BRICS RS-06-19 M. D. Pedersen: Logics for The Applied π Calculus

Basic Research in Computer Science

Logics for The Applied π Calculus

Michael David Pedersen

BRICS Report Series **RS-06-19**

ISSN 0909-0878 **December 2006**

Copyright © 2006, Michael David Pedersen.
BRICS, Department of Computer Science
University of Aarhus. All rights reserved.

Reproduction of all or part of this work is permitted for educational or research use on condition that this copyright notice is included in any copy.

See back inner page for a list of recent BRICS Report Series publications. Copies may be obtained by contacting:

BRICS
Department of Computer Science
University of Aarhus
IT-parken, Aabogade 34
DK–8200 Aarhus N
Denmark

Telephone: +45 8942 9300
Telefax: +45 8942 5601
Internet: BRICS@brics.dk

BRICS publications are in general accessible through the World Wide Web and anonymous FTP through these URLs:

http://www.brics.dk
ftp://ftp.brics.dk
This document in subdirectory RS/06/19/

Logics for The Applied π Calculus

◊

Master's thesis

Michael David Pedersen

June 12, 2006

(Revised August 7, 2006)

◊

Aalborg University
Department of Computer Science · Fredrik Bajers Vej 7E
DK-9210 Aalborg Ø

Preface

This BRICS report was originally a master's thesis documenting work in the specialisation year at the Department of Computer Science, Aalborg University, and is based on a project proposal by Hans Hüttel. The thesis strengthens and goes beyond the work presented in the DAT5 project entitled *A Logic of Frames in the Applied π Calculus.*

Familiarity with process calculi (e.g. CCS, the π calculus and associated modal logics) corresponding to the level given in the DAT4 course *Semantics and Verification* and the DAT6 course *Semantics and Verification 2* is assumed. Mathematics skills corresponding to bachelor level computer science, including an understanding of basic set theory, equivalence relations, definition and proof by induction, will be of advantage.

Michael David Pedersen,
December 18, 2006.

Summary

The Applied π calculus by Abadi and Fournet is a uniform and generic extension of the π calculus for which particular primitives such as tuples and cryptographic functions can be defined and used on a per-application basis. This is in contrast to the basic π calculus in which the only type of messages that can be communicated are atomic names. The key idea in Applied π is to integrate arbitrary terms into the calculus and its semantics based on a *function signature*, and to base equality between terms on an *equational theory* instead of syntactic equality on names.

Processes in Applied π also differ from standard π calculus processes by integrating the notion of *active substitutions* in the process expression itself. An active substitution is a binding of a term to a variable which intuitively represents a message sent by a process. The *frame* of a process can then be thought of as the substitution arising by combining all active substitutions embedded in the process expression, and is hence a representation of environment knowledge. Indistinguishability of two frames can be expressed in terms of a binary relation, $\approx_s$, called *static equivalence*. Roughly speaking, two frames are statically equivalent if an environment cannot distinguish the frames by testing for equality between any pair of terms constructed from variables in the frames.

Frames and static equivalence play a central role in labelled bisimilarity (which coincides with observational equivalence) of processes in Applied π: two processes are bisimilar if they can simulate each others actions *and* if their frames are statically equivalent, i.e. if the processes can be distinguished neither on their dynamic behaviour nor on their static characteristics.

The main objective of this report is a logic for the Applied π calculus which characterises labelled bisimilarity. The motivation is similar to that of Applied π itself, namely generality: the logic can be adapted to a particular application simply by defining a suitable function signature and equational theory. Since labelled bisimilarity contains a condition on static equivalence on frames, the first step towards a logic for Applied π is a logic for frames which characterises static equivalence. A logic for frames in turn relies on a definition of static equivalence which does not contain a universal quantification over arbitrary terms.

The first major contribution of this report is an strong version of static equivalence, $\approx_{ss}$, in which frames may be distinguished by testing on reduc-

tion of terms in addition to equality. Strong static equivalence is particularly useful because a refined definition, $\approx'_{ss}$, can be given which does not depend on universal quantification over arbitrary terms. The refined definition is based on the notion of *ecores* – an extension and generalisation of the cores used by Boreale et al. in the Spi calculus – which intuitively are the minimal terms deducible from a frame which are relevant for deciding static equivalence. We then show that $\approx_{ss}$ and $\approx'_{ss}$ coincide under certain general conditions (namely in independent convergent subterm theories). The refined definition is used as a basis for a logic of frames and to show that this logic characterises strong static equivalence. A further refinement, $\approx''_{ss}$, which is also shown to be equivalent to $\approx_{ss}$, is given and used as a basis for characteristic formulae in the logic for frames. Finally we show that strong static equivalence coincides with the standard static equivalence ($\approx_s$) under certain conditions. These conditions are for example satisfied by theories of symmetric key encryption and by theories of public key encryption if public keys are assumed to always be known.

The second major contribution of this report is a first order logic for frames, $\mathcal{LF}$, which characterises strong static equivalence and which is amenable to construction of characteristic formulae. $\mathcal{LF}$ includes atomic propositions which facilitate direct reasoning about the terms in a frame. The logic also includes first order quantification, intuitively over the set of terms which can be deduced from the frame by an environment, hence facilitating indirect reasoning about environment knowledge.

The third and final major contribution of this report is a modal logic, $\mathcal{LA}$, for Applied π which characterises labelled bisimilarity on processes. $\mathcal{LA}$ is obtained by adding suitable Hennessy-Milner style modalities to $\mathcal{LF}$. Consequently the logic can be used to reason both about the dynamic behaviour and static characteristics of processes. Finally, we demonstrate how $\mathcal{LA}$ can be applied to capture a well known attack on the Needham-Schroeder Public Key Protocol.

Resumé

Anvendt π-kalkylen af Abadi and Fournet er en ensartet og generel udvidelse af π-kalkylen, hvor primitiver såsom lister og kryptografiske funktioner kan defineres alt efter den ønskede anvendelse af kalkylen. Dette er i modsætning til π-kalkylen, hvor det kun er muligt at kommunikere atomare navne. Idéen i Anvendt π er at integrere vilkårlige termer i kalkylen og dennes semantik baseret på en *funktionssignatur*, og basere lighed mellem termer på en *ligningsteori* i stedet for syntaktisk lighed mellem navne.

Processer i Anvendt π afviger også fra almindelige π-kalkyleprocesser ved at integrere begrebet *aktive substitutioner* i selve syntaksen for processer. En aktiv substitution er en binding af en term til en variabel, som intuitivt set repræsenterer en besked sendt af processen. *Rammen* af en proces kan da betragtes som den substitution man opnår ved at samle alle aktive substitutioner fra processen, og er dermed en repræsentation af omgivelsernes viden. Observerbar lighed på to rammer kan udtrykkes i en binær relation kaldet *statisk ækvivalens*. Groft set er to rammer statisk ækvivalente, hvis de ikke kan skelnes af omgivelserne ved at teste lighed mellem vilkårlige par af termer konstrueret fra variabler i rammerne.

Rammer og statisk ækvivalens spiller en vigtig rolle i mærket bisimulering mellem processer i Anvendt π (som også sammenfalder med observerbar lighed mellem processer): To processer er bisimuleringsækvivalente, hvis de kan simulere hinandens handlinger, *og* deres rammer er statisk ækvivalente, dvs. hvis processerne hverken kan skelnes på deres dynamiske adfærd eller statiske egenskaber.

Det primære mål med denne rapport er en logic for Anvendt π som karakteriserer mærket bisimulering. Motivationen er, tilsvarende Anvendt π, generalitet: Logikken kan tilpasses bestemte anvendelsesområder ved simpelthen at definere en passende funktionssignatur og ligningsteori. Eftersom mærket bisimulering indeholder en betingelse om statisk ækvivalens af rammer, er det første skridt mod en logik for Anvendt π en logik for rammer, som karakteriserer statisk ækvivalens. En logic for rammer må endvidere bero på en definition af statisk ækvivalens som ikke indeholder universel kvantificering over vilkårlige termer.

Det første væsentlige bidrag i denne rapport er en stærk udgave af statisk ækvivalens, $\approx_{ss}$, hvor rammer kan skelnes på test a mulige termreduktioner såvel

som på lighed mellem termer. Stærk statisk ækvivalens er specielt brugbar, fordi en forfinet definition, $\approx'_{ss}$, kan udledes, som ikke beror på universel kvantificering over vilkårlige termer. Den forfinede definition er baseret på *ecores* – en udvidelse og generalisering af cores som anvendt af Boreale et al. på Spi-kalkylen – som intuitivt set er de minimale termer, der kan udledes fra en ramme, og som spiller en rolle i statisk ækvivalens. Vi viser at $\approx_{ss}$ og $\approx'_{ss}$ sammenfalder under bestemte generelle antagelser (nemlig i uafhængige konvergerende undertermsteorier). Den forfinede definition bruges som grundlag for en rammelogik og til at vise, at denne logic karakteriserer stærk statisk ækvivalens. En yderligere forfining, $\approx''_{ss}$, som vi også viser sammenfalder med $\approx_{ss}$, bliver brugt som grundlag for at udlede karakteristiske formler i rammelogikken. Endelig viser vi at stærk statisk ækvivalens sammenfalder med standardudgaven ($\approx_s$) under bestemte betingelser. Disse betingelser er for eksempel opfyldt af teorier med symmetrisk kryptering, og af teorier med offentlig-nøglekryptering hvor offentlige nøgler altid er kendt af alle parter.

Det andet væsentlige bidrag i denne rapport er en førsteordens rammelogic, $\mathcal{LF}$, som karakteriseser stærk statisk ækvivalens, og som giver anledning til konstruktion af karakteristiske formler. $\mathcal{LF}$ indeholder atomare udsagn, der muliggør direkte ræsonnementer om termerne i en ramme. Logikken indeholder også førsteordens kvantificering, intuitivt set over mængden af termer, som omgivelserne kan udlede fra en ramme, hvilket muliggør indirekte ræsonnementer om omgivelsernes viden.

Det tredje og sidste væsentlige bidrag i denne rapport er en modallogik, $\mathcal{LA}$, for Anvendt π som karakteriserer mærket bisimulering mellem processer. $\mathcal{LA}$ er en udvidelse af $\mathcal{LF}$ med passende Hennessy-Milner-modaliteter. Dermed opnås en logic, som kan ræsonnere om den dynamiske adfærd, såvel som de statiske egenskaber, af processer. Endelig demonstrerer vi, hvordan $\mathcal{LA}$ kan anvendes til at udtrykke et velkendt angreb på Needham-Schroeder Public Key protokollen.

