for some $d \in \{a, b, c\}$ and $s_1, \ldots, s_k$ are canonicalized terms (i.e. $s_i = \mathsf{can}(s_i)$ for all $1 \leq i \leq k$), we have that*

$$[C[s_1, \ldots, s_k]]^d_{\simeq} =_{\mathsf{E}_d} [\mathsf{can}(C[s_1, \ldots, s_k])]^d_{\simeq}.$$

Proof. By induction on C:

1. if $C = a$ is an atom, then $C[s_1, \ldots, s_k] = a$ and $[C[s_1, \ldots, s_k]]^d_{\simeq} = [a]^d_{\simeq} = a$ since $C = a$ is in the d-domain. Moreover $\mathsf{can}(C[s_1, \ldots, s_k]) = \mathsf{can}(a) = a$ and therefore $[\mathsf{can}(C[s_1, \ldots, s_k])]^d_{\simeq} = a$. As both the left-hand side $[C[s_1, \ldots, s_k]]^d_{\simeq}$ and right-hand side $[\mathsf{can}(C[s_1, \ldots, s_k])]^d_{\simeq}$ are equal to a, we immediately conclude that they are equal modulo E_d:
$$[C[s_1, \ldots, s_k]]^d_{\simeq} =_{\mathsf{E}_d} [\mathsf{can}(C[s_1, \ldots, s_k])]^d_{\simeq}.$$

2. if $C = \_^i$ for some $1 \leq i \leq k$ is a hole, we have that $C[s_1, \ldots, s_k] = s_i$ and that $\mathsf{can}(C[s_1, \ldots, s_k]) = \mathsf{can}(s_i) = s_i$ (since s_i are canonicalized by hypothesis). Therefore the right-hand side $[C[s_1, \ldots, s_k]]^d_{\simeq} = [s_i]^d_{\simeq}$ and the left-hand side $[\mathsf{can}(C[s_1, \ldots, s_k])]^d_{\simeq} = [s_i]^d_{\simeq}$ are the same term $[s_i]^d_{\simeq}$. We immediately conclude that they are equal modulo E_d:
$$[C[s_1, \ldots, s_k]]^d_{\simeq} =_{\mathsf{E}_d} [\mathsf{can}(C[s_1, \ldots, s_k])]^d_{\simeq}.$$

3. if $C = f(C_1, \ldots, C_l)$, we have that $f \in \mathcal{F}_d$. By the induction hypothesis we have that
$$[C_i[s_1, \ldots, s_k]]^d_{\simeq} =_{\mathsf{E}_d} [\mathsf{can}(C_i[s_1, \ldots, s_k])]^d_{\simeq} \text{ for all } 1 \leq i \leq l. \tag{A.8}$$
We have that
$$\begin{array}{rcl} [\mathsf{can}(C[s_1, \ldots, s_k])]^d_{\simeq} & = & [\mathsf{can}(f(C_1[s_1, \ldots, s_k], \ldots, C_l[s_1, \ldots, s_k]))]^d_{\simeq} \\ & & \text{(by choice of } C_1, \ldots, C_l) \\ & =_{\mathsf{E}_d} & f([\mathsf{can}(C_1[s_1, \ldots, s_k])]^d_{\simeq}, \ldots, \mathsf{can}(C_l[s_1, \ldots, s_k]))]^d_{\simeq} \\ & & \text{(by Lemma A.13)} \\ & =_{\mathsf{E}_d} & f([C_1[s_1, \ldots, s_k]]^d_{\simeq}, \ldots, [C_l[s_1, \ldots, s_k]]^d_{\simeq}) \\ & & \text{(by Equation (A.8))} \\ & = & [f(C_1[s_1, \ldots, s_k], \ldots, C_l[s_1, \ldots, s_k])]^d_{\simeq} \\ & & \text{(by Lemma A.1)} \\ & = & [C[s_1, \ldots, s_k]]^d_{\simeq} \\ & & \text{(by choice of } C_1, \ldots, C_l) \end{array}$$
which implies:
$$[C[s_1, \ldots, s_k]]^d_{\simeq} =_{\mathsf{E}_d} [\mathsf{can}(C[s_1, \ldots, s_k])]^d_{\simeq}.$$

□

We obtain the following corollary, which is just another way to state the lemma above.

Corollary A.2. *If s is a pure d-term for some $d \in \{a, b, c\}$ and σ is a canonicalized substitution (i.e. $\mathsf{can}(\sigma) = \sigma$), then*

$$[s\sigma]^d_{\simeq} =_{\mathsf{E}_d} [\mathsf{can}(s\sigma)]^d_{\simeq}.$$

Proof. Let C be a context obtained from s by replacing all variables $x \in \mathcal{D}om(\sigma)$ with holes $\_^i$ $(1 \leq i \leq n)$ such that $s\sigma = C[x_1\sigma, \ldots, x_n\sigma]$ where $\mathcal{D}om(\sigma) = \{x_1, \ldots, x_n\}$.

Because $\mathsf{can}(\sigma) = \sigma$, it follows that $x_i\sigma$ is canonicalized for all $1 \leq i \leq n$. By Lemma A.14, we then have that

$$[C[x_1\sigma, \ldots, x_n\sigma]]^d_{\simeq} =_{\mathsf{E}_d} [\mathsf{can}(C[x_1\sigma, \ldots, x_n\sigma])]^d_{\simeq}.$$

But this is equivalent to $[s\sigma]^d_{\simeq} =_{\mathsf{E}_d} [\mathsf{can}(s\sigma)]^d_{\simeq}$, which is what we had to prove.

□

We now need a lemma to characterize the canonical form of a non-atomic term.

Lemma A.15. *Let d be a domain $d \in \{a, b\}$, let $f \in \mathcal{F}_d$ be a function symbol of arity $\mathsf{ar}(f) = k$ and let $t_1, \ldots, t_k$ be terms. Then*

$$\mathsf{can}(f(t_1, \ldots, t_k)) = \begin{cases} f(\mathsf{can}(t_1), \ldots, \mathsf{can}(t_k)) & \text{if } C[[s_1]_{\simeq}, \ldots, [s_l]_{\simeq}] \neq_{\mathsf{E}_d} [s_i] \\ & \text{for any } 1 \leq x \leq l \\ s_y & \text{if } C[[s_1]_{\simeq}, \ldots, [s_l]_{\simeq}] =_{\mathsf{E}_d} [s_x]_{\simeq} \\ & \text{for some } 1 \leq x \leq l \\ & \text{for some } 1 \leq y \leq l \text{ such that } [s_x]_{\simeq} = [s_y]_{\simeq}, \end{cases}$$

where $C, s_1, \ldots, s_l$ are such that $f(\mathsf{can}(t_1), \ldots, \mathsf{can}(t_k)) = C[[s_1, \ldots, s_l]]$.

Proof. We distinguish the two cases:

1. First assume that $C[[s_1]_{\simeq}, \ldots, [s_l]_{\simeq}] =_{\mathsf{E}_d} [s_x]_{\simeq}$ for some $1 \leq x \leq l$. We will show that $\mathsf{can}(f(t_1, \ldots, t_k)) = s_y$ for some $1 \leq y \leq l$ such that $[s_x]_{\simeq} = [s_y]_{\simeq}$.

   Let $n_1, \ldots, n_l$ be fresh names such that $n_i = n_j$ iff $s_i \simeq s_j$ for all $1 \leq i, j \leq l$. Let $\sigma = \{[s_i]_{\simeq} \mapsto n_i\}_{1 \leq i \leq k}$ be a replacement (σ is well-defined since $[s_i]_{\simeq} = [s_j]_{\simeq}$ implies $n_i = n_j$ for all $1 \leq i, j \leq l$). As E_d is stable by replacement of atoms by terms, we have that $C[[s_1]_{\simeq}, \ldots, [s_l]_{\simeq}]\sigma =_{\mathsf{E}_d} [s_x]_{\simeq}\sigma$, which is equivalent to $C[n_1, \ldots, n_l] = n_x$. Therefore, by the definition of can, we have that $\mathsf{can}(f(t_1, \ldots, t_k)) = s_y$ for some $1 \leq y \leq l$ such that $n_x = n_y$. But $n_x = n_y$ implies $s_x \simeq s_y$, which concludes what we wanted to show.

2. Next assume that

   $$C[[s_1]_{\simeq}, \ldots, [s_l]_{\simeq}] \neq_{\mathsf{E}_d} [s_x]_{\simeq} \text{ for any } 1 \leq x \leq l. \tag{A.9}$$

   We will show that $\mathsf{can}(f(t_1, \ldots, t_k)) = f(\mathsf{can}(t_1), \ldots, \mathsf{can}(t_k))$.