Contents

1 Introduction **1**
- 1.1 Brief History . . . 1
- 1.2 The Applied π Calculus . . . 2
- 1.3 Static Equivalence . . . 3
- 1.4 Process Logics . . . 4
- 1.5 Objectives and Contributions . . . 4
- 1.6 Outline of the Report . . . 6

2 Preliminaries **7**
- 2.1 Syntax of Applied π . . . 7
- 2.2 Semantics of Applied π . . . 10
- 2.3 Equivalences in Applied π . . . 18
- 2.4 Summary . . . 23

3 A Refined Definition of Strong Static Equivalence **25**
- 3.1 Previous Work . . . 25
- 3.2 Initial Definitions . . . 28
- 3.3 The Refined Definition . . . 37
- 3.4 Examples . . . 39
- 3.5 Summary . . . 42

4 Results On The Refined Definition **45**
- 4.1 $\approx'_{ss}$ implies $\approx_{ss}$ . . . 45
- 4.2 $\approx_{ss}$ implies $\approx'_{ss}$ . . . 51
- 4.3 Conditions under which $\approx_s$ and $\approx_{ss}$ Coincide . . . 56
- 4.4 Summary . . . 62

5 A Logic for Frames **63**
- 5.1 Motivation . . . 63
- 5.2 Syntax and Semantics . . . 67
- 5.3 Characterisation . . . 69
- 5.4 Characteristic Formulae . . . 73
- 5.5 Summary . . . 82

6 A Logic for Applied π **83**
6.1 Syntax 83
6.2 Semantics 84
6.3 Match and Matching Input Modality 85
6.4 Characterisation 88
6.5 Summary 91

7 An Application to the Needham-Schroeder Public Key Protocol **93**
7.1 The Needham-Schroeder Public Key Protocol 93
7.2 The Specification 95
7.3 Logical Properties 100
7.4 Summary 104

8 Conclusion **105**
8.1 Contributions 105
8.2 Future Work 106

Chapter 1

Introduction

As computer and software systems play increasingly important roles in society, they are also trusted with more and more safety-critical tasks. Classical examples include monitoring of nuclear reactors and automated metro trains which operate without drivers. Clearly the correct operation of such software systems is of utmost importance. Correctness and security is also a concern in the financial world where transactions are performed online, and in everyday email-communication where sensitive data may be transmitted over the Internet. Hence there is an increasing interest in formal methods for modelling software systems and verifying that they work as intended.

Recent years have seen the advent of ubiquitous computing: the integration of small and unobtrusive computing devices into our environment and every day life. For example modern cars have embedded computers which monitor and report on various functions, wrist clocks may contain computers with calendars notifying of scheduled events, and tiny computers with environmental sensors ("motes") can be deployed to form wireless networks which monitor e.g. seismic activity. This raises enormous scientific challenges in ensuring that complex ubiquitous computer networks function correctly and safely. Indeed, "Science for Global Ubiquitous Computing" is the title of one of the seven Grand Challenges for Computer Research [16] proposed for the upcoming decades.

1.1 Brief History

Process calculi form the basis of a large range of formal methods, and was founded by Robin Milner in his 1980 publication on the Calculus of Communicating Systems (CCS) [21]. CCS enables software systems and communication protocols to be modelled abstractly as processes which can synchronise on channel names and which can then be reasoned about in the calculus. This later developed into the π calculus [22] with the novelty of allowing private channel names (i.e. with restricted scope) and allowing communication of channel names themselves (based on the notion of scope extension), resulting in a vast increase of expressive power. It then became possible to model concepts such as access control and mobility.

The π calculus has sparked a large body of further research resulting in a number of extensions to the calculus, e.g. the Polyadic π calculus [25, Section 3.3] which allows communication of tuples of names instead of just singleton names. Another such extension is Abadi and Gordon's Spi calculus [5] which adds cryptographic primitives, thus enabling reasoning about cryptographic protocols.

1.2 The Applied π Calculus

In a 2001 publication [4], Abadi and Fournet presented the Applied π calculus, a uniform and generic extension of the π calculus for which particular primitives such as tuples and cryptographic functions can be defined and used on a per-application basis. This is in contrast to the basic π calculus in which the only type of messages that can be communicated are atomic names. The key idea in Applied π is to integrate arbitrary terms into the calculus and its semantics based on a *function signature*, and to base equality between terms on an *equational theory*.

The function signature specifies the set of function symbols which can be used to iteratively form terms over names, and the equational theory, $=_\mathcal{E}$, is an equivalence relation which defines when two terms are equal. As an example, the signature may contain two binary function symbols *enc* and *dec* which can be used for symmetric encryption and decryption, respectively. This signature gives rise to terms on the form $enc(b, k^+)$, which intuitively represents the name b encrypted with the public key k^+. The associated equational theory should define the relationship between encryption and decryption, in which case the equality $dec(enc(b, k^+), k^-) =_\mathcal{E} b$ would hold (where k^- is the corresponding private key of k^+).

Processes in Applied π also differ from standard π calculus processes by integrating the notion of *active substitutions*, which intuitively capture the history of messages sent by a process in the process expression itself. Take as an example the following Applied π process:

$$P = (\nu k)\overline{a}\langle enc(b, k^+)\rangle.\overline{a}\langle k^-\rangle.P'$$

Here the name k is fresh, indicating that it is a the seed for a public-private key pair unknown to the environment. P performs two outputs on channel a: first the name b encrypted with public key k^+, and then the corresponding private key k^-. The semantics of Applied π prescribes that P reduces to the process

$$(\nu k)(\{^{enc(b,k^+)}/_{x_1}\}|\{^{k^-}/_{x_2}\}|P')$$

The sub-processes $\{^{enc(b,k^+)}/_{x_1}\}$ and $\{^{k^-}/_{x_2}\}$ are active substitutions which represent the messages sent by P, and hence the information made available to the environment. Informally, the *frame* $\varphi(P)$ of the process P can then be thought of as the substitution arising by combining all active substitutions embedded in the process while preserving the restriction on k:

$$\varphi(P) = (\nu k)\{^{enc(b,k^+)}/_{x_1},\ ^{k^-}/_{x_2}\}$$

In this way $\varphi(P)$ is a representation of the terms known by the environment as a result of intercepting P's output activities on unsecured communication channels.

Applied π can be used to model protocols by defining a signature and equational theory appropriate to a given protocol. An example of existing work to this end is [1] in which Abadi et al. reason about guessing attacks. Furthermore, automated analysis of models can be carried using Blanchet's tool ProVerif [8]. ProVerif has been employed by Kremer and Ryan in [19] to analyse security protocol models in Applied π for known-pair and chosen-text attacks (relying on weaknesses in block chaining modes), and in [18] to analyse an electronic voting protocol.

1.3 Static Equivalence

Indistinguishability of two frames can be expressed in terms of a binary relation called *static equivalence*. Intuitively two frames are statically equivalent if an environment cannot distinguish the frames by testing for equality between any pair of terms which are based on the frames. Take as an example the following frames:

$$\begin{aligned}
\varphi_1 &= (\nu k, b)\{^{enc(b,k^+)}/_{x_1},\ ^{k^-}/_{x_2},\ ^{b}/_{x_3}\}\\
\varphi_2 &= (\nu k, b)\{^{enc(enc(b,b^+),k^+)}/_{x_1},\ ^{k^-}/_{x_2},\ ^{enc(b,b^+)}/_{x_3}\}\\
\varphi_3 &= (\nu k, b)\{^{enc(enc(b,k^+),k^+)}/_{x_1},\ ^{k^-}/_{x_2},\ ^{b}/_{x_3}\}
\end{aligned}$$

Then φ_1 and φ_2 are statically equivalent; the intuition is that the equality $dec(x_1, x_2) =_{\mathcal{E}} x_3$ holds in both frames – because $dec(enc(b, k^+), k^-) =_{\mathcal{E}} b$ and $dec(enc(enc(b, b^+), k^+), k^-) =_{\mathcal{E}} enc(b, b^+)$ – and *any* other equality over terms built from variables and non-bound names which holds in one frame also holds in the other. In particular, the private key b^- is now known in φ_2 and therefore the environment cannot decrypt x_1 twice and use the result to distinguish the two frames. On the other hand, the frames φ_1 and φ_3 are *not* statically equivalent, because the equality $dec(dec(x_1, x_2), x_2) =_{\mathcal{E}} x_3$ holds in φ_3 but not in φ_1 (nor in φ_2).

The claim that φ_1 and φ_2 are statically equivalent is a bold one, because we may have overlooked some equality which holds in one frame but not in the other. In fact one may question whether static equivalence is even decidable. The answer is positive for certain equational theories and negative for others; these issues has been investigated by Abadi and Cortier in [2, 3] and by Borgström in [10]. The notions of frames and static equivalence also feature in the work by Boreale et al.[9] in which proof techniques for the Spi calculus are developed, although frames are not an explicit part of the process syntax in the Spi calculus. We shall return to this work in Chapter 3.

Static equivalence is interesting in its own right as a formal definition of indistinguishability of environment knowledge, but it also a useful notion for defining bisimilarity. Roughly speaking, two processes are bisimilar if they can simulate each other's actions *and* if their frames are statically equivalent; i.e. if the processes can be distinguished neither on their dynamic behaviour nor on their static characteristics.

1.4 Process Logics

Verification of a software system using a process calculus typically involves three steps. First, an abstract model of the software system is expressed as a *model process* P_m which captures the essential behaviour of the system. Second, the idealised behaviour, which the system designer expects the system to exhibit, is expressed as the *specification process* P_s. Third, it is shown that P_m and P_s are equivalent according to a suitable notion of bisimulation equivalence.

An alternative, or complementary, way of verifying software systems is to model the system in a process calculi and then express correctness properties as formulae of a modal logic for the relevant process calculi, e.g. the Hennessy-Milner Logic (HML) for CCS [15] and for the π calculus [23], or the Spi calculus logic in [13]. Here the modalities are used to reason about *possible* or *necessary* actions. Another classical example is the BAN logic [11] which has been widely used for verification of authentication protocols, and here the modality is on *belief* (although this logic is not based explicitly on a process calculus). Generally, a process logic has the advantage that correctness properties may be expressed more directly and clearly as a formula than as a specification process, and a logical approach provides a different perspective on the verification problem.

A common result for process logics is that they characterise a suitable notion of bisimulation equivalence between processes, i.e. that that two processes are equivalent exactly if they satisfy the same set of formulae. Hence if a specification process satisfies some correctness property and the model process does not, then the model and specification are not bisimilar.

1.5 Objectives and Contributions

The main objective of the work documented in this report is a logic for the Applied π calculus. This logic would have the same appeal as the Applied π calculus itself has in the first place, namely generality, in that a logic for Applied π may capture other logics (such as the Spi logics) as a special case. We design the logic for Applied π in three stages, each of which provide a significant contribution to the theoretical development of Applied π.

The first contribution of this report is a stronger definition of static equivalence (henceforth referred to as *strong static equivalence*) in which frames may be distinguished by testing on reduction of terms in addition to equality. We argue that strong static equivalence is meaningful in applications; for example, in contrast to the standard definition, strong static equivalence distinguishes the following two frames:

$$\begin{aligned} \varphi &\triangleq (\nu a, c, k)\{{}^{enc(a,k^+)}/_{x_1}, {}^{k^-}/_{x_2}\} \\ \varphi' &\triangleq (\nu a, c, k)\{{}^{enc(a,k^+)}/_{x_1}, {}^{c^-}/_{x_2}\} \end{aligned}$$

Here we have that $dec(x_1, x_2)$ is reducible in the first frame but not in the second. There is no distinguishing equality, so the frames are statically equivalent according to the standard definition. However we also show that there are theories in which strong static equivalence coincides with the standard static equivalence; these include theories with symmetric key encryption and theories with public key encryption in which public keys are always known.

In addition to being meaningful in applications, strong static equivalence is useful because a refined definition can be given (inspired by the notion of cores in [9]) which does not rely on universal quantification over arbitrary terms and which places explicit conditions on certain syntactic equalities and reductions.

This leads us to the second contribution of this report, namely a first order logic for frames in Applied π which characterises strong static equivalence. The logic is designed based on the refined definition of strong static equivalence, which also gives a direct approach to construction of characteristic formulae for frames (a characteristic formula for frame φ is a finite formula which is satisfied by exactly those frames that are strong statically equivalent to φ). Characteristic formulae can be useful in theoretical developments but they also demonstrate that the logic is highly expressive, since a single formulae can capture all the essential properties (with respect to strong static equivalence) of a frame. Indeed the strong version of static equivalence is devised exactly with characteristic formulae in mind.

The logic of frames includes atomic propositions for asserting equality, reduction and syntactic equality between terms, and also includes first order quantification, intuitively over the set of terms which can be deduced from the frame by an environment. Hence a formula may be given to assert that e.g.
there exists a term y (known by the environment) such that x_1 can be decrypted with y to obtain x_2

The third and final contribution of this report is a modal logic for Applied π which characterises a bisimulation equivalence on processes. This logic is obtained by adding suitable Hennessy-Milner style modalities to the logic of frames, allowing assertions such as
process P can perform an input of an encrypted term M_1 followed by some output, after which the decryption key for the term M_1 is known by the environment

1.6 Outline of the Report

The report is structured as follows. In Chapter 2 we introduce the Applied π calculus in greater detail along with other necessary preliminaries, including a high-level definition of strong static equivalence. Chapter 3 gives a refined definition of strong static equivalence which is a necessary basis for characteristic formulae, and Chapter 4 gives a proof that the refined definition coincides with the standard definition under certain conditions. Chapter 5 presents a first-oder logic for frames after the discussing underlying considerations, and shows that the logic characterises static equivalence under certain conditions. The chapter ends with a construction of characteristic formulae. In Chapter 6 the logic of frames is extended to a logic for Applied π and it is shown that the logic characterises bisimilarity. An application of the logic to the analysis of the Needham-Schroeder Public Key Protocol is then given in Chapter 7. The report is concluded in Chapter 8 along with pointers to future work.

Chapter 2

Preliminaries

In this chapter we introduce the Applied π calculus based on [4]. Section 1 presents the syntax with emphasis on how arbitrary terms are represented in the calculus. Section 2 presents the semantics with emphasis on equality between terms and the underlying notion of term rewrite systems. Section 3 presents some important equivalence relations, including two notions of static equivalence which will feature extensively throughout the report.

2.1 Syntax of Applied π

One of the novelties of Applied π over the Pi calculus is that arbitrary terms can be passed over channels and tested for equality. We thus start by defining terms, and then proceed to define the syntax of two kinds of processes which feature in the calculus.

2.1.1 Terms

We define terms following the treatment in [7, Section 3.1]. Terms are constructed by applying function symbols to names, and the available function symbols may vary on a per application basis. The *signature* Σ is a set of available functions symbols where each $f \in \Sigma$ is associated with a natural number k, which is the *arity* of f. Functions of arity 0 are called constant symbols. We denote the set of function symbols in Σ with arity k by $\Sigma^{(k)}$.

Let $\mathcal{N}$ be the set of names, ranged over by $a, b, c, \ldots, k$, let $\mathcal{X}$ be the set of variables, ranged over by x, y, z and let $\mathcal{U}$ be the set $\mathcal{N} \cup \mathcal{X}$, ranged over by u. Then the set of terms in the signature, $T(\Sigma,\mathcal{U})$, is defined inductively as follows

- $\mathcal{U} \subseteq T(\Sigma,\mathcal{U})$
- for all $k \geq 0$:
  $f(M_1, \ldots, M_k) \in T(\Sigma,\mathcal{U})$ if $f \in \Sigma^{(k)}$ and $M_1, \ldots, M_k \in T(\Sigma,\mathcal{U})$

The set of names, respectively variables, occurring in a term $M \in T(\Sigma,\mathcal{U})$ will be denoted by $n(M)$, respectively $v(M)$, and we will use the abbreviation $n(M_1, M_2)$ to denote $n(M_1) \cup n(M_2)$.

Often we will be interested in identifying substrings at certain positions of a term $M \in T(\Sigma,\mathcal{U})$. The set of positions in M, $pos(M)$, is a set of strings over the alphabet of positive integers and is defined inductively as

- If $M \in \mathcal{U}$, then $pos(M) \triangleq \{\epsilon\}$ where ϵ denotes the empty string.
- If $M = f(M_1, \ldots, M_k)$ for some $f \in \Sigma^{(k)}$ then

$$pos(M) \triangleq \{\epsilon\} \cup \bigcup_{i=1}^{k} \{iw \mid w \in pos(M_i)\}$$

We say that a position w' is a *prefix* of a position w if there exists w'' such that $w = w'w''$. The subterm of M at position w is denoted by $M|_w$ and is defined inductively on the length of w as follows:

$$\begin{aligned} M|_\epsilon &\triangleq M \\ f(M_1, \ldots, M_k)|_{iw} &\triangleq M_i|_w \end{aligned}$$

Finally, we define the set of subterms of M as

$$st(M) \triangleq \{M|_w \mid w \in pos(M)\}$$

2.1.2 Plain Processes

We now proceed to the syntax of processes in the Applied π calculus. A process can either be a plain processes or an extended processes. Plain processes are similar to processes in the Pi calculus except that arbitrary terms may be sent along channels:

$$A, B, C ::= \mathbf{0} \mid A|B \mid !A \mid (\nu n)A \mid \textit{if } M_1 = M_2 \textit{ then } A \textit{ else } B \mid u(x).A \mid \overline{u}\langle M\rangle.A$$

The set of plain processes is denoted by $\mathcal{A}$. The informal meaning of each process construct is explained briefly below.

1. $\mathbf{0}$ is the null process which cannot do anything.
2. $A|B$ is parallel composition in which A and B execute concurrently.
3. $!A$ is replication, the intuition being that an unlimited number of copies of A are executed in parallel.
4. $(\nu n)A$ is name restriction, meaning that the name n is private, or "fresh", in A, and thus can only be used in A.

5. *if* $M_1 = M_2$ *then* A *else* B is a process which behaves as A if $M_1 = M_2$ and as B otherwise. The equality is more general than syntactic equality, which will be clear when the semantics is presented in the next section.

6. $u(x).A$ is message input. A term M can be input on the channel u and bound to the variable x, whereafter the process behaves as A with x substituted for M.

7. $\overline{u}\langle M\rangle.A$ is message output. A term M is output on the channel u, where after the process continues as A.

2.1.3 Extended Processes

Extended processes are plain processes extended with the notion of *active substitutions*, which play an essential role in Applied π. In general, a substitution is a total function $\sigma : \mathcal{X} \to T(\Sigma,\mathcal{U})$ which maps variables to terms [7, Section 3.1]. Following standard conventions, we shall use $x\sigma$ to denote the substitution σ applied to x and we let $dom(\sigma)$ and $im(\sigma)$ denote the domain and the image of σ, respectively. We define the extended substitution $\sigma\{{}^{M}/_{y}\}$ as

$$x\sigma\{{}^{M}/_{y}\} = \begin{cases} M & \text{if } y = x \\ x\sigma & \text{if } y \neq x \end{cases}$$

The extension $\sigma\sigma_2$ is obtained by iteratively extending σ with each element of σ_2 in the obvious way.

We write $\emptyset$ for the substitution σ with $dom(\sigma) = \emptyset$ and we shall generally use the notation $\{{}^{M_1}/_{x_1}, \ldots, {}^{M_k}/_{x_k}\}$ to define the substitution σ where $\sigma(x_i) = M_i$ (for $1 \leq i \leq k$) and $\sigma(y) = y$ for all $y \in \mathcal{X} - \{x_1, \ldots, x_k\}$. When the number of variables in the substitution is insignificant we shall use the notation $\{{}^{M_i}/_{x_i}\}_{i\in I}$ where I denotes some sequence, and in general we shall use the notation $(x)_{i\in I}$ to denote the sequence $(x_1, \ldots, x_k)$ for some k. A substitution can be extended to a mapping $\hat{\sigma} : T(\Sigma,\mathcal{U}) \to T(\Sigma,\mathcal{U})$ where $\hat{\sigma}(M)$ is the term resulting from replacing all $x \in v(M)$ with $x\sigma$. A similar extension can be made to processes, giving a substitution $\hat{\hat{\sigma}} : \mathcal{P} \to \mathcal{P}$ where $\hat{\hat{\sigma}}(P)$ is the process resulting by replacing each term M in P for $\hat{\sigma}(M)$. In the following we shall use substitutions and their extensions interchangeably. We shall also assume that any substitution σ is cycle-free in the sense that none of the variables in the domain of σ occur in terms in the image of σ.

A grammar for extended processes can now be given:

$$P, Q, R ::= A \mid P|Q \mid (\nu n)P \mid (\nu v)P \mid \{{}^{M}/_{x}\}$$

The set of extended processes is denoted by $\mathcal{P}$; note that $\mathcal{A} \subset \mathcal{P}$. Considered as a process, the *active substitution* $\{{}^{M}/_{x}\}$ may "attack" any other process it is put in parallel with and apply the substitution to this process. Hence the parallel composition $\{{}^{M}/_{x}\}|P$ is intuitively equivalent to the process $P\{{}^{M}/_{x}\}$.

Processes can be protected against substitution by using variable restriction, so e.g. the substitution $\{{}^{M}/_{x}\}$ will have no effect on the process $(\nu x)P$.

Note that active substitutions in general cannot be under input/output guards, cannot be part of a conditional, and cannot be replicated. Several active substitutions can be put in parallel, in which case we write $\{{}^{M_1}/_{x_1}\}|\ldots|\{{}^{M_k}/_{x_k}\} = \{{}^{M_1}/_{x_1}, \ldots, {}^{M_k}/_{x_k}\}$. When multiple names are under restriction, we shall write $(\nu n_1, \ldots, n_k)P$ instead of $(\nu n_1), \ldots, (\nu n_1)P$, and we shall write $(\nu \tilde{n})P$ when the particular names under restriction are unimportant.

The set of free names, $fn(P)$, of an extended process is defined as follows:

$$\begin{aligned}
fn(\mathbf{0}) &= \emptyset \\
fn(P|Q) &= fn(P) \cup fn(Q) \\
fn(!P) &= fn(P) \\
fn((\nu n)P) &= fn(P) - \{n\} \\
fn(\textit{if } M_1 = M_2 \textit{ then } P \textit{ else } Q) &= n(M_1) \cup n(M_2) \cup fn(P) \cup fn(Q) \\
fn(a(x).P) &= \{a\} \cup fn(P) \\
fn(\overline{a}\langle M\rangle.P) &= \{a\} \cup n(M) \cup fn(P) \\
fn(\{{}^{M}/_{x}\}) &= n(M)
\end{aligned}$$

The set of bound names, $bn(P)$, of an extended process is defined as follows:

$$\begin{aligned}
bn(\mathbf{0}) &= \emptyset \\
bn(P|Q) &= bn(P) \cup bn(Q) \\
bn(!P) &= bn(P) \\
bn((\nu n)P) &= bn(P) \cup \{n\} \\
bn(\textit{if } M_1 = M_2 \textit{ then } P \textit{ else } Q) &= bn(P) \cup bn(Q) \\
bn(a(x).P) &= bn(P) \\
bn(\overline{a}\langle M\rangle.P) &= bn(P) \\
bn(\{{}^{M}/_{x}\}) &= \emptyset
\end{aligned}$$

The free variables $fv(P)$ and the bound variables $bv(P)$ are defined similarly. The set of *names* of an extended process is defined as $n(P) = fn(P) \cup bn(P)$.