   Let $n_1, \ldots, n_l$ be fresh names such that $n_i = n_j$ iff $s_i \simeq s_j$ for all $1 \leq i, j \leq l$. We will show by contradiction that $C[n_1, \ldots, n_l] \neq_{\mathsf{E}_d} n_x$ for any $1 \leq x \leq l$. Assume by contradiction that $C[n_1, \ldots, n_l] =_{\mathsf{E}_d} n_x$ for some $1 \leq x \leq l$. We consider the replacement $\sigma = \{n_i \mapsto [s_i]_{\simeq}\}$. The replacement is well-defined since $n_i = n_j$ implies $[s_i]_{\simeq} = [s_j]_{\simeq}$. Since the equational theory E_d is stable by replacement of atoms by terms, we obtain that $C[n_1, \ldots, n_l]\sigma =_{\mathsf{E}_d} n_x\sigma$, i.e. $C[[s_1]_{\simeq}, \ldots, [s_l]_{\simeq}] =_{\mathsf{E}_d} [s_x]_{\simeq}$. But this is false by Equation (A.9). Therefore our assumption was false and we have that $C[n_1, \ldots, n_l] \neq_{\mathsf{E}_d} n_x$ for any $1 \leq x \leq l$.

   Therefore, by the definition of can, we have that $\mathsf{can}(f(t_1, \ldots, t_k)) = f(\mathsf{can}(t_1), \ldots, \mathsf{can}(t_k))$, which is what we wanted to show.

□

Lemma A.16. *Let s be a term such that $\mathsf{can}(s) = C[[t_1, \ldots, t_k]]$ with $\mathsf{domain}(C) = d \in \{a, b, c\}$. Then*

$$C[[t_1]_\simeq, \ldots, [t_k]_\simeq] \neq_{\mathsf{E}_d} [t_x]_\simeq \text{ for any } 1 \leq x \leq k.$$

Proof. By the definition of $C[[t_1, \ldots, t_k]]$, C cannot be a hole $\_^i$ for any $1 \leq i \leq k$. Therefore we distinguish two cases:

1. if $C = a$ is an atom, then $C[[t_1, \ldots, t_k]] = a \neq_{\mathsf{E}_d} [t_x]_\simeq$ for any $1 \leq x \leq k$ by Lemma 6.1.
2. if $C = f(C_1, \ldots, C_l)$ for some contexts $C_1, \ldots, C_l$, let $\tilde{t} = t_1, \ldots, t_k$ denote the sequence of terms $t_1, \ldots, t_k$. We have that $\mathsf{can}(s) = f(C_1[\tilde{t}], \ldots, C_l[\tilde{t}])$.

   By Lemma A.10, we have that $\mathsf{can}(C[t_1, \ldots, t_k]) = \mathsf{can}(\mathsf{can}(s)) = \mathsf{can}(s) = C[t_1, \ldots, t_k]$. Therefore $\mathsf{can}(C[\tilde{t}]) = \mathsf{can}(f(C_1[\tilde{t}], \ldots, C_l[\tilde{t}])) = f(C_1[\tilde{t}], \ldots, C_l[\tilde{t}])$.

   Since $C_i[\tilde{t}]$ is a subterm of $C[\tilde{t}]$, by Lemma A.10 we have $\mathsf{can}(C_i[\tilde{t}]) = C_i[\tilde{t}]$ for all $1 \leq i \leq l$. Therefore $\mathsf{can}(f(C_1[\tilde{t}], \ldots, C_l[\tilde{t}])) = f(C_1[\tilde{t}], \ldots, C_l[\tilde{t}]) = f(\mathsf{can}(C_1[\tilde{t}]), \ldots, \mathsf{can}(C_l[\tilde{t}]))$. By Lemma A.15, we have therefore that $C[[t_1]_\simeq, \ldots, [t_k]_\simeq] \neq_{\mathsf{E}_d} [t_x]_\simeq$ for any $1 \leq x \leq k$, which is what we wanted to prove.

□

Lemma A.17. *Let $d \in \{a, b, c\}$ be a domain, let C be a pure d-context and let $t_1, \ldots, t_k$ be terms in canonical form such that $domain(t_i) \neq d$ for any $1 \leq i \leq k$. Then*

$$\mathsf{can}(C[t_1, \ldots, t_k]) = D[t_1, \ldots, t_k]$$

for some pure d-context D such that $C[[t_1]_\simeq, \ldots, [t_k]_\simeq] =_{\mathsf{E}_d} D[[t_1]_\simeq, \ldots, [t_k]_\simeq]$.

Proof. We will proceed by well-founded induction on C. We distinguish three cases:

1. if $C = a$ is an atom then $C[t_1, \ldots, t_k] = a$ and $\mathsf{can}(C[t_1, \ldots, t_k]) = \mathsf{can}(a) = a = C[t_1, \ldots, t_k]$. We conclude by choosing $D = C$: $\mathsf{can}(C[t_1, \ldots, t_k]) = D[t_1, \ldots, t_k]$ and $C[[t_1]_\simeq, \ldots, [t_k]_\simeq] = a =_{\mathsf{E}_d} a = D[[t_1]_\simeq, \ldots, [t_k]_\simeq]$.
2. if $C = \_^i$ for some $1 \leq i \leq k$, we have that $\mathsf{can}(C[t_1, \ldots, t_k]) = \mathsf{can}(t_i) = t_i = C[t_1, \ldots, t_k]$. We trivially conclude by choosing $D = C = \_^i$: $\mathsf{can}(C[t_1, \ldots, t_k]) = D[t_1, \ldots, t_k]$ and $C[[t_1]_\simeq, \ldots, [t_k]_\simeq] = [t_i]_\simeq =_{\mathsf{E}_d} [t_i]_\simeq = D[[t_1]_\simeq, \ldots, [t_k]_\simeq]$.
3. if $C = f(C_1, \ldots, C_l)$ for a function symbol f and contexts $C_1, \ldots, C_l$, let $\tilde{t} = t_1, \ldots, t_k$ denote the sequence of terms $t_1, \ldots, t_k$ and $[\tilde{t}]_\simeq = [t_1]_\simeq, \ldots, [t_k]_\simeq$ denote the sequence of terms $[t_1]_\simeq, \ldots, [t_k]_\simeq$. We have by the induction hypothesis that there exist pure d-contexts E_i $(1 \leq i \leq l)$ such that $\mathsf{can}(C_i[\tilde{t}]) = E_i[\tilde{t}]$ and $C_i[[\tilde{t}]_\simeq] =_{\mathsf{E}_d} E_i[[\tilde{t}]_\simeq]$ for all $1 \leq i \leq l$.

   We then have that $f(\mathsf{can}(C_1[\tilde{t}]), \ldots, \mathsf{can}(C_l[\tilde{t}])) = f(E_1[\tilde{t}], \ldots, E_l[\tilde{t}])$. Let $E = f(E_1, \ldots, E_l)$. We have that $f(\mathsf{can}(C_1[\tilde{t}]), \ldots, \mathsf{can}(C_l[\tilde{t}])) = E[[\tilde{t}]]$ by the choice of E.

   We distinguish two cases:

   (a) if $E[[t_1]_\simeq, \ldots, [t_k]_\simeq] =_{\mathsf{E}_d} [t_x]_\simeq$ for some $1 \leq x \leq k$, then by Lemma A.15 we have that $\mathsf{can}(f(C_1[\tilde{t}], \ldots, C_l[\tilde{t}])) = t_y$ for some $1 \leq y \leq k$ such that $t_y \simeq t_x$. If we choose $D = \_^y$, we immediately obtain our conclusion:

   $$\mathsf{can}(C[t_1, \ldots, t_k]) = t_y = D[t_1, \ldots, t_k]$$

   and, as $C_i[[\tilde{t}]_\simeq] =_{\mathsf{E}_d} E_i[[\tilde{t}]_\simeq]$ for all $1 \leq i \leq l$ by the induction hypothesis,

   $$C[[\tilde{t}]_\simeq] = f(C_1[[\tilde{t}]_\simeq], \ldots, C_l[[\tilde{t}]_\simeq]) =_{\mathsf{E}_d} f(E_1[[\tilde{t}]_\simeq], \ldots, E_l[[\tilde{t}]_\simeq]) = E[[\tilde{t}]_\simeq] =_{\mathsf{E}_d} [t_x]_\simeq = [t_y]_\simeq = D[[\tilde{t}]_\simeq].$$

(b) if $E[[t_1]_\simeq, \ldots, [t_k]_\simeq] \neq_{\mathsf{E}_d} [t_x]_\simeq$ for any $1 \leq x \leq k$, then by Lemma A.15 we have that $\mathsf{can}(f(C_1[\tilde{t}], \ldots, C_l[\tilde{t}])) = E[\tilde{t}]$. If we choose $D = E$, we immediately conclude:

$$\mathsf{can}(C[t_1, \ldots, t_k]) = D[t_1, \ldots, t_k]$$

and, as $C_i[[\tilde{t}]_\simeq] =_{\mathsf{E}_d} E_i[[\tilde{t}]_\simeq]$ for all $1 \leq i \leq l$ by the induction hypothesis,

$$C[[\tilde{t}]_\simeq] = f(C_1[[\tilde{t}]_\simeq], \ldots, C_l[[\tilde{t}]_\simeq]) =_{\mathsf{E}_d} f(E_1[[\tilde{t}]_\simeq], \ldots, E_l[[\tilde{t}]_\simeq]) = E[[\tilde{t}]_\simeq] = D[[\tilde{t}]_\simeq].$$