In this report we shall mainly be interested in extended processes, so henceforth we refer to these simply as *processes*.

2.2 Semantics of Applied π

Having defined the syntax in the previous section, we now proceed by giving meaning to the syntactical constructs in Applied π. We start by considering how equality between terms can be defined on a per application basis, which relies on the notion of an equational theory. We then define structural equivalence, which is the foundation of the reduction relation and the labelled operational semantics which are subsequently introduced.

2.2.1 Rewrite Systems and The Equational Theory

Test on equality between terms is a central feature of the conditional process construct. In order to define the semantics for conditionals, a definition of equality must therefore be given. This in turn relies on the notion of *term rewrite systems* (TRS) to which we now give a brief introduction, following the treatment in [7].

Term rewrite systems are used to specify how terms can be evaluated step by step. First, a *rewrite rule* r is on the form $L >_r R$ where $L, R \in T(\Sigma, \mathcal{X})$ and $v(R) \subseteq v(L)$. L will be referred to as the *left hand side* (LHS) or the *redex*, and R as the *right hand side* (RHS) or the *reduct*. The function symbol occurring as the outermost function symbol of the redex will be referred to as the *destructor function*.

A term rewrite system $\mathcal{R}$ is then a set of rewrite rules. We say that a term M_1 is *less general than*, or *an instance of*, a term M_2 if there exists a unifying substitution θ such that $M_1 = M_2\theta$. A term M_1 *reduces primitively* to M_2 using some rule $L >_r R$, written $M_1 >_r M_2$, if $M_1 = L\theta$ and $M_2 = R\theta$ for some substitution θ. More generally though, we are interested in the rewrite relation $>$ induced by the rewrite system. We then define $M_1 > M_2$ if and only if some subterm $M_1|_w$ is an instance of L with some unifying substitution θ and $M_2 = M_1\{^{R\theta}/_{M_1|_w}\}$, and we say that M_1 rewrites (or reduces) to M_2.

Example 2.2.1. Consider a rewrite system $\mathcal{R}$ over the signature Σ consisting of a binary function symbol for pair construction, $[\cdot, \cdot]$, and the unary function symbols $fst(\cdot)$ and $snd(\cdot)$ for projection of first and second coordinates of a pair, respectively. $\mathcal{R}$ can be defined from the following two rewrite rules:

$$fst([z_1, z_2]) >_{r_1} z_1$$
$$snd([z_1, z_2]) >_{r_2} z_2$$

We then have that e.g. $fst([snd([a, b]), c]) >_{r_1} snd([a, b])$ as an example of a primitive rewrite. We also have that $fst([snd([a, b]), c]) > fst([b, c]) > b$ and $fst([snd([a, b]), c]) > snd([a, b]) > b$ □

Here are some important notions concerning rewrite systems:

- We denote by $>^*$ the reflexive and transitive closure of the relation $>$ and by $<>^*$ the symmetric, reflexive and transitive closure of $>$.
- A term M_1 is *reducible* if there exists a term M_2 s.t. $M_1 > M_2$. M_1 is in *normal form* (or irreducible) if it is not reducible.
- A term M_2 is *a normal form* of M_1 if $M_1 >^* M_2$ and M_2 is in normal form. If M_1 has a unique normal form, this is denoted by $M_1\!\downarrow$.

- A TRS is called *confluent* if whenever $M_1 >^* M_2$ and $M_1 >^* M_3$ there exists an M_4 such $M_2 >^* M_4$ and $M_3 >^* M_4$.
- A TRS is called *terminating* if there is no infinite chain $M_1 > M_2 > M_3 \ldots$.
- A TRS is called *normalising* if every term has a normal form.
- A TRS is called *convergent* if it is both confluent and terminating.

Note that *terminating* implies *normalising*, although the converse does not necessarity hold.

Having introduced the basics of term rewrite systems we can now give a definition of equality between terms.

Definition 1 (Equational Theory). *An equational theory $=_\mathcal{E}$, generated from a TRS $\mathcal{R}$, is an equivalence relation on $T(\Sigma,\mathcal{U})$ which satisfies the following rules:*

$$\begin{array}{ll}
(\textsc{Rewrite}) & \dfrac{}{M_1 =_\mathcal{E} M_2}\ \textit{if } (M_1 >_r M_2) \in \mathcal{R} \\[2ex]
(\textsc{Reflexivity}) & \dfrac{}{M =_\mathcal{E} M} \\[2ex]
(\textsc{Symmetry}) & \dfrac{M_2 =_\mathcal{E} M_1}{M_1 =_\mathcal{E} M_2} \\[2ex]
(\textsc{Transitivity}) & \dfrac{M_1 =_\mathcal{E} M_3 \quad M_3 =_\mathcal{E} M_2}{M_1 =_\mathcal{E} M_2} \\[2ex]
(\textsc{Function}) & \dfrac{M_1 =_\mathcal{E} M_1', \ldots, M_k =_\mathcal{E} M_k'}{f(M_1, \ldots, M_k) =_\mathcal{E} f(M_1', \ldots, M_k')}\ \textit{where } f \in \Sigma^{(k)} \\[2ex]
(\textsc{Substitution}) & \dfrac{M_1 =_\mathcal{E} M_2}{M_1\theta =_\mathcal{E} M_2\theta}\ \textit{for all substitutions } \theta
\end{array}$$

It is shown in [7, Theorem 3.1.12] that the equational theory $=_\mathcal{E}$ generated from $\mathcal{R}$ coincides with the reflexive, symmetric and transitive closure of the rewrite relation, i.e. $M_1 =_\mathcal{E} M_2 \Leftrightarrow M_1 <>^* M_2$. This gives us a degree of freedom in later proofs, for as we shall discover, the definition in terms of an equational theory is not always convenient to work with. Another important result, which we shall use extensively in this report, says that $M_1 =_\mathcal{E} M_2 \Leftrightarrow M_1\!\downarrow = M_2\!\downarrow$ for confluent and normalising rewrite systems [7, Theorem 2.1.9]

Example 2.2.2. As an example of an equational theory, we now define the theory $\mathcal{E}_{sym}$ of symmetric key encryption, pairing, and hash functions. When doing so, we must define a signature by specifying function symbols and their associated arities, and we must define the rewrite system used for generating the equational theory.

Theory $\mathcal{E}_{sym}$ **.**
Signature: $enc(\cdot,\cdot)$, $dec(\cdot,\cdot)$, $[\cdot,\cdot]$, $fst(\cdot)$, $snd(\cdot)$,$f(\cdot)$.
Rewrite System:

$$\begin{aligned} dec(enc(z_1, z_2), z_2) &>_{r_1} z_1 \\ fst([z_1, z_2]) &>_{r_2} z_1 \\ snd([z_1, z_2]) &>_{r_3} z_2 \end{aligned}$$

The hash property of the function f, i.e. that f is one-way and collision-free, is reflected by the lack of axioms involving f. Example terms in this theory are $enc(a, k)$, $dec(enc(a, k), k)$ and $enc([f(a), [b, c]], k)$. Here are some example equalities in $\mathcal{E}_{sym}$:

$$\begin{aligned} dec(enc(a, k), k) &=_{\mathcal{E}} a \\ dec(enc([f(a), [b, c]], k), k) &=_{\mathcal{E}} [f(a), [b, c]] \\ fst(dec(enc([f(a), [b, c]], k), k)) &=_{\mathcal{E}} f(a) \end{aligned}$$

□

Example 2.2.3. In public key encryption schemes a public key k^+, typically known to any participating principal, can be used to encrypt messages, but only a corresponding private key k^-, known only by a single principal, can be used for decryption. The idea behind public key encryption can be expressed in the following equational theory, where we again have pairing and hash functions:

Theory $\mathcal{E}_{pub}$ **.**
Signature: $enc(\cdot,\cdot)$, $dec(\cdot,\cdot)$, $\cdot^+$, $\cdot^-$, $[\cdot,\cdot]$, $fst(\cdot)$, $snd(\cdot)$,$f(\cdot)$.
Rewrite System:

$$\begin{aligned} dec(enc(z_1, {z_2}^+), z_2^-) &>_{r_1} z_1 \\ fst([z_1, z_2]) &>_{r_2} z_1 \\ snd([z_1, z_2]) &>_{r_3} z_2 \end{aligned}$$

We have the following equalities:

$$\begin{aligned} &dec(enc(a, k^+), k^-) =_{\mathcal{E}} a \text{ but} \\ &dec(enc(a, k^+), k^+) \neq_{\mathcal{E}} a \end{aligned}$$

□

Example 2.2.4. The theories introduced in the two previous examples are fairly simple. When demonstrating concepts and results later in the report we need to take more complex rewrite rules into account. Hence we now define an extended version of the public-key theory in which two keys are used for both encryption and decryption.

Theory $\mathcal{E}_{pub2}$ **.**
Signature: $enc(\cdot,\cdot)$, $dec(\cdot,\cdot)$, $\cdot^+$, $\cdot^-$, $[\cdot,\cdot]$, $fst(\cdot)$, $snd(\cdot)$,$f(\cdot)$.
Rewrite System:

$$\begin{aligned} dec(enc(z_1, f(z_2{}^+, z_3{}^+)), f(z_2^-, z_3^-)) &>_{r_1} z_1 \\ fst([z_1, z_2]) &>_{r_2} z_1 \\ snd([z_1, z_2]) &>_{r_3} z_2 \end{aligned}$$ □

In this report we shall restrict our attention to a certain class of equational theories known as *convergent subterm theories* (the following definition is adapted from [2, Definition 1]):

Definition 2 (Convergent Subterm Theories). *An equational theory is a convergent subterm theory if it is generated from a convergent TRS* $\mathcal{R} = \bigcup_{i=1}^{n}\{L_i >_{r_i} R_i\}$ *where each* R_i *is a proper subterm of* L_i*.*

It follows from the subterm condition (i.e. that R_i is a proper subterm of L_i) that the rewrite system generating convergent subterm rewrite system is also terminating and hence normalising. Since convergence implies confluence, the aforementioned result that $M_1 =_{\mathcal{E}} M_2 \Leftrightarrow M_1{\downarrow} = M_2{\downarrow}$ holds for the theories which we consider in this report.

Recall that a destructor function is a function symbol which occurs as the outermost function symbol in the redex of some rewrite rule (e.g. $dec(\cdot,\cdot)$, $fst(\cdot)$, $snd(\cdot)$ etc.). We shall make one additional assumption about the theories which we consider, namely that rewrite rules are *independent* in the sense that they *only* contain destructor functions as the outermost function symbol; i.e. we disregard rewrite rules such as

$$dec(enc(fst(z_1, z_2), z_3), z_3) > z_1$$

which intuitively says that only encrypted first components of pairs can be decrypted. This assumption is not very strong though, for the vast majority of theories with rules on the above form are not convergent. Take for example the following terms in the theory arising from the above rule:

$$\begin{aligned} M_1 &= dec(enc(fst([a, b]), k), k) \\ M_2 &= dec(enc(a, k), k) \\ M_3 &= a \end{aligned}$$

Then $M_1 >^* M_2$ and $M_1 >^* M_3$ but there is no M_4 such that $M_2 >^* M_4$ and $M_3 >^* M_4$, i.e. confluence and hence convergence fails. An example of a rewrite system which *is* convergent and in which destructor functions occur internally in a redex is given in [1] (where it is used to model the property that decryption keys cannot be guessed):

$$\begin{aligned} dec(enc(z_1, z_2), z_2) &>_{r_1} z_1 \\ enc(dec(z_1, z_2), z_2) &>_{r_2} z_1 \end{aligned}$$

For consider the following term:

$$M_1 = enc(dec(enc(a, k), k), k)$$

Then M_1 can be reduced using both rule r_1 and r_2, but the resulting reduct is always the same (namely $enc(a, k)$), indicating that the rewrite system is convergent. However, disregarding this type of rules does not result in a great loss of generality.