□

Lemma A.18. *Let $d \in \{a, b, c\}$ be a domain, let C be a pure d-context and let $t_1, \ldots, t_k$ be terms in canonical form such that $\mathsf{domain}(t_i) \neq d$ for any $1 \leq i \leq k$. Then*

$$\mathsf{can}(C[t_1, \ldots, t_k]) \simeq \begin{cases} t_x & \text{if } C[[t_1]_\simeq, \ldots, [t_k]_\simeq] =_{\mathsf{E}_d} [t_x]_\simeq \\ & \quad \text{for some } 1 \leq x \leq k \\ C[t_1, \ldots, t_k] & \text{if } C[[t_1]_\simeq, \ldots, [t_k]_\simeq] \neq_{\mathsf{E}_d} [t_x]_\simeq \\ & \quad \text{for any } 1 \leq x \leq k. \end{cases}$$

Proof. By Lemma A.17, we have that there exists a pure D-context such that $\mathsf{can}(C[t_1, \ldots, t_k]) = D[t_1, \ldots, t_k]$ and such that $C[[t_1]_\simeq, \ldots, [t_k]_\simeq] =_{\mathsf{E}_d} D[[t_1]_\simeq, \ldots, [t_k]_\simeq]$. We distinguish between two cases:

1. if $C[[t_1]_\simeq, \ldots, [t_k]_\simeq] =_{\mathsf{E}_d} [t_x]_\simeq$ for some $1 \leq x \leq k$, then $D[[t_1]_\simeq, \ldots, [t_k]_\simeq] =_{\mathsf{E}_d} [t_x]_\simeq$ as well. We will show by contradiction that $D = \_^y$ is a hole $\_^y$ for some $1 \leq y \leq l$. Assume by contradiction that D is not such a hole. Then by Lemma A.16 we have that $D[[t_1]_\simeq, \ldots, [t_k]_\simeq] \neq_{\mathsf{E}_d} [t_x]_\simeq$, which is not the case. Therefore our assumption was false and $D = \_^y$ is a hole for some $1 \leq y \leq l$. Then $D[[t_1]_\simeq, \ldots, [t_k]_\simeq] = [t_y]_\simeq$. As $D[[t_1]_\simeq, \ldots, [t_k]_\simeq] =_{\mathsf{E}_d} [t_x]_\simeq$, we have that $[t_y]_\simeq =_{\mathsf{E}_d} [t_x]_\simeq$. By Lemma 6.1, we obtain that $[t_y]_\simeq = [t_x]_\simeq$, which implies $t_x \simeq t_y$.

   We have that $\mathsf{can}(C[t_1, \ldots, t_k]) = D[t_1, \ldots, t_k] = t_y \simeq t_x$, which is what we had to show.

2. if $C[[t_1]_\simeq, \ldots, [t_k]_\simeq] \neq_{\mathsf{E}_d} [t_x]_\simeq$ for some $1 \leq x \leq k$, then $D[[t_1]_\simeq, \ldots, [t_k]_\simeq] \neq_{\mathsf{E}_d} [t_x]_\simeq$ for all $1 \leq x \leq k$ as well.

   Therefore D cannot be a hole $\_^x$ for some $1 \leq x \leq k$. This implies $\mathsf{domain}(C[t_1, \ldots, t_k]) = \mathsf{domain}(D[t_1, \ldots, t_k]) = d$. As $[C[t_1, \ldots, t_k]]^d_\simeq = C[[t_1]_\simeq, \ldots, [t_k]_\simeq] =_{\mathsf{E}_d} D[[t_1]_\simeq, \ldots, [t_k]_\simeq] = [D[t_1, \ldots, t_k]]^d_\simeq$, we obtain by Lemma A.6 that $D[t_1, \ldots, t_k] \simeq C[t_1, \ldots, t_k]$. But we have $\mathsf{can}(C[t_1, \ldots, t_k]) = D[t_1, \ldots, t_k]$ and therefore $\mathsf{can}(C[t_1, \ldots, t_k]) \simeq C[t_1, \ldots, t_k]$, which is what we had to show.

□

A.1.4 Main Proof

We have now proved all technical results and we are ready to get into the main proof. We show that a rewrite step performed at the top of a term preserves the $\simeq$ relation between the canonical forms.

Lemma A.19. *If $l \approx r \in \mathsf{E}$ is an identity and σ is a substitution, we have that $\mathsf{can}(l\sigma) \simeq \mathsf{can}(r\sigma)$.*

Proof. As $l \approx r \in \mathsf{E}$, we have that there exists $d \in \{a, b\}$ such that $l \approx r \in \mathsf{E}_d$.
We have that

$$\begin{array}{rcll} [\mathsf{can}(l\sigma)]^d_\simeq & = & [\mathsf{can}(l\mathsf{can}(\sigma))]^d_\simeq & \text{(Corollary A.1)} \\ & =_{\mathsf{E}_d} & [l\mathsf{can}(\sigma)]^d_\simeq & \text{(Corollary A.2)} \\ & = & l[\mathsf{can}(\sigma)]^d_\simeq & \text{(Definition of } [\cdot]^d_\simeq). \end{array}$$

We obtained that $[\mathsf{can}(l\sigma)]^d_\simeq = l[\mathsf{can}(\sigma)]^d_\simeq$. By a similar reasoning, we have that $[\mathsf{can}(r\sigma)]^d_\simeq =_{\mathsf{E}_d} r[\mathsf{can}(\sigma)]^d_\simeq$. But as $l \approx r \in \mathsf{E}_d$, trivially have that $r[\mathsf{can}(\sigma)]^d_\simeq =_{\mathsf{E}_d} l[\mathsf{can}(\sigma)]^d_\simeq$. Therefore we conclude that

$$[\mathsf{can}(l\sigma)]^d_\simeq =_{\mathsf{E}_d} [\mathsf{can}(r\sigma)]^d_\simeq. \tag{A.10}$$

Let $t_1, \ldots, t_k$ be terms with $\mathsf{domain}(t_i) \neq d$ (for any $1 \leq i \leq k$) such that

$$x(\mathsf{can}(\sigma)) = C_x[t_1, \ldots, t_k]$$

for some pure d-context C_x for any variable $x \in \mathcal{D}om(\sigma)$. Then there exist pure d-contexts C, D such that $l\mathsf{can}(\sigma) = C[t_1, \ldots, t_k]$, $r\mathsf{can}(\sigma) = D[t_1, \ldots, t_k]$, $[l\mathsf{can}(\sigma)]^d_\simeq = C[[t_1]_\simeq, \ldots, [t_k]_\simeq]$ and $[r\mathsf{can}(\sigma)]^d_\simeq = D[[t_1]_\simeq, \ldots, [t_k]_\simeq]$.

We will show that $C[[t_1]_\simeq, \ldots, [t_k]_\simeq] =_{\mathsf{E}_d} D[[t_1]_\simeq, \ldots, [t_k]_\simeq]$. Indeed, as l and r a pure d-terms, we have by Lemma A.1 that $[l\mathsf{can}(\sigma)]^d_\simeq = l[\mathsf{can}(\sigma)]^d_\simeq$ and that $[r\mathsf{can}(\sigma)]^d_\simeq = r[\mathsf{can}(\sigma)]^d_\simeq$. As $l \approx r \in \mathsf{E}_d$, we have that $l[\mathsf{can}(\sigma)]^d_\simeq =_{\mathsf{E}_d} r[\mathsf{can}(\sigma)]^d_\simeq$. Therefore $[l\mathsf{can}(\sigma)]^d_\simeq =_{\mathsf{E}_d} [r\mathsf{can}(\sigma)]^d_\simeq$. But $[l\mathsf{can}(\sigma)]^d_\simeq = C[[t_1]_\simeq, \ldots, [t_k]_\simeq]$ and $[r\mathsf{can}(\sigma)]^d_\simeq = D[[t_1]_\simeq, \ldots, [t_k]_\simeq]$ and therefore

$$C[[t_1]_\simeq, \ldots, [t_k]_\simeq] =_{\mathsf{E}_d} D[[t_1]_\simeq, \ldots, [t_k]_\simeq].$$

We will now show that $\mathsf{domain}(\mathsf{can}(l\mathsf{can}(\sigma))) = \mathsf{domain}(\mathsf{can}(r\mathsf{can}(\sigma)))$. We distinguish between two cases:

1. if $C[[t_1]_\simeq, \ldots, [t_k]_\simeq] =_{\mathsf{E}_d} [t_x]_\simeq$ or if $D[[t_1]_\simeq, \ldots, [t_k]_\simeq] =_{\mathsf{E}_d} [t_x]_\simeq$ for some $1 \leq x \leq k$, we have that
$$C[[t_1]_\simeq, \ldots, [t_k]_\simeq] =_{\mathsf{E}_d} [t_x]_\simeq =_{\mathsf{E}_d} D[[t_1]_\simeq, \ldots, [t_k]_\simeq]$$
since we have already shown $C[[t_1]_\simeq, \ldots, [t_k]_\simeq] =_{\mathsf{E}_d} D[[t_1]_\simeq, \ldots, [t_k]_\simeq]$.

   By Lemma A.18 we obtain that $\mathsf{can}(l\mathsf{can}(\sigma)) = \mathsf{can}(C[t_1, \ldots, t_k]) \simeq t_x$ and that $\mathsf{can}(r\mathsf{can}(\sigma)) = \mathsf{can}(D[t_1, \ldots, t_k]) \simeq t_x$. Therefore we have that $\mathsf{domain}(\mathsf{can}(l\mathsf{can}(\sigma))) = \mathsf{domain}(t_x)$ and that $\mathsf{domain}(\mathsf{can}(r\mathsf{can}(\sigma))) = \mathsf{domain}(t_x)$ which immediately implies what we want to show: $\mathsf{domain}(\mathsf{can}(l\mathsf{can}(\sigma))) = \mathsf{domain}(\mathsf{can}(r\mathsf{can}(\sigma)))$.

2. otherwise, $C[[t_1]_\simeq, \ldots, [t_k]_\simeq] \neq_{\mathsf{E}_d} [t_x]_\simeq$ and $D[[t_1]_\simeq, \ldots, [t_k]_\simeq] \neq_{\mathsf{E}_d} [t_x]_\simeq$ for any $1 \leq x \leq k$. By Lemma A.18, we obtain that $\mathsf{can}(l\mathsf{can}(\sigma)) = \mathsf{can}(C[t_1, \ldots, t_k]) \simeq C[t_1, \ldots, t_k]$ and that $\mathsf{can}(r\mathsf{can}(\sigma)) = \mathsf{can}(D[t_1, \ldots, t_k]) \simeq D[t_1, \ldots, t_k]$.

   Neither of the contexts C and D can be a hole $\_^i$ (for any $1 \leq i \leq k$) since otherwise $C[[t_1]_\simeq, \ldots, [t_k]_\simeq] = [t_i]_\simeq$ or $D[[t_1]_\simeq, \ldots, [t_k]_\simeq] = [t_i]_\simeq$, which would contradict our assumption.

   Therefore $\mathsf{domain}(C[t_1, \ldots, t_k]) = \mathsf{domain}(D[t_1, \ldots, t_k]) = d$. As $\mathsf{can}(l\mathsf{can}(\sigma)) \simeq C[t_1, \ldots, t_k]$ and $\mathsf{can}(r\mathsf{can}(\sigma)) \simeq D[t_1, \ldots, t_k]$, we obtain $\mathsf{domain}(\mathsf{can}(l\mathsf{can}(\sigma))) = \mathsf{domain}(\mathsf{can}(r\mathsf{can}(\sigma)))$, which is what we wanted to prove.

In either case, we have shown that $\mathsf{domain}(\mathsf{can}(l\mathsf{can}(\sigma))) = \mathsf{domain}(\mathsf{can}(r\mathsf{can}(\sigma)))$. As we have $\mathsf{can}(l\mathsf{can}(\sigma)) = \mathsf{can}(l\sigma)$ and $\mathsf{can}(r\mathsf{can}(\sigma)) = \mathsf{can}(r\sigma)$ by Corollary A.1, we obtain that

$$\mathsf{domain}(\mathsf{can}(r\sigma)) = \mathsf{domain}(\mathsf{can}(l\sigma)). \tag{A.11}$$

From Equation (A.10) and Equation (A.11), we obtain by Lemma A.6 that $\mathsf{can}(l\sigma) \simeq \mathsf{can}(r\sigma)$, which is what we had to prove.

□

Lemma A.20. *If $s_1 \simeq t_1, \ldots, s_k \simeq t_k$ and $s_1, \ldots, s_k, t_1, \ldots, t_k$ are terms in canonical form, then $\mathsf{can}(f(s_1, \ldots, s_k)) \simeq \mathsf{can}(f(t_1, \ldots, t_k))$.*

Proof. Let the contexts C, D and the terms $u_1, \ldots, u_l$ be such that $f(s_1, \ldots, s_k) = C[[u_1, \ldots, u_l]]$ and $f(t_1, \ldots, t_k) = D[[u_1, \ldots, u_l]]$. Let $d = \mathsf{domain}(C) = \mathsf{domain}(D) = \mathsf{domain}(f)$ be the domain of the function symbol f. By Lemma A.15, we distinguish between the following situations:

1. if $C[[u_1]_{\simeq},\ldots,[u_l]_{\simeq}] =_{\mathsf{E}_d} [u_x]_{\simeq}$ for some $1 \leq x \leq l$ we have that $\mathsf{can}(f(s_1,\ldots,s_k)) = u_y$ for some $1 \leq y \leq l$ such that $u_y \simeq u_x$. This implies $\mathsf{can}(f(s_1,\ldots,s_k)) \simeq u_x$.

   By Lemma A.8, we have that $f(s_1,\ldots,s_k) \simeq f(t_1,\ldots,t_k)$ and by Lemma A.6 we have that $[f(s_1,\ldots,s_k)]^d_{\simeq} =_{\mathsf{E}_d} [f(t_1,\ldots,t_k)]^d_{\simeq}$. But $[f(s_1,\ldots,s_k)]^d_{\simeq} = C[[u_1]_{\simeq},\ldots,[u_l]_{\simeq}] =_{\mathsf{E}_d} [u_x]_{\simeq}$. Therefore $[f(t_1,\ldots,t_k)]^d_{\simeq} =_{\mathsf{E}_d} [u_x]_{\simeq}$ and, by Lemma A.15, $\mathsf{can}(f(t_1,\ldots,t_k)) = u_z$ for some $1 \leq z \leq l$ such that $u_z \simeq u_x$. As we already have that $\mathsf{can}(f(s_1,\ldots,s_k)) \simeq u_x$, we obtain $\mathsf{can}(f(t_1,\ldots,t_k)) \simeq \mathsf{can}(f(s_1,\ldots,s_k))$.

2. if $C[[u_1]_{\simeq},\ldots,[u_l]_{\simeq}] \neq_{\mathsf{E}_d} [u_x]_{\simeq}$ for any $1 \leq x \leq l$ we have that $\mathsf{can}(f(s_1,\ldots,s_k)) = f(\mathsf{can}(s_1),\ldots,\mathsf{can}(s_k))$.

   By Lemma A.8, we have that $f(s_1,\ldots,s_k) \simeq f(t_1,\ldots,t_k)$ and by Lemma A.6 we have that $[f(s_1,\ldots,s_k)]^d_{\simeq} =_{\mathsf{E}_d} [f(t_1,\ldots,t_k)]^d_{\simeq}$. But $[f(s_1,\ldots,s_k)]^d_{\simeq} = C[[u_1]_{\simeq},\ldots,[u_l]_{\simeq}] \neq_{\mathsf{E}_d} [u_x]_{\simeq}$ for any $1 \leq x \leq l$. Therefore by Lemma A.15, $\mathsf{can}(f(t_1,\ldots,t_k)) = f(\mathsf{can}(t_1),\ldots,\mathsf{can}(t_k))$.

   As we have that $\mathsf{can}(f(s_1,\ldots,s_k)) = f(\mathsf{can}(s_1),\ldots,\mathsf{can}(s_k))$ and also $\mathsf{can}(f(t_1,\ldots,t_k)) = f(\mathsf{can}(t_1),\ldots,\mathsf{can}(t_k))$, we immediately conclude that $\mathsf{can}(f(s_1,\ldots,s_k)) \simeq \mathsf{can}(f(t_1,\ldots,t_k))$.

□

Corollary A.3. *If* $\mathsf{can}(s) \simeq \mathsf{can}(t)$*, then* $\mathsf{can}(C[s]) \simeq \mathsf{can}(C[t])$.

Proof. We show by induction on C that $\mathsf{can}(C[\mathsf{can}(s)]) \simeq \mathsf{can}(C[\mathsf{can}(t)])$. We distinguish among the following cases:

1. if $C = a$ is an atom then $C[\mathsf{can}(s)] = C[\mathsf{can}(t)] = a$ and $\mathsf{can}(C[\mathsf{can}(s)]) = a = \mathsf{can}(C[\mathsf{can}(t)])$ from where we immediately conclude.