2.2.2 Structural Equivalence

Semantics for Applied π is based on a structural equivalence relation on processes. Here an *evaluation context* is a process with a hole which is not under a replication, a conditional, an input, or an output. *Structural equivalence* $\equiv$ is then defined as the smallest equivalence relation on $\mathcal{P}$ which is closed under α-conversion on names and variables, under application of evaluation context, and which satisfies the following:

Par-0	$P \equiv P\|\mathbf{0}$
Par-A	$P\|(Q\|R) \equiv (P\|Q)\|R$
Par-C	$P\|Q \equiv Q\|P$
Repl	$!P \equiv P\|!P$
New-0	$(\nu n)\mathbf{0} \equiv \mathbf{0}$
New-C	$(\nu u)(\nu v)P \equiv (\nu v)(\nu u)P$
New-Par	$P\|(\nu u)Q \equiv (\nu u)(P\|Q)$ if $u \notin fn(P) \cup fv(P)$
Alias	$(\nu x)\{^M/_x\} \equiv \mathbf{0}$
Subst	$\{^M/_x\}\|P \equiv \{^M/_x\}\|P\{^M/_x\}$
Rewrite	$\{^{M_1}/_x\} \equiv \{^{M_2}/_x\}$ when $M_1 =_{\mathcal{E}} M_2$

Note that $\equiv$ is not a congruence because of the limitations on the type of contexts which may be applied (in effect, only closure under parallel composition and under restriction is required). Note also that the first 7 rules are standard, as given in e.g. [25, Section 2.2], except that there is no scope extrusion for conditionals.

The last three rules give meaning to active substitutions. Alias says that an active substitution with its variable under restriction cannot do anything and, together with Par-**0**, how an arbitrary active substitution can be introduced. Subst formalises the intuition given in the previous subsection that an active substitution may apply its substitution to any process it runs in parallel with, and the Rewrite rule relates structural equivalence and equivalence in the underlying equational theory.

2.2.3 Internal Reduction

An *internal reduction* relation $\rightarrow$, which explains how processes behave, can now be defined based on structural equivalence. In the following, $C[P]$ denotes the evaluation context $C[-]$ applied to the process P:

$$
\begin{array}{ll}
\textsc{Comm} & \overline{a}\langle x\rangle.P|a(x).Q \rightarrow P|Q \\
\textsc{Then} & \dfrac{M_1 =_{\mathcal{E}} M_2}{\text{if } M_1 = M_2 \text{ then } P \text{ else } Q \rightarrow P} \\
\textsc{Else} & \dfrac{M_1 \neq_{\mathcal{E}} M_2}{\text{if } M_1 = M_2 \text{ then } P \text{ else } Q \rightarrow Q} \\
\textsc{Ctx} & \dfrac{P \rightarrow Q}{C[P] \rightarrow C[Q]} \\
\textsc{Struct} & \dfrac{P \equiv P', P \rightarrow Q, Q \equiv Q'}{P' \rightarrow Q'}
\end{array}
$$

Note how equality in the equational theory plays an essential role in the rules involving conditionals. As usual, $\rightarrow^*$ is the transitive closure of $\rightarrow$. The Comm rule appears concerning at a first glance: only variables (and not terms) can be output, and the output seems to disappear in the reduced process. This simplicity of communication in the reduction semantics relies on active substitutions and structural equivalence to work as we shall see in the following example.

Example 2.2.5. We assume the theory $\mathcal{E}_{sym}$ of symmetric encryption and pairing introduced in Example 2.2.2. The following reduction then illustrates the communication mechanism for the case of a pair, as well as the reduction semantics for conditionals.

$$
\begin{array}{l}
\quad\overline{a}\langle[M,s]\rangle \mid a(x).\text{if } snd(x) = s \text{ then } \overline{b}\langle fst(x)\rangle \\
\equiv \overline{a}\langle[M,s]\rangle \mid (\nu x)\{^{[M,s]}/_{x}\} \mid a(x).\text{if } snd(x) = s \text{ then } \overline{b}\langle fst(x)\rangle \\
\equiv (\nu x)(\overline{a}\langle[M,s]\rangle \mid \{^{[M,s]}/_{x}\} \mid a(x).\text{if } snd(x) = s \text{ then } \overline{b}\langle fst(x)\rangle) \\
\equiv (\nu x)(\overline{a}\langle x\rangle \mid \{^{[M,s]}/_{x}\} \mid a(x).\text{if } snd(x) = s \text{ then } \overline{b}\langle fst(x)\rangle) \\
\rightarrow (\nu x)(\text{if } snd(x) = s \text{ then } \overline{b}\langle fst(x)\rangle \mid \{^{[M,s]}/_{x}\}) \\
\equiv (\nu x)(\text{if } snd([M,s]) = s \text{ then } \overline{b}\langle fst([M,s])\rangle \mid \{^{[M,s]}/_{x}\}) \\
\rightarrow (\nu x)(\overline{b}\langle fst\ (\text{[M,s]})\rangle \mid \{^{[M,s]}/_{x}\}) \\
\equiv \overline{b}\langle fst\ (\text{[M,s]})\rangle
\end{array}
$$

□

2.2.4 Labelled Operational Semantics

The labelled operational semantics extends the reduction relation from the previous subsection by explicitly labelling reductions with actions. This allows reasoning about processes which interact with their environment, and is also the basis of a bisimulation relation on processes which we shall introduce later. The relation which shall be defined is on the form $P \xrightarrow{\alpha} P'$ where the label α is either an input $a(M)$ or an output $\overline{a}\langle u\rangle$. The bound output label $(\nu u)\overline{a}\langle u\rangle$ will be used to reflect that the name or variable u is under restriction. The following rules, together with the reduction rules from the previous section, define the labelled operational semantics relation $P \xrightarrow{\alpha} P'$.

In	$a(x).A \xrightarrow{a(M)} A\{^M/_x\}$
Out-Atom	$\overline{a}\langle u\rangle.A \xrightarrow{\overline{a}\langle u\rangle} A$
Open-Atom	$\dfrac{P \xrightarrow{\overline{a}\langle u\rangle} P' \quad u \neq a}{(\nu u)P \xrightarrow{(\nu u)\overline{a}\langle u\rangle} P'}$
Scope	$\dfrac{P \xrightarrow{\alpha} P' \quad u \text{ does not occur in } \alpha}{(\nu u)P \xrightarrow{\alpha} (\nu u)P'}$
Par	$\dfrac{P \xrightarrow{\alpha} P' \quad bv(\alpha)\cap fv(Q) = bn(\alpha)\cap fn(Q) = \emptyset}{P\vert Q \xrightarrow{\alpha} P'\vert Q}$
Struct	$\dfrac{P \equiv Q \quad Q \xrightarrow{\alpha} Q' \quad Q' \equiv P'}{P \xrightarrow{\alpha} P'}$

Note that this in an early semantics, i.e. input actions take immediate effect.

Example 2.2.6. Here is an example of a process which communicates with the environment in the labelled semantics, again assuming the theory $\mathcal{E}_{sym}$ from Example 2.2.2:

$$
\begin{aligned}
&(\nu k)\overline{a}\langle enc(M,k)\rangle.\overline{a}\langle k\rangle.a(z).\text{if } z = M \text{ then } \overline{c}\langle o\rangle \\
\equiv\xrightarrow{(\nu x)\overline{a}\langle x\rangle}&(\nu k)(\{^{enc(M,k)}/_x\}|\overline{a}\langle k\rangle.a(z)\text{if } z = M \text{ then } \overline{c}\langle o\rangle) \\
\equiv\xrightarrow{(\nu y)\overline{a}\langle y\rangle}&(\nu k)(\{^{enc(M,k)}/_x\}|\{^k/_y\}|a(z).\text{if } z = M \text{ then } \overline{c}\langle o\rangle) \\
\xrightarrow{a(dec(x,y))}&(\nu k)(\{^{enc(M,k)}/_x\}|\{^k/_y\}|\text{if } dec(x,y) = M \text{ then } \overline{c}\langle o\rangle) \\
\rightarrow&(\nu k)(\{^{enc(M,k)}/_x\}|\{^k/_y\}|\text{if } dec(enc(M,k),k) = M \text{ then } \overline{c}\langle o\rangle) \\
\rightarrow&(\nu k)(\{^{enc(M,k)}/_x\}|\{^k/_y\}|\overline{c}\langle o\rangle)
\end{aligned}
$$

The structural equivalence in the first two steps is used to introduce an active substitution with a variable under restriction – this restriction then dissappears in the subsequent bound output. □

2.3 Equivalences in Applied π

In this section four notions of equivalence on processes are introduced, namely observational equivalence, two kinds of static equivalence, and labelled bisimilarity.

2.3.1 Observational Equivalence

The first equivalence we define on extended processes is *observational equivalence*. For this purpose we say that a process P has a barb on channel a, written $P \Downarrow a$, if $P \rightarrow^* C[\overline{a}\langle M\rangle.P']$ for some evaluation context $C[-]$.

Definition 3 (Observational Equivalence). *Observational equivalence, $\approx$, is the largest symmetric relation $\mathcal{R}$ on closed extended processes with the same domain such that $P\mathcal{R}Q$ implies:*

1. *if $P \Downarrow a$, then $Q \Downarrow a$.*
2. *if $P \rightarrow^* P'$ then $Q \rightarrow^* Q'$ and $P'\mathcal{R}Q'$ for some Q'.*
3. *$C[P]\mathcal{R}C[Q]$ for all evaluation contexts $C[-]$.*

Observational equivalence corresponds to the *open barbed bisimilarity* studied in [26] by Sangiorgi and Walker. They also show that this bisimilarity does *not* coincide with the barbed congruence in [25, Section 7.2], where closure under application of evaluation contexts is separated from the bisimulation conditions.

The definition of observational equivalence suffers from a universal quantification over all contexts. Later we shall give an equivalence based on the labelled operational semantics which does not have this problem. It depends on the notion of static equivalence which is the subject of the next subsection.

2.3.2 Static Equivalence

We now arrive at two notions which will receive considerable attention in the rest of this report, namely that of *frames* and that of *static equivalence*. Frames, ranged over by φ, are limited forms of extended processes built only from name restriction and active substitutions, and are on the form

$$\varphi = (\nu\tilde{n})\{{}^{M_1}/_{x_1}\} \mid \cdots \mid \{{}^{M_k}/_{x_k}\}$$

where for technical convenience we assume that each M_i is irreducible. Note that this is slightly different from the definition of frames in [4] where restrictions may occur interleaved with active substitutions, terms need not be irreducible and variables may be private in addition to names. A frame can be considered a substitution in which names may be private and can therefore be applied to variables in the usual way. We shall often use the short hand notation $\varphi = (\nu\tilde{n})\{{}^{M_i}/_{x_i}\}_{i\in I}$ or $\varphi = (\nu\tilde{n})\sigma$ (where σ is a substitution) instead of writing out the sequence of active substitutions, and we shall write $(\nu*)\sigma$ for the closed frame where all occurring names are private.