2. if $C = \_^1$ is the empty context, we have that $C[\mathsf{can}(s)] = \mathsf{can}(s)$ and that $C[\mathsf{can}(t)] = \mathsf{can}(t)$. Therefore $\mathsf{can}(C[\mathsf{can}(s)]) = \mathsf{can}(\mathsf{can}(s))$ and $\mathsf{can}(C[\mathsf{can}(t)]) = \mathsf{can}(\mathsf{can}(t))$. By Lemma A.10, $\mathsf{can}(\mathsf{can}(s)) = \mathsf{can}(s)$ and $\mathsf{can}(\mathsf{can}(t)) = \mathsf{can}(t)$. Therefore $\mathsf{can}(C[\mathsf{can}(s)]) = \mathsf{can}(s) \simeq \mathsf{can}(t) = \mathsf{can}(C[\mathsf{can}(t)])$, which is what we had to prove.

3. if $C = f(C_1,\ldots,C_k)$, we have by the induction hypothesis that

   $$\mathsf{can}(C_i[\mathsf{can}(s)]) \simeq \mathsf{can}(C_i[\mathsf{can}(t)]) \text{ for all } 1 \leq i \leq k.$$

   Therefore, by Lemma A.20, we have that

   $$\mathsf{can}(f(\mathsf{can}(C_1[\mathsf{col}(s)]),\ldots,\mathsf{can}(C_k[\mathsf{col}(s)]))) \simeq \mathsf{can}(f(\mathsf{can}(C_1[\mathsf{col}(t)]),\ldots,\mathsf{can}(C_k[\mathsf{col}(t)]))),$$

   i.e. $\mathsf{can}(C[\mathsf{col}(s)]) \simeq \mathsf{can}(C[\mathsf{col}(t)])$, which is what we have to prove.

We have shown that $\mathsf{can}(C[\mathsf{can}(s)]) \simeq \mathsf{can}(C[\mathsf{can}(t)])$. By Lemma A.12, we have that $\mathsf{can}(C[s]) = \mathsf{can}(C[\mathsf{can}(s)])$ and that $\mathsf{can}(C[t]) = \mathsf{can}(C[\mathsf{can}(t)])$. As we have shown that $\mathsf{can}(C[\mathsf{can}(s)]) \simeq \mathsf{can}(C[\mathsf{can}(t)])$, it follows that $\mathsf{can}(C[s]) \simeq \mathsf{can}(C[t])$, which is what we had to prove.

□

Proposition A.1. *For two terms* $s, t \in T(\mathcal{F}, \mathcal{A})$*, we have that*

$$s =_{\mathsf{E}} t \text{ iff } \mathsf{can}(s) \simeq \mathsf{can}(t).$$

Proof. The reverse implication, that $\mathsf{can}(s) \simeq \mathsf{can}(t)$ implies $s =_{\mathsf{E}} t$, follows by Lemma A.5 and Lemma A.9.

For the direct implication, as $s =_{\mathsf{E}} t$, it follows that there exist an integer $n \in \mathbb{N}$ and terms $t_1,\ldots,t_n \in T(\mathcal{F},\mathcal{A})$ such that

$$s = t_1 \to_{\mathsf{E}} t_2 \to_{\mathsf{E}} t_3 \ldots \to_{\mathsf{E}} t_n = t.$$

For any $1 \leq i \leq n-1$, we have that $t_i = C_i[l_i\sigma_i]$ and $t_{i+1} = C_i[r_i\sigma_i]$ for some equation $l_i \approx r_i \in \mathsf{E}$ (or $r_i \approx l_i \in \mathsf{E}$), some context C_i and some substitution σ_i.

We now show that $\mathsf{can}(C_i[l_i\sigma_i]) \simeq \mathsf{can}(C_i[r_i\sigma_i])$ for all $1 \leq i \leq n-1$. By Lemma A.19, we have that $\mathsf{can}(l_i\sigma) \simeq \mathsf{can}(r_i\sigma)$. By Corrolary A.3, we have that $\mathsf{can}(C_i[l_i\sigma]) \simeq \mathsf{can}(C_i[r_i\sigma])$. We have shown that $\mathsf{can}(C_i[l_i\sigma_i]) \simeq \mathsf{can}(C_i[r_i\sigma_i])$ for all $1 \leq i \leq n$. But for all $1 \leq i \leq n-1$, either $t_i = C_i[l_i\sigma_i]$ and $t_{i+1} = C_i[r_i\sigma_i]$ or $t_i = C_i[r_i\sigma_i]$ and $t_{i+1} = C_i[l_i\sigma_i]$. Either way, we have that $\mathsf{can}(t_i) \simeq \mathsf{can}(t_{i+1})$ for all $1 \leq i \leq n-1$.

As $s = t_1$ and $t = t_n$ and $\mathsf{can}(t_i) \simeq \mathsf{can}(t_{i+1})$ for all $1 \leq i \leq n-1$, we obtain by transitivity of $\simeq$ that $\mathsf{can}(s) \simeq \mathsf{can}(t)$, which is what we had to show.

□

A.1.5 Linking Canonical and Collapsed Forms

We will now show that col and can are the same function up to $\simeq$.

Lemma A.21. *For any term s, we have that $\mathsf{col}(s) \simeq \mathsf{can}(s)$.*

Proof. By induction on the size of s.

1. if $s = a$ is an atom, then $\mathsf{col}(s) = a = \mathsf{can}(s)$ by definition and therefore $\mathsf{col}(s) \simeq \mathsf{can}(s)$.

2. if $s = f(t_1, \ldots, t_l)$ for some function symbol $f \in \mathcal{F}_d$ of arity $\mathsf{ar}(f) = l$, we have by Lemma A.11, that $\mathsf{can}(f(t_1, \ldots, t_l)) = \mathsf{can}(f(\mathsf{can}(t_1), \ldots, \mathsf{can}(t_l)))$. By the induction hypothesis that $\mathsf{col}(t_i) \simeq \mathsf{can}(t_i)$ for every $1 \leq i \leq l$ and therefore, by Lemma A.20, we have that $\mathsf{can}(f(t_1, \ldots, t_l)) \simeq \mathsf{can}(f(\mathsf{col}(t_1), \ldots, \mathsf{col}(t_l))))$.

   Let $C, s_1, \ldots, s_k$ be such that

   $$f(\mathsf{col}(t_1), \ldots, \mathsf{col}(t_l)) = C[[s_1, \ldots, s_k]]$$

   and let $n_1, \ldots, n_k$ be fresh names such that $n_x = n_y$ iff $s_x \simeq s_y$ for all $1 \leq x, y \leq k$.

   As C cannot be the empty context (by the definition of $C[[s_1, \ldots, s_k]]$), we have that $s_1, \ldots, s_k$ are subterms of the terms $\mathsf{col}(t_1), \ldots, \mathsf{col}(t_l)$, which implies that for all $1 \leq i \leq k$ there exists a $1 \leq j \leq l$ such that $|s_i| \leq |\mathsf{col}(t_j)|$. By Lemma 6.3, we have that for all $1 \leq i \leq l$, $|\mathsf{col}(t_i)| \leq |t_i|$. Therefore for all $1 \leq i \leq k$, there exists a $1 \leq j \leq l$ such that $|s_i| \leq |t_j|$. But t_j $(1 \leq j \leq l)$ are strict subterms of t and therefore we can apply the induction hypothesis on s_i $(1 \leq i \leq k)$ to obtain that $\mathsf{can}(s_i) \simeq \mathsf{col}(s_i)$ for all $1 \leq i \leq k$.

   By Lemma 6.4, as s_i is a subterm of some term $\mathsf{col}(t_1), \ldots, \mathsf{col}(t_l)$, we have that $\mathsf{col}(s_i) = s_i$ for all $1 \leq i \leq k$. Therefore $\mathsf{can}(s_i) \simeq s_i$ for all $1 \leq i \leq k$. By Proposition A.1, we have that $s_x \simeq \mathsf{can}(s_x) \simeq \mathsf{can}(s_y) \simeq s_y$ iff $s_x =_{\mathsf{E}} s_y$ for all $1 \leq x, y \leq k$. Therefore $n_x = n_y$ iff $s_x =_{\mathsf{E}} s_y$.

   We distinguish two situations:

   (a) if $C[[n_1, \ldots, n_k]] =_{\mathsf{E}_d} n_i$ for some i, by definition $\mathsf{can}(f(\mathsf{col}(t_1), \ldots, \mathsf{col}(t_l))) = s_x$ for some index $1 \leq x \leq k$ such that $n_x = n_i$. By the definition of col, we have that $\mathsf{col}(f(\mathsf{col}(t_1), \ldots, \mathsf{col}(t_l))) = s_y$ for some index $1 \leq y \leq k$ such that $n_y = n_i$. As $n_x = n_i = n_y$ we obtain that $s_x \simeq s_y$ by choice of $n_1, \ldots, n_k$, from which we obtain that $\mathsf{can}(f(\mathsf{col}(t_1), \ldots, \mathsf{col}(t_l))) \simeq \mathsf{col}(f(\mathsf{col}(t_1), \ldots, \mathsf{col}(t_l)))$. But we have already seen that

   $$\mathsf{can}(f(t_1, \ldots, t_l)) \simeq \mathsf{can}(f(\mathsf{col}(t_1), \ldots, \mathsf{col}(t_l))))$$

   and therefore $\mathsf{can}(f(t_1, \ldots, t_l))) \simeq \mathsf{col}(f(\mathsf{col}(t_1), \ldots, \mathsf{col}(t_l)))$. By Lemma 6.5 we have that $\mathsf{col}(f(\mathsf{col}(t_1), \ldots, \mathsf{col}(t_l))) = \mathsf{col}(f(t_1, \ldots, t_l))$. Therefore we obtain what we wanted to prove: $\mathsf{can}(f(t_1, \ldots, t_l))) \simeq \mathsf{col}(f(t_1, \ldots, t_l))$.

(b) if $C[[n_1,\ldots,n_k]] \neq_{\mathsf{E}_d} n_i$ for any i, by definition we have $\mathsf{can}(f(\mathsf{col}(t_1),\ldots,\mathsf{col}(t_l))) = f(\mathsf{can}(t_1),\ldots,\mathsf{can}(t_l))$. We already have $\mathsf{can}(f(\mathsf{col}(t_1),\ldots,\mathsf{col}(t_l))) \simeq \mathsf{can}(f(t_1,\ldots,t_l))$. Therefore $\mathsf{can}(f(t_1,\ldots,t_l)) \simeq f(\mathsf{can}(t_1),\ldots,\mathsf{can}(t_l))$.

By the definition of col, we have that $\mathsf{col}(f(\mathsf{col}(t_1),\ldots,\mathsf{col}(t_l))) = f(\mathsf{col}(t_1),\ldots,\mathsf{col}(t_l))$. But by Lemma 6.5, we have that $\mathsf{col}(f(\mathsf{col}(t_1),\ldots,\mathsf{col}(t_l))) = \mathsf{col}(f(t_1,\ldots,t_l))$. Therefore $\mathsf{col}(f(t_1,\ldots,t_l)) = f(\mathsf{col}(t_1),\ldots,\mathsf{col}(t_l))$. By the induction hypothesis $\mathsf{col}(t_i) \simeq \mathsf{can}(t_i)$ for all $1 \leq i \leq l$ and therefore $f(\mathsf{col}(t_1),\ldots,\mathsf{col}(t_l)) \simeq f(\mathsf{can}(t_1),\ldots,\mathsf{can}(t_l))$. Therefore $\mathsf{col}(f(t_1,\ldots,t_l)) \simeq f(\mathsf{can}(t_1),\ldots,\mathsf{can}(t_l))$.

We have already seen that $\mathsf{can}(f(t_1,\ldots,t_l)) \simeq f(\mathsf{can}(t_1),\ldots,\mathsf{can}(t_l))$ and therefore we conclude that $\mathsf{can}(f(t_1,\ldots,t_l)) \simeq \mathsf{col}(f(t_1,\ldots,t_l))$, which is what we wanted to show.

□

Corollary A.4. *For any terms s,t we have that $s =_E t$ iff $\mathsf{col}(s) \simeq \mathsf{col}(t)$.*

Proof. We have that $s =_{\mathsf{E}} t$ iff, by Proposition A.1, $\mathsf{can}(s) \simeq \mathsf{can}(t)$. But $\mathsf{can}(s) \simeq \mathsf{col}(s)$ and $\mathsf{can}(t) \simeq \mathsf{col}(t)$ by Lemma A.21. Therefore $s =_{\mathsf{E}} t$ iff $\mathsf{col}(s) \simeq \mathsf{col}(t)$.

□

We are now ready to prove the fundamental lemma:

Lemma 6.6 (Fundamental Collapse Lemma)**.** *If $s =_{\mathsf{E}} t$, then $\mathsf{col}(s) = C[[s_1,\ldots,s_k]]$, $\mathsf{col}(t) = D[[s_{k+1},\ldots,s_{k+l}]]$ such that $\mathsf{domain}(C) = \mathsf{domain}(D)$ and $C[n_1,\ldots,n_k] =_{\mathsf{E}_d} D[n_{k+1},\ldots,n_{k+l}]$ where $d = \mathsf{domain}(C)$ and $n_1,\ldots,n_{k+l}$ are fresh names such that $n_i = n_j$ iff $s_i =_{\mathsf{E}} s_j$ for all $1 \leq i,j \leq k+l$.*

Proof. If $s =_{\mathsf{E}} t$, we have that $\mathsf{col}(s) \simeq \mathsf{col}(t)$ by Corollary A.4. As $\mathsf{col}(s) \simeq \mathsf{col}(t)$, it follows by Lemma A.6 that $\mathsf{domain}(\mathsf{col}(s)) = \mathsf{domain}(\mathsf{col}(t)) = d$ for some $d \in \{a,b,c\}$ and that $[s]^d_{\simeq} =_{\mathsf{E}_d} [t]^d_{\simeq}$.

Let C, D be contexts such that $\mathsf{col}(s) = C[[s_1,\ldots,s_k]]$ and $\mathsf{col}(t) = D[[s_{k+1},\ldots,s_{k+l}]]$. Let $n_1,\ldots,n_{k+l}$ be fresh names such that $n_x = n_y$ iff $s_x =_{\mathsf{E}} s_y$ for all $1 \leq x,y \leq k+l$. By Corollary A.4, we obtain that $n_x = n_y$ iff $\mathsf{col}(s_x) \simeq \mathsf{col}(s_y)$ for all $1 \leq x,y \leq k+l$.

As $s_1,\ldots,s_{k+l}$ are subterms of $\mathsf{col}(s)$ and $\mathsf{col}(t)$, it follows by Lemma 6.4 that $\mathsf{col}(s_i) = s_i$ for all $1 \leq i \leq k+l$. Therefore we obtain that $n_x = n_y$ iff $s_x \simeq s_y$ for all $1 \leq x,y \leq k+l$.

As $[s]^d_{\simeq} =_{\mathsf{E}_d} [t]^d_{\simeq}$, we have that $C[[s_1]_{\simeq},\ldots,[s_k]_{\simeq}] =_{\mathsf{E}_d} D[[s_{k+1}]_{\simeq},\ldots,[s_{k+l}]_{\simeq}]$. As $n_x = n_y$ iff $s_x \simeq s_y$ (for all $1 \leq x,y \leq k+l$), the replacement $\sigma = \{[s_x] \mapsto n_x\}_{1\leq x\leq k+l}$ is well-defined. As E_d is stable by replacement of atoms by terms, we obtain that $C[[s_1]_{\simeq},\ldots,[s_k]_{\simeq}]\sigma =_{\mathsf{E}_d} D[[s_{k+1}]_{\simeq},\ldots,[s_{k+l}]_{\simeq}]\sigma$. This is equivalent to $C[n_1,\ldots,n_k] =_{\mathsf{E}_d} D[n_{k+1},\ldots,n_{k+l}]$, which is what we wanted to prove.

□

A.2 Proof of Lemma 6.13

This section consists of the proof of Lemma 6.13.

Lemma 6.13. *If P is a trace over the d-domain and*

$$(P,\emptyset,\emptyset) \xrightarrow{\S_1}_{\mathcal{F},\mathsf{E}} (P_1,\varphi_1,\sigma_1) \xrightarrow{\S_2}_{\mathcal{F},\mathsf{E}} \ldots \xrightarrow{\S_n}_{\mathcal{F},\mathsf{E}} (P_n,\varphi_n,\sigma_n)$$

then

$$(P,\emptyset,\emptyset) \xrightarrow{\S'_1}_{\mathcal{F}_d,\mathsf{E}_d} (P_1,\varphi'_1,\sigma'_1) \xrightarrow{\S'_2}_{\mathcal{F}_d,\mathsf{E}_d} \ldots \xrightarrow{\S'_n}_{\mathcal{F}_d,\mathsf{E}_d} (P_n,\varphi'_n,\sigma'_n))$$

for some φ'_n, σ'_n and $\S'_\_$ such that $\varphi_n \vdash_{\mathsf{E}} x\sigma_n$ implies $\varphi'_n \vdash_{\mathsf{E}_d} x\sigma'_n$ and $x\sigma_n =_{\mathsf{E}} y\sigma_n$ implies $x\sigma'_n =_{\mathsf{E}_d} y\sigma'_n$ for all $x,y \in \mathcal{D}om(\sigma_n)$.