Any extended process P can be mapped to its associated frame, intuitively by deleting all plain processes, extending all restrictions to the outer level and normalising terms in active substitution.

Definition 4. *We define the frame of an extended process A inductively as follows:*

$$\begin{array}{rcll}
\varphi(P) & \triangleq & (\nu\emptyset)\emptyset & \textit{if } P \textit{ is a plain process} \\
\varphi(\{{}^{M}/_{x}\}) & \triangleq & \{{}^{M\downarrow}/_{x}\} & \\
\varphi((\nu n)P) & \triangleq & (\nu\tilde{n}n)\sigma & \textit{where } \varphi(P) = (\nu\tilde{n})\sigma \\
\varphi((\nu x)P) & \triangleq & (\nu\tilde{n})\sigma\sigma & \textit{where } \varphi(P) = (\nu\tilde{n})\sigma \\
\varphi(P_1 \mid P_2) & \triangleq & (\nu\tilde{n}_1\tilde{n}_2)\sigma_1\sigma_2 & \textit{where } \varphi(P_1) = (\nu\tilde{n}_1)\sigma_1 \\
& & & \textit{and } \varphi(P_2) = (\nu\tilde{n}_2)\sigma_2 \\
& & & \textit{and } \tilde{n}_1 \cap n(\sigma_2) = \tilde{n}_2 \cap n(\sigma_2) = \emptyset
\end{array}$$

In the last case α-conversion may be necessary to ensure that $\tilde{n}_1 \cap n(\sigma_2) = \tilde{n}_2 \cap n(\sigma_2) = \emptyset$. The case for private variables states that active substitutions containing private *variables* must be applied to obtain frames which only contain private *names*.

As the example reduction in the previous section showed, each output action performed by a process extends the active substitution. Hence, at any given time, the frame of an extended process P represents the messages communicated by P and thereby also the "static" knowledge available to the environment. This gives rise to an equivalence relation $\approx_s$ on frames called *static equivalence* where, intuitively, two frames are statically equivalent if the environment cannot distinguish them, e.g if two frames represent the same knowledge.

The only way the environment can deduce information from a frame is by testing for equality between terms of the frame. This is captured in the following definition.

Definition 5 (Equality in frames). *Two terms M_1 and M_2 are equal in frame φ, written $(M_1 = M_2)\varphi$ if and only if $n(M_1, M_2) \cap bn(\varphi) = \emptyset$ and $M_1\varphi =_{\mathcal{E}} M_2\varphi$.*

Note the condition that $n(M_1, M_2) \cap bn(\varphi) = \emptyset$, which is necessary for scope extension to work. To see this, take the frame $\varphi = (\nu a)\{{}^{[a,b]}/_{x}\}$ in the theory $\mathcal{E}_1$. An environment may check that $snd(x) = b$ holds in φ as follows:

$$\begin{array}{l}
\text{if } snd(x) = b \text{ then } P_{YES} \text{ else } P_{NO} | (\nu a)\{{}^{[a,b]}/_{x}\} \\
\rightarrow (\nu a)(\text{if } snd(x) = b \text{ then } P_{YES} \text{ else } P_{NO} | \{{}^{[a,b]}/_{x}\}) \\
\rightarrow (\nu a)(\text{if } snd([a,b]) = b \text{ then } P_{YES} \text{ else } P_{NO} | \{{}^{[a,b]}/_{x}\}) \\
\rightarrow (\nu a)(P_{YES} | \{{}^{[a,b]}/_{x}\})
\end{array}$$

Step 2 in this reduction relies on extending the scope of a, which is only possible because a is not free in the terms tested in the conditional. Hence the equality $fst(x) = a$ should *not* hold in φ, since the scope of the name a cannot be extended to any conditional process in which a is free.

We are now ready to give the all-important definition of static equivalence between frames.

Definition 6 (Static equivalence). *Two frames φ and φ' are statically equivalent, written $\varphi \approx_s \varphi'$, if and only if $dom(\varphi) = dom(\varphi')$ and*

$$(M_1 = M_2)\varphi \Leftrightarrow (M_1 = M_2)\varphi' \text{ for all terms } M_1 \text{ and } M_2$$

Furthermore, we say that two processes P and Q are statically equivalent, written $P \approx_s Q$, if and only if $\varphi(P) \approx_s \varphi(Q)$.

Example 2.3.1. We revisit the example frames from the introduction:

$$\begin{aligned}
\varphi_1 &= (\nu k, b)\{{}^{enc(b,k^+)}/_{x_1}, {}^{k^-}/_{x_2}, {}^{b}/_{x_3}\} \\
\varphi_2 &= (\nu k, b)\{{}^{enc(enc(b,b^+),k^+)}/_{x_1}, {}^{k^-}/_{x_2}, {}^{enc(b,b^+)}/_{x_3}\} \\
\varphi_3 &= (\nu k, b)\{{}^{enc(enc(b,k^+),k^+)}/_{x_1}, {}^{k^-}/_{x_2}, {}^{b}/_{x_3}\}
\end{aligned}$$

Then $\varphi_1 \approx_s \varphi_2$ while $\varphi_1 \not\approx_s \varphi_3$, the relevant equality being $dec(x_1, x_2) =_{\mathcal{E}} x_3$. This holds in φ_1 because $dec(enc(b, k^+), k^-) =_{\mathcal{E}} b$, and it holds in φ_2 because $dec(enc(enc(b, b^+), k^+), k^-) =_{\mathcal{E}} enc(b, b^+)$. However, it does not hold in φ_3 because $dec(enc(enc(b, k^+), k^+), k^-) \neq_{\mathcal{E}} b$. □

One could consider defining static equivalence in a more direct manner as follows: $\varphi \approx_s \varphi'$ *iff* $dom(\varphi) = dom(\varphi')$ *and for every pair of terms* M_1, M_2 *with* $v(M_1, M_2) \subseteq v(\varphi)$ *and* $n(M_1, M_2) \cap (bn(\varphi) \cup bn(\varphi')) = \emptyset$ *it holds that* $M_1\varphi =_{\mathcal{E}} M_2\varphi \Leftrightarrow M_1\varphi' =_{\mathcal{E}} M_2\varphi'$. Indeed this is done in e.g. [10], but it is *not* equivalent to the definition we have given here. For with our definition it holds for any two frames φ and φ' that $bn(\varphi) \neq bn(\varphi') \Rightarrow \varphi \not\approx_s \varphi'$. For suppose that $a \in bn(\varphi) - bn(\varphi')$ (where $-$ is the set difference operator). Then the equality $a =_{\mathcal{E}} a$ holds in φ but not in φ'.

The universal quantification over terms in the definition of static equivalence poses problems for practical application. Abadi and Cortier show in [2] that static equivalence is not decidable in general, but they also show that it is decidable for convergent subterm theories as defined in Definition 2. In [3] they give decidability results for a bigger class of theories.

2.3.3 Labelled Bisimilarity

Labelled bisimilarity avoids the universal quantification over contexts which haunts observational equivalence and instead relies on static equivalence.

Definition 7 (Labelled bisimilarity). *$(\approx_l)$ is the largest symmetric relation $\mathcal{R}$ such that $P\mathcal{R}Q$ implies:*

1. *$P \approx_s P'$;*

2. *if $P \to^* P'$ then $Q \to^* Q'$ and $P'\mathcal{R}Q'$ for some Q';*

3. *if $P \xrightarrow{\alpha} P'$ and $fv(\alpha) \subseteq dom(P)$ and $bn(\alpha) \cap fn(Q) = \emptyset$ then $Q \to^* \xrightarrow{\alpha} \to^* Q'$ and $P'\mathcal{R}Q'$ for some Q'.*

It is shown in [4, Theorem 1] that observational equivalence and labelled bisimilarity coincide.

2.3.4 Strong Static Equivalence

A notion of static equivalence has now been defined with the intuition that statically equivalent frames should be indistinguishable by observer processes which are able to test for equality between terms in frames. However, one could question whether equality is the only relevant predicate to test for when deciding static equivalence. Consider the following example frames for motivation:

$$\begin{aligned}\varphi &\triangleq (\nu a, c, k)\{{}^{enc(a,k^+)}/_{x_1}, {}^{k^-}/_{x_2}\}\\ \varphi' &\triangleq (\nu a, c, k)\{{}^{enc(a,k^+)}/_{x_1}, {}^{c^-}/_{x_2}\}\end{aligned}$$

The first frame contains an encrypted name and the corresponding private key which can be used for decryption. The second frame contains the same encrypted name but does not contain the corresponding private key for decryption. Now it turns out that $\varphi \approx_s \varphi'$. The only relevant attempt at constructing a distinguishing equality is as follows:

$$x_1 =_{\mathcal{E}} enc(dec(x_1, x_2), k^+)$$

but this equality does not hold in either frame since $k \in bn(\varphi)$ and $k \in bn(\varphi')$.

The fact that $\varphi \approx_s \varphi'$ might be slightly surprising. For we then get that the following Applied π processes are bisimilar:

$$\begin{aligned}P &\triangleq (\nu a, c, k)(\overline{b}\langle enc(a, k^+)\rangle.\overline{b}\langle k^-\rangle\\ P' &\triangleq (\nu a, c, k)(\overline{b}\langle enc(a, k^+)\rangle.\overline{b}\langle c^-\rangle\end{aligned}$$

Hence a process which sends an encrypted term and subsequently reveals its private decryption key is bisimilar to a (more sound) process which sends an encrypted term but does not leak the corresponding private decryption key! Is this sensible? Well, the intuition that no observer process can distinguish P and P' is intact, since an observer will be able to decrypt the encrypted term from P but will never be in a position to look at the contents (i.e. test for equality) since the name a is private. An observer who tests equality on a would hence not be able to communicate with P since scope extrusion to the observer fails.

Implementations of cryptographic functions (e.g. the OpenSSL library [27]) typically don't provide means of testing whether decryption with a given decryption key is successful or not. However, in many applications it is assumed that such information is available, e.g. by appending a publicly known token to the clear text before encryption; it can then be checked if the decryption output also contains this token.

In other applications including e.g. pairing and lists, strongly typed programming languages such as Java will throw runtime exceptions if e.g. the *fst* projection function is applied to an object M which is not a pair. More specifically, if such object M is received over a network socket, a typecast to the pair type will fail. Hence a principal will be able to detect whether $fst(M)$ is reducible or not.

So reductions (which can also be viewed as computation steps) are most often observable in practical applications: it would be feasible for a process to be able to test if a decryption succeeds or not, even though it may not be able to look at the decrypted term. And it would be feasible for a process to test if application of a projection function to a pair succeeds or not. This motivates us to extend the Applied π syntax with an additional test on reductions thus:

$$P, Q, R ::= ... \mid \text{if } M >^k \text{ then } P \text{ else } Q$$

Intuitively such test succeeds if M can rewrite in k steps to some M'. The definition of internal reduction, $\rightarrow$, can then be extended accordingly:

$$\textsc{Then-2} \qquad \frac{\exists M'.M >^k M'}{\text{if } M >^k \text{ then } P \text{ else } Q \rightarrow P}$$

$$\textsc{Else-2} \qquad \frac{\neg\exists M'.M >^k M'}{\text{if } M >^k \text{ then } P \text{ else } Q \rightarrow Q}$$

Note that a test of the form $M_1 > M_2$ is not strong enough for our purpose. For this test would not be able to distinguish the frames φ and φ' with public-key encrypted terms above. The reason is that M_2 cannot be the private name a, for scope extrusion to an observer making this test would fail.

Before defining a stronger version of static equivalence in which rewrites are observable, we first define the notion of *rewrite in frame* along the same lines as equality in frames.

Definition 8 (Rewrite in frames). *Say that a term M can rewrite in k steps in frame φ, written $(M >^k)\varphi$, if and only if $n(M) \cap bn(\varphi) = \emptyset$ and there exists a term M' such that $M\varphi >^k M'$.*

Definition 9 (Strong static equivalence). *Two frames φ and φ' are strong statically equivalent, written $\varphi \approx_{ss} \varphi'$, if and only if $dom(\varphi) = dom(\varphi')$ and:*

1. *$(M_1 =_{\mathcal{E}} M_2)\varphi \Leftrightarrow (M_1 =_{\mathcal{E}} M_2)\varphi'$ for all terms M_1 and M_2.*

2. *$(M >^k)\varphi \Leftrightarrow (M >^k)\varphi'$ for all terms M.*

Furthermore, we say that two extended processes P and Q are strong statically equivalent, written $P \approx_{ss} Q$, if and only if $\varphi(P) \approx_{ss} \varphi(Q)$.

It is immediate that $\approx_{ss}$ implies $\approx_s$. The example given in the start of this subsection demonstrated that $\approx_{ss}$ does not generally imply $\approx_s$, so the additional test on rewrites increases the distinguishing power of processes. However we shall later pin down a class of equational theories in which also $\approx_s$ implies $\approx_{ss}$, i.e. in which the two definitions coincide. To give a preliminary example of this result, take the symmetric-key counterpart of the frames given earlier:

$$\begin{aligned}\varphi_2 &\triangleq (\nu a, c, k)\{^{enc(a,k)}/_{x_1}, {}^{k}/_{x_2}\}\\ \varphi_2' &\triangleq (\nu a, c, k)\{^{enc(a,k)}/_{x_1}, {}^{c}/_{x_2}\}\end{aligned}$$

Then $\varphi_2 \not\approx_s \varphi_2'$ because the following equality distinguishes the two frames:

$$x_1 =_{\mathcal{E}} enc(dec(x_1, x_2), x_2)$$

Clearly also $\varphi_2 \not\approx_{ss} \varphi_2'$ because $dec(x_1, x_2) >^1$ holds in φ but not in φ'.

2.4 Summary

In this chapter we have introduced syntax and semantics for the Applied π calculus, a uniform and generic extension of the π calculus for which particular primitives such as tuples and cryptographic functions can be defined and used on a per-application basis. The key idea is to integrate arbitrary terms into the calculus and its semantics based on a *function signature*, and to base equality between terms on an *equational theory* instead of syntactic equality on names.

We have seen how a *frame* is obtained by extracting all *active substitutions* of a process, and we have defined an indistinguishability relation $\approx_s$ on frames called *static equivalence*. Roughly speaking, two frames are statically equivalent if an environment cannot distinguish the frames by testing for equality between any pair of terms constructed from variables in the frames. Furthermore, we have defined a stronger static equivalence, $\approx_{ss}$, in which term reductions are observable in addition to equality between terms.

Frames and static equivalence play a central role in labelled bisimilarity (which coincides with observational equivalence) of processes in Applied π: two processes are bisimilar if they can simulate each others actions *and* if their frames are statically equivalent, i.e. if the processes can be distinguished neither on their dynamic behaviour nor on their static characteristics.

Chapter 3

A Refined Definition of Strong Static Equivalence

We now set out to find refined definition of strong static equivalence which provides a suitable basis for a logic for frames and is amenable to construction of characteristic formulae. First we consider some previous work to this end in Section 1 and pay particular attention to work by Boreale et al. on characteristic formulae for frames in the Spi calculus. Section 2 continues with a number of preliminary definitions. These are then used in the refined definition of strong static equivalence given in Section 3. Section 4 demonstrates the refined definition with a number of examples designed to strengthen our belief that the standard and refined definitions of strong static equivalence coincide.

3.1 Previous Work

As mentioned earlier, Abadi and Cortier have shown that static equivalence is decidable for the class of independent convergent subterm theories which was introduced in Subsection 2.2.1 [2, 3]. Therefore one would expect that they at some stage construct a finite set of equalities which is sufficient for deciding static equivalence. This is indeed the case; to decide static equivalence between two frames φ and φ', they associate a finite set of equalities $Eq(\varphi)$ and $Eq(\varphi')$ with each frame such that $\varphi \approx_s \varphi'$ if and only if the two frames satisfy the equalities from each other's set. Hence an alternative definition of static equivalence which does not contain an universal quantification over terms can be made based on this approach. Unfortunately this definition would not serve as a sufficient basis for characteristic formulae. For a characteristic formula must be defined based on a single frame, whereas the above approach defines equivalence based on two specific frames.

Since static equivalence is known to be undecidable for certain theories even when (equational) equality between terms is decidable [2], there are theories in which characteristic formulae do not exists or are not computable (assuming that satisfaction is decidable for the logic in question). For decidability of static equivalence reduces to computing characteristic formulae and deciding satisfaction: to decide if $\varphi \approx_s \varphi'$, compute the characteristic formula $\mathcal{C}_\varphi$ of φ and check if $\mathcal{C}_\varphi \vDash \varphi'$. Hence we focus our efforts on independent convergent subterm theories in which static equivalence is known to be decidable. Although decidability in itself does not guarantee existence of characteristic formulae, we shall learn that characteristic formulae with respect to *strong* static equivalence can indeed be constructed in convergent subterm theories.

In [9] the problem of finding an alternative, tractable definition of static equivalence is attacked for the particular case of (a variant of) the Spi calculus with symmetric key encryption. This definition does yield a construction of characteristic formulae, the key idea being that of a *core*. We will give the intuition of this idea by an example, and in the next section we formalise and generalise the relevant notions in the hope of getting similar results for a wider class of equational theories not restricted to symmetric encryption. Consider the following two frames:

$$\begin{aligned}\varphi &= (\nu a, c, k)\{{}^{enc(enc(a,c),k)}/_{x_1},\ {}^{k}/_{x_2},\ {}^{enc(a,c)}/_{x_3}\}\\ \varphi' &= (\nu a, c, k)\{{}^{enc(a,k))}/_{x_1},\ {}^{k}/_{x_2},\ {}^{a}/_{x_3}\}\end{aligned}$$

Let $M_i = x_i\varphi$ and $M'_i = x_i\varphi'$. Then the core $core(x_i, \varphi)$ of x_i in φ is, intuitively, the largest subterm of M_i which cannot be further decomposed using the information available in φ. In the above example we can apply the decryption function to $M_1 = enc(enc(a, c), k)$ with the key k available through x_2 and get $dec(enc(enc(a, c), k), k) =_\mathcal{E} enc(a, c)$. Hence M_1 is not a core. But the subterm $N_1 = enc(a, c)$ *is* a core, because we cannot apply any decryption function to reveal a since the key c is not available in the frame. The other cores in φ and φ' can be derived similarly and are summarised below, where for brevity we define define $N_i = core(x_i, \varphi)$ and $N'_i = core(x_i, \varphi')$:

$$\begin{aligned}N_1 &= enc(a, c) & N'_1 &= a\\ N_2 &= k & N'_2 &= k\\ N_3 &= enc(a, c) & N'_3 &= a\end{aligned}$$

Observe that $\varphi \approx_s \varphi'$; two interesting equalities which both frames satisfy are $enc(x_3, x_2) =_\mathcal{E} x_1$ (by reflexivity) and $dec(x_1, x_2) = x_3$ (by axiom). There are two key reasons why these equalities hold in both frames:

1. Two cores in φ are equal exactly when the corresponding cores in φ' are equal. Specifically, $N_1 = N_3$ and $N'_1 = N'_3$
2. Corresponding terms M_i and M'_i are equal *up to cores*, in the sense that M_i and M'_i are composed from exactly the same functions applied to their respective cores. Specifically, $M_1 = enc(N_3, N_2)$ and $M'_1 = enc(N'_3, N'_2)$

The alternative, tractable definition of static equivalence given in [9] for frames in the Spi calculus with symmetric key encryption is essentially based on the two observations above.

We will use these ideas as a basis for devising a general refined definition of *strong* static equivalence which coincides with the standard definition of strong static equivalence in *any* independent convergent subterm theory (i.e. not just in symmetric key encryption theories). A straight forward generalisation presents a number of challenges though, many of which become apparent simply by moving from symmetric key encryption to public key encryption with pairs and hash functions. Take for example the counterparts of the above frames with public key encryption, pairing and hash functions in the theory $\mathcal{E}_{pub}$:

$$\varphi_2 = (\nu a, c, k)\{{}^{enc([a,enc(b,c^+)],k^+)}/_{x_1},\ {}^{k^-}/_{x_2},\ {}^{enc(b,c)}/_{x_3}\}$$
$$\varphi_2' = (\nu a, c, k)\{{}^{enc([a,b],f(k)^+)}/_{x_1},\ {}^{f(k)^-}/_{x_2},\ {}^{b}/_{x_3}\}$$

Now how should one define e.g. $core(x_1, \varphi_2)$? The intuition is that the core is the smallest term that can be revealed from x_1, but both a and $enc(b, c^+)$ satisfy this intuition. Hence it would seem that a term can now have several cores. And what about subterms such as the public key k^+ in M_1 – should this be considered a core of x_1? Probably not, since it cannot be revealed from x_1. Hence we would intuitively expect to obtain the following cores of the two frames:

$$\begin{aligned} N_1 &= a & N_1' &= a \\ N_2 &= enc(b, c^+) & N_2' &= b \\ N_3 &= k^- & N_3' &= f(k)^- \\ N_4 &= enc(b, c^+) & N_3' &= b \end{aligned}$$

Lets investigate how the idea with equality between cores and equality of terms up to cores works in our new setting.

1. *Equality between cores.* The equalities $N_2 = N_4$ and $N_2' = N_4'$ do hold as expected. However, important information is lost if we only consider equality between cores. There is for instance no syntactic equality relating the public and private keys. Hence the fact that $snd(dec(x_1, x_2)) =_{\mathcal{E}} x_3$ does *not* follow by syntactic equality between keys as it would in the symmetric key case.

2. *Equality up to cores.* We would like to express that e.g. $M_1 = enc([a, enc(b, c^+)], k^+)$ and $M_1' = enc([a, b], f(k)^+)$ can be written as the same context over cores. This gives rise to a dilemma about how to define the notion of a context. Since k^+ is not a core, we would have to include the private name k in the context and write $M_1 = enc([N_1, N_2], k^+)$, i.e. obtaining the context $enc([y_1, y_2], k^+)$. But this context will not work for M_1' because the encryption key in this term is $f(k)^+$: $M_1' \neq$

$enc([N'_1, N'_2], k^+)$. So it would not appear that the two terms can be written as the same context over cores.

In order to overcome these problems and in order to make our terminology precise, a number of preliminary definitions – including that of a context, revelation from terms and cores – will be given in the next section.

3.2 Initial Definitions

In this section we give some preliminary definitions which are needed in the refined strong static equivalence. We start by making precise the notion of *contexts*. We then define *revelation*, a binary relation which expresses when a term can reveal one of its subterms (e.g. by decryption with an appropriate key). Revelation is a central ingredient in the *analysis* of a frame, which intuitively is the set of all terms that can be revealed from the frame. Finally the analysis leads to a general definition of *cores* in Applied π.

As we demonstrated in the last section, the standard notion of cores is insufficient. Therefore we extend our notion of cores to include more terms from the analysis, and we coin these *ecores* (for *extended cores*). We end the section with a discussion of *correlated variables* and *partitioning contexts*, notions which are required in order to bound the number of contexts considered in the refined definition of strong static equivalence.

3.2.1 Contexts

We start with a definition of contexts.

Definition 10 (Contexts). *A general context $C^{\mathcal{N}}[x_1, \ldots, x_k]$ is a term where $\mathcal{N}$ is a set of names, $v(C^{\mathcal{N}}[x_1, \ldots, x_k]) = \{x_1, \ldots, x_k\}$ and $n(C^{\mathcal{N}}[x_1, \ldots, x_k]) \subseteq \mathcal{N}$. The instantiation $C^{\mathcal{N}}[M_1, \ldots, M_k]$ is the term $C^{\mathcal{N}}[x_1, \ldots, x_k]\{{}^{M_1}/_{x_1}, \ldots, {}^{M_k}/_{x_k}\}$. A context which is just a variable or a name is called a trivial context. In this report we will only consider two special cases, namely contexts $C^{\emptyset}[\tilde{x}]$ (i.e. with no names) and contexts $C^{\{\perp\}}[\tilde{x}]$ which may contain only a distinguished wild card name $\perp$. We shall use the short hand $C[\tilde{x}]$ for the former and $C^{\perp}[\tilde{x}]$ for the latter.*

We shall take the liberty of writing $C^{\mathcal{N}}[\tilde{x}]$ instead of $C^{\mathcal{N}}[x_1, \ldots, x_k]$ when the indices on variables are irrelevant or understood. We shall assume that the sequence $\tilde{y}$ binds to its enclosing context, so if e.g. two contexts $C_1[\tilde{y}]$ and $C_2[\tilde{y}]$ are under consideration, it is not necessarily the case that $v(C_1[\tilde{y}]) = v(C_2[\tilde{y}])$.

3.2.2 Revelation

We now formalise the general idea that a term can reveal one of its subterms, e.g. that a pair can reveal its first and second components because it can be submitted to the first and second projection functions. The first definition expresses which superterm a reduct can be said to originate from, and we call the appropriate superterm the *revelator*.

Definition 11 (Revelator). *Let $L >_r R$ be a rewrite rule and let w be the position of R in L, i.e. $L|_w = R$. Suppose that $M_1 >_r M_2$, i.e. M_1 rewrites primitively to M_2 using rewrite rule r. We then say that $M_1|_{w'}$ is a revelator of the reduction, and we write $M_1 >_r^{M_1|_{w'}} M_2$, if w' is a prefix of w, i.e. if there is a position w'' such that $w = w'w''$.*

Example 3.2.1. Consider the term $M = enc(a, k^+)$ in the public-key theory $\mathcal{E}_{pub}$. The following reductions then hold:

$$dec(M, k^-) > a$$
$$fst([k^+, M]) > k^+$$

Here M is a revelator of the first reduction, and hence a can be revealed from M by using the appropriate private key. However, M is *not* a revelator of the second reduction; this corresponds to the intution that the public key cannot be extracted from a cipher text. □

Definition 12 (Destructor context). *Let $L[\tilde{z}] >_r R[\tilde{z}]$ be a rewrite rule and let $D[x, \tilde{x}]$ be a context which is unifiable with $L[\tilde{z}]$. Let w be the position of the variable x, i.e. $D[x, \tilde{x}]|_w = x$. Then $D[x, \tilde{x}]$ is a destructor context if $L[\tilde{z}]|_w$ is a revelator.*

Using the notions of revelator and destructor contexts, the revelation $M \succ_S M'$ can now be defined. Informally, this asserts that M' is a reduct of some term constructed by applying functions to M with arguments from the set S.

Definition 13 (Revelation Relation). *Let S be a set of terms. The revelation, $\succ_S$, is a binary relation over terms where we define $M \succ_S M|_w$ if and only if there is a destructor context $D[x, \tilde{x}]$ and terms $\tilde{T} \subseteq S$ such that $D[M, \tilde{T}] >_r^M M|_w$.*

It follows from this definition that revelation is based on the rewrite rules of a given TRS. Henceforth we shall always assume that any TRS we consider is equipped with a revelation relation.

Example 3.2.2. Take the set $S = \{enc([b, enc(c, {k_2}^+)], {k_1}^+), [d, k_1^-]\}$ as an example (in the theory $\mathcal{E}_{pub}$). Then we have that:

$$[d, k_1^-] \succ_S d$$
$$[d, k_1^-] \succ_S k_1^-$$
$$enc([b, enc(c, {k_2}^+)], {k_1}^+) \not\succ_S [b, enc(c, {k_2}^+)]$$

The last revelation does not hold since k_1^- is not directly available in the set S and hence cannot be used for revelation in single rewrite-step. □

3.2.3 Analysis

Next we define the *analysis* $\mathcal{A}(M, S)$ of a term M with respect to a set of terms S. $\mathcal{A}(M, S)$ is intuitively the set of terms that M can reveal by repeated revelation from the terms in S. A term may reveal multiple subterms, some of which may be identical, and it will be important for the later development that each instance is represented in the analysis. It will also be important that analysis terms can be ordered by position in their parents, and therefore this information is recorded in the analysis. More precisely, we then get that the analysis $\mathcal{A}(M, S)$ is a set of pairs $(M|_w, w)$ where $M|_w$ is a term revealed from M. Here is the formal definition:

Definition 14 (Term analysis). *Let S be a set of terms and let $M \in S$. Then the analysis of M with respect to S is defined inductively as follows:*

$$\begin{aligned} \mathcal{A}^0(M, S) &= \{(M, \epsilon)\} \\ \mathcal{A}^{i+1}(M, S) &= \mathcal{A}^i(M, S) \cup \{(M|_w, w) | \exists N \in \mathcal{A}^i(M, S) \ s.t.\ N \succ_{\bigcup_{T \in S} \mathcal{A}^i(T,S)} M|_w\} \\ \mathcal{A}(M, S) &= \bigcup_{i \in \omega} \mathcal{A}^i(S) \end{aligned}$$

We further define $\mathcal{A}(M, \varphi) \triangleq \mathcal{A}(M, im(\varphi) \cup fn(\varphi))$ for frames φ.

In many situations the positions included in the analysis are irrelevant and we shall instead consider the *term analysis multiset* $\mathcal{A}^*(M, S)$ defined as

$$\mathcal{A}^*(M, S) = \{M|_w \quad | \quad (M|_w, w) \in \mathcal{A}(M, S)\}$$

Since our point of view will always be apparent from the context, we shall not distinguish between $\mathcal{A}(M, S)$ and $\mathcal{A}^*(M, S)$ in the future.

Example 3.2.3. Let S be the set of terms from the previous example (in the theory $\mathcal{E}_{pub}$). Then

$$\begin{aligned} \mathcal{A}([d, k_1^-], S) =& \{[d, k_1^-], d, k_1^-\} \\ \mathcal{A}(enc([b, enc(c, {k_2}^+)], {k_1}^+), S) =& \{enc([b, enc(c, {k_2}^+)], {k_1}^+), [b, enc(c, {k_2}^+)], b, \\ & enc(c, {k_2}^+)\} \end{aligned}$$

Note that $enc(c, {k_2}^+)$ is part of the second analysis because the decryption key k_1^- is available in the first analysis. However, c is not in the second analysis because the key k_2^- is unknown. □

Sometimes we shall be interested in the combined analysis of an entire set of terms. This *set analysis* is defined as follows:

Definition 15 (Set analysis). *Let S be a set of terms. Then the analysis of S is defined as $\mathcal{A}(S) = \bigcup_{M \in S} \mathcal{A}(M, S)$.*

Example 3.2.4. Continuing the example with the set S from above (in the theory $\mathcal{E}_{pub}$), we get that

$$\mathcal{A}(S) = \{enc([b, enc(c, {k_2}^+)], {k_1}^+), [d, k_1^-], d, k_1^-, [b, enc(c, {k_2}^+)], b, enc(c, {k_2}^+)]\}$$

□

There are of course alternative ways of defining analysis. One could be to give an inductive definition of set analysis and then define term analysis from this. However the analysis of a term M in φ cannot be defined simply as

$$\mathcal{A}(M, S) = \{M' \mid M' \in st(M) \wedge M' \in \mathcal{A}(S)\}$$

since there may be a subterm of M which is not accessible by revelation from M but which can nevertheless be revealed from some other term in S, so a more involved definition would be necessary. Another alternative would be to define analysis as the fixed point of a suitable function, but this is largely a matter of taste.

3.2.4 Cores and Ecores

We are now equipped to give a general definition of *cores* in the Applied π calculus. This is a first step towards defining the terms which are essentially sufficient for deciding strong static equivalence between two frames.

Definition 16 (Core). *The cores of term a M with respect to the set S is the maximum set of terms $(M|_w, w)$ from $\mathcal{A}(M, S)$ where $M|_w$ is $\succ$-irreducible:*

$$cores(M, S) = \{(M|_w, w) \ \mid \ (M|_w, w) \in \mathcal{A}(M, S) \wedge M|_w \not\succ_{\mathcal{A}(S)}\}$$

If $(N, w) \in cores(M, S)$ we call N a core. The cores of a term M in a frame φ is defined as $cores(M, \varphi) = cores(M, im(\varphi) \cup fn(\varphi))$. If $x \in dom(\varphi)$ then $cores(x, \varphi) = cores(x\varphi, \varphi)$. Finally, we define the cores of a frame $\varphi = (\nu\tilde{n})\{^{M_i}/_{x_i}\}_{i \in I}$ as

$$cores(\varphi) = \bigcup_{i \in I} \{(M_i|_w, w, i) \mid (M_i|_w, w) \in cores(M_i, \varphi)\} \cup \{(n, \_, \_) \mid n \in fn(\varphi)\}$$

We will mainly be interested in cores of frames which are defined as sets of triples, again for the purpose of ordering – i.e. we will order cores in a frame by the index of their parent term and by their position in the parent term. We have also defined the cores of a frame to include all free names in the frame (the place holder _ for index and position simply indicates that these values are irrelevant for free names). The reason is that free names are important when deciding static equivalence. Take e.g. the frame $\varphi = \{^{f(a)}/_{x_1}\}$ where a is a free name; then the environment can test that $a = f(a)$ holds in φ, so a should be considered a core in φ. Another motivation for including free names in the core set of a frame is that any frame with free names $n_1, \ldots, n_k$ is statically equivalent to a closed frame (i.e. with no free names) but where each $n_1, \