Proof. We assume w.l.o.g. that $d = a$ since the case where $d = b$ is analogous. Let init/0 be any constant in $\mathcal{F}_a$. The idea is to obtain the second trace from the first trace by abstracting any elements that starts with a symbol from $\mathcal{F}_b$ by the constant init. This abstraction is formalized by the function abs, which is defined inductively on ground terms as follows:

1. $\mathrm{abs}(f(t_1, \ldots, t_k)) = f(\mathrm{abs}(t_1), \ldots, \mathrm{abs}(t_k))$ if $f \in \mathcal{F}_a$
2. $\mathrm{abs}(f(t_1, \ldots, t_k)) = \mathrm{init}$ if $f \in \mathcal{F}_b$

The function abs enjoys the following two properties:

Claim A.1. *If $s =_{\mathsf{E}} t$ and s and t are collapsed, we have that $\mathrm{abs}(s) =_{\mathsf{E}_a} \mathrm{abs}(t)$.*

Proof. Let $s = C[[s_1, \ldots, s_k]]$ and $t = D[[s_{k+1}, \ldots, s_{k+l}]]$. As $s =_{\mathsf{E}} t$ and s and t are collapsed, by Lemma 6.6, we have that $\mathsf{root}(C)$ and $\mathsf{root}(D)$ come from the same signature $\mathcal{F}_d$. If $d = b$, then $\mathrm{abs}(s) = \mathrm{abs}(t) = \mathrm{init}$ and we are done.

Otherwise $c = a$ and $\mathrm{abs}(s) = C[\mathrm{init}, \ldots, \mathrm{init}]$ and $\mathrm{abs}(t) = D[\mathrm{init}, \ldots, \mathrm{init}]$. By Lemma 6.6, we have that $C[n_1, \ldots, n_k] =_{\mathsf{E}_a} D[n_{k+1}, \ldots, n_{k+l}]$, where $\{n_i\}_{1 \leq i \leq k+l}$ are fresh names such that $n_x = n_y$ iff $s_x =_{\mathsf{E}} s_y$ for all $1 \leq x, y \leq k + l$. As E is stable by replacement of arbitrary terms for names, we have that $C[n_1, \ldots, n_k]\{n_i \mapsto \mathrm{init}\}_{1 \leq i \leq k} =_{\mathsf{E}_a} D[n_{k+1}, \ldots, n_{k+l}]\{n_i \mapsto \mathrm{init}\}_{k+1 \leq i \leq k+l}$ and therefore $\mathrm{abs}(s) =_{\mathsf{E}_a} \mathrm{abs}(t)$.

□

Claim A.2. *If t is a pure $\mathcal{F}_a$ term and σ is a collapsed substitution, we have that $\mathrm{abs}(\mathsf{col}(t\sigma)) =_{\mathsf{E}_a} \mathrm{abs}(t\sigma)$.*

Proof. We do the proof by induction on the size of $t\sigma$. If t is a variable then $\mathsf{col}(t\sigma) = t\sigma$ (since we assumed σ to be collapsed) and we are done. Otherwise, $\mathsf{root}(t) \in D_a$. Let $t = f(t_1, \ldots, t_k)$. We distinguish two cases by the definition of col:

1. either $\mathsf{col}(t\sigma) = \mathsf{col}(f(t_1\sigma, \ldots, t_k\sigma)) = f(\mathsf{col}(t_1\sigma), \ldots, \mathsf{col}(t_k\sigma))$, in which case we can easily conclude by the induction hypothesis
2. or $f(\mathsf{col}(t_1\sigma), \ldots, \mathsf{col}(t_k\sigma)) = C[[s_1, \ldots, s_l]]$ for some context C and some terms $s_1, \ldots, s_l$ such that $C[[n_1, \ldots, n_l]] =_{\mathsf{E}} n_x$ for some $1 \leq x \leq l$. In this case, $\mathsf{col}(t\sigma) = s_x$.

   We have therefore by Claim A.1 that $\mathrm{abs}(\mathsf{col}(t\sigma)) = \mathrm{abs}(s_x) = \mathrm{init}$. By the induction hypothesis, we have that $\mathrm{abs}(t\sigma) =_{\mathsf{E}_a} f(\mathrm{abs}(\mathsf{col}(t_1\sigma)), \ldots, \mathrm{abs}(\mathsf{col}(t_k\sigma)))$. By the definition of abs, the latter is equal to $\mathrm{abs}(f(\mathsf{col}(t_1\sigma), \ldots, \mathsf{col}(t_k\sigma)))$. Therefore $\mathrm{abs}(t\sigma) = \mathrm{abs}(f(\mathsf{col}(t_1\sigma), \ldots, \mathsf{col}(t_k\sigma)))$ and we obtain $\mathrm{abs}(t\sigma) = \mathrm{abs}(C[s_1, \ldots, s_l]) = C[\mathrm{init}, \ldots, \mathrm{init}]$. As E is stable by replacement of names for terms and $C[[n_1, \ldots, n_l]] =_{\mathsf{E}} n_x$, it follows that $C[\mathrm{init}, \ldots, \mathrm{init}] =_{\mathsf{E}} \mathrm{init}$. and therefore we can conclude $\mathrm{abs}(\mathsf{col}(t\sigma)) =_{\mathsf{E}_a} \mathrm{abs}(t\sigma)$.

□

We now define σ'_i and φ'_i (for $1 \leq i \leq n$). We assume w.l.o.g. that σ_i and φ_i are collapsed ($1 \leq i \leq n$) and let $\sigma'_i(x) = \mathrm{abs}(\sigma(x))$ and $\varphi'_i(w) = \mathrm{abs}(\varphi(w))$ (for all $1 \leq i \leq n$, $x \in \mathcal{D}om(\sigma_i)$, $w \in \mathcal{D}om(\varphi_i)$).

Definition A.6. We say that a collapsed term $t = C[[t_1, \ldots, t_k]]$ is *i-good* ($1 \leq i \leq n$) if there exist pure $\mathcal{F}_a$ recipes $r_1, \ldots, r_k$ such that $\varphi'_i \vdash^{r_j}_{\mathsf{E}_a} \mathrm{abs}(t_j)$ and if t_j is an i-good term ($1 \leq j \leq k$).

It is easy to see that a $(i-1)$-good term is also an i-good term. We now prove by induction on i that there exist $\S'_1, \ldots, \S'_i$ such that the recipes in $\S'_1, \ldots, \S'_i$ are pure $\mathcal{F}_a$-recipes and:

$$(P, \emptyset, \emptyset) \xrightarrow{\S'_1} (P_1, \varphi'_1, \sigma'_1) \xrightarrow{\S'_2} \ldots \xrightarrow{\S'_i} (P_i, \varphi'_i, \sigma'_i)$$

and that $\varphi_i(w)$ and $\sigma_i(x)$ are i-good (for all $w \in \mathcal{D}om(\varphi_i)$ and for all $x \in \mathcal{D}om(\sigma_i)$). We proceed with the proof distinguish among the possible actions at step i:

1. if the i-th action is νx, then $\S_i$ is the empty string. We choose $\S'_i$ also to be the empty string.

   By the operational semantics, we have that $\sigma_i = \sigma_{i-1} \cup \{x \mapsto n\}$ for some fresh name n. We only need to show that $\sigma_i(x)$ is an i-good term, since the other terms are i-good directly by the induction hypothesis. We consider the context $C = n$ (a context with 0 holes) and $k = 0$ in the definition of i-good and we can trivially conclude that $\sigma_i(x)$ is i-good.

   We also need to establish that the transition works, which is obviously the case, since $\sigma'_i(x) = \mathrm{abs}(\sigma_i(x)) = n$.

2. if the i-th action is an assignment $x := t$, then $\S_i$ is the empty string. We choose $\S'_i$ also as the empty string.

   By the operational semantics, we have that $\sigma_i = \sigma_{i-1} \cup \{x \mapsto \mathsf{col}(t\sigma_{i-1})\}$. As t is a term appearing in the protocol, it is a pure $\mathcal{F}_a$-term.

   Let C be the context obtained from t by replacing all variables with holes. Then $\sigma_i(x) = C[t_1, \ldots, t_k]$ for some i-good terms t_j $(1 \leq j \leq k)$ – the terms t_j are equal to $\sigma_{i-1}(y_j)$ for some $y_j \in \mathcal{D}om(\sigma_{i-1})$ $(1 \leq j \leq k)$. Let $t_j = C_j[[t^j_1, \ldots, t^j_{k_j}]]$.

   If C is the empty context we have that $\sigma_i(x) = \sigma_{i-1}(y)$ for some $y \in \mathcal{D}om(\sigma_{i-1})$ and therefore we can conclude by the induction hypothesis.

   Otherwise, if $C \neq \_$, let d be such that $\mathsf{root}(C) \in \mathcal{F}_d$. For $1 \leq j \leq k$, let $C'_j = C_j$ if $\mathsf{root}(C_j) \in \mathcal{F}_d$ and let $C_j = \_$ otherwise. Let $C' = C[C'_1, \ldots, C'_k]$. We have that $t\sigma_{i-1} = C'[[s_1, \ldots, s_l]]$ such that s_j is an i-good term (each s_j $(1 \leq j \leq l)$ is either some $t_{j'}$ $(1 \leq j' \leq k)$ which is i-good by the induction hypothesis or some $t^{j'}_z$ $(1 \leq j' \leq k,\ 1 \leq z \leq k_{j'})$ which is i-good because it is an alien term of $t_{j'}$, which is i-good by the induction hypothesis).

   Then $\mathsf{col}(t\sigma_{i-1})$ is either equal to some s_j $(1 \leq j \leq l)$ or to $C[s_1, \ldots, s_l]$. In the first case, we conclude because we have already seen that all s_j $(1 \leq j \leq l)$ are i-good. In the second case, we have that $\mathsf{root}(C) \in \mathcal{F}_a$ and therefore $\mathsf{root}(s_j) \in \mathcal{F}_b$ $(1 \leq j \leq l)$. Therefore $\mathrm{abs}(s_j) = \mathrm{init}$ and it is sufficient to choose $r_j = \mathrm{init}$ to obtain $\varphi'_i \vdash^{r_j}_{\mathsf{E}_a} \mathrm{abs}(s_j)$ $(1 \leq j \leq l)$. Furthermore we already know that all s_j are i-good and therefore we can conclude that $C[s_1, \ldots, s_l]$ is i-good.

   We also need to show that this transition in the conclusion works, i.e. that $x\sigma'_i =_{\mathsf{E}_a} t\sigma'_{i-1}$. We know that $\mathsf{col}(x\sigma_i) =_{\mathsf{E}} \mathsf{col}(t\sigma_{i-1})$ (by the hypothesis). We also have that $x\sigma'_i = \mathrm{abs}(x\sigma_i)$ and that $t\sigma'_{i-1} = \mathrm{abs}(t\sigma_{i-1})$. Using Claim A.2 and Claim A.1, we immediately conclude.

3. if the i-th action is a test $[s = t]$, then $\S_i$ is the empty string. We choose $\S'_i$ also as the empty string.

   As $\varphi_i = \varphi_{i-1}$ and $\sigma_i = \sigma_{i-1}$, it is sufficient to show that $s\sigma'_{i-1} =_{\mathsf{E}_a} t\sigma'_{i-1}$. But $s\sigma'_{i-1} = \mathrm{abs}(s\sigma_{i-1})$ and $t\sigma'_{i-1} = \mathrm{abs}(t\sigma_{i-1})$ by the definition of σ'_{i-1} and abs.

   By Claim A.2 we have that $\mathrm{abs}(\mathsf{col}(s\sigma_{i-1})) =_{\mathsf{E}_a} \mathrm{abs}(s\sigma_{i-1})$ and analogously for t.

   We conclude by Claim A.1 that $\mathsf{col}(t\sigma_{i-1}) =_{\mathsf{E}} \mathsf{col}(s\sigma_{i-1})$ (which is true by the hypothesis) implies $\mathrm{abs}(\mathsf{col}(t\sigma_{i-1})) =_{\mathsf{E}_a} \mathrm{abs}(\mathsf{col}(s\sigma_{i-1}))$.

4. if the i-th action is an output $\mathbf{send}(t)$, we have that $\varphi_i = \varphi_{i-1} \cup \{w_{|\mathcal{D}om(\varphi)|+1} \mapsto \mathsf{col}(t\sigma_{i-1})\}$ and that $\S_i$ is the empty string. We choose $\S'_i$ also as the empty string.

   We first have to establish that $\varphi_i(w_{|\mathcal{D}om(\varphi)|+1}) = \mathsf{col}(t\sigma_{i-1})$ is an i-good term, which is exactly the same as in the case of the assignment $x := t$ (see second item above).

   We also have to establish that this transition works in the conclusion, i.e. that $\varphi'_i(w_{|\mathcal{D}om(\varphi)|+1}) =_{\mathsf{E}_a} t\sigma'_{i-1}$ knowing that $\varphi_i(w_{|\mathcal{D}om(\varphi)|+1}) = \mathsf{col}(t\sigma_{i-1})$ (i.e. that the transition works in the hypothesis). We can conclude by Claim A.2 ($\varphi'_i(w_{|\mathcal{D}om(\varphi)|+1}) = \mathrm{abs}(\varphi_i(w_{|\mathcal{D}om(\varphi)|+1})) = \mathrm{abs}(\mathsf{col}(t\sigma_{i-1})) =_{\mathsf{E}_a} \mathrm{abs}(t\sigma_{i-1}) = t\sigma'_{i-1}$).

5. if the i-th action is an input **receive**(x), we have that $\sigma_i = \sigma_{i-1} \cup \{x \mapsto \mathsf{col}(r\varphi_{i-1})\}$ for some recipe r such that $\S_i = \mathbf{receive}(r)$.

   We prove by induction on r that $\mathsf{col}(r\varphi_{i-1})$ is an i-good term and at the same time we construct a pure $\mathcal{F}_a$-recipe r' such that $r'\varphi'_{i-1} =_{\mathsf{E}_a} \mathrm{abs}(r\varphi_{i-1})$. We choose $\S_i$ to be $\mathbf{receive}(r')$. This means in particular that the transition in the hypothesis will work ($x\sigma'_i =_{\mathsf{E}_a} r'\varphi'_{i-1}$).

   Therefore all we need is the proof of i-goodness and the construction of r' by induction.

   (a) base case: If r is a parameter $w \in \mathcal{W}$, we have that $r\varphi_{i-1}$ is i-good by the induction hypothesis (the outer induction). We choose $r' = w$ and we obtain $r'\varphi'_{i-1} = w\varphi'_{i-1} = \mathrm{abs}(w\varphi_{i-1}) = \mathrm{abs}(r\varphi_{i-1})$.

   (b) Let $r\varphi_{i-1} = C[[t_1, \ldots, t_k]]$. Let c be such that $\mathsf{root}(C) \in \mathcal{F}_c$.
   Each t_j $(1 \leq j \leq k)$ is such that $t_j = r_j\varphi_{i-1}$ for some recipe $r_j \subset r$ or an alien subterm of $\varphi_{i-1}(w)$ for some parameter $w \in \mathcal{D}om(\varphi_{i-1})$. In the first case we know that $t_j = \mathsf{col}(t_j)$ is an i-good term by the outer induction hypothesis (t_j is an alien subterm of $\varphi_{i-1}(w)$ for some $w \in \mathcal{D}om(\varphi_{i-1})$) and in the second case we that $\mathsf{col}(t_j)$ is an i-good term by the inner induction hypothesis. Similarly, there exist pure $\mathcal{F}_a$-recipes over φ'_{i-1} for $\mathrm{abs}(\mathsf{col}(t_j))$.
   Let $\mathsf{col}(t_j) = C_j[[t^j_1, \ldots, t^j_{k_j}]]$ and let $C'_j = C_j$ if $\mathsf{root}(C_j) \in \mathcal{F}_c$ and let $C'_j = \_$ otherwise $(1 \leq j \leq k)$. Let $C' = C[C_1, \ldots, C_k]$. As $t^j_{j'}$ are the alien subterms of an i-good term, it follows that $t^j_{j'}$ are i-good terms $(1 \leq j \leq k,\ 1 \leq j' \leq k_j)$ and there exist pure $\mathcal{F}_a$-recipes over φ'_{i-1} for $\mathrm{abs}(t^j_{j'})$.
   Then $r\varphi_{i-1} = C'[[s_1, \ldots, s_l]]$ where each s_j $(1 \leq j \leq l)$ is either some $t_{j'}$ $(1 \leq j' \leq k)$ or some $t^{j'}_z$ $(1 \leq j' \leq k,\ 1 \leq z \leq k_{j'})$. In either case s_j $(1 \leq j \leq l)$ are i-good terms and there exist pure $\mathcal{F}_a$-recipes over φ'_{i-1} for $\mathrm{abs}(s_j)$. As $\mathsf{col}(r\varphi_{i-1})$ is

      i. either some s_j, in which case we conclude directly that it is an i-good term and there is a pure $\mathcal{F}_a$-recipe over φ'_{i-1} for $\mathrm{abs}(s_j)$

      ii. or it is $C'[[s_1, \ldots, s_l]]$, in which it is also easy to establish that $C'[[s_1, \ldots, s_l]]$ is an i-good term (its alien subterms are i-good terms and there are pure $\mathcal{F}_a$-recipes over φ'_{i-1} for their abstractions).
      We also need to show that there is a pure $\mathcal{F}_a$-recipe over φ'_{i-1} for $C'[[s_1, \ldots, s_l]]$. In the case $\mathsf{root}(C') \in \mathcal{F}_b$, we simply choose the recipe $r' = \mathrm{init}$. Otherwise, if $\mathsf{root}(C') = \mathsf{root}(r) = f \in \mathcal{F}_a$ and $r = f(r_1, \ldots, r_m)$, we let $r' = f(r'_1, \ldots, r'_m)$. It is then easy to show by induction that $r'\varphi'_{i-1} =_{\mathsf{E}_a} \mathrm{abs}(r\varphi_{i-1})$.

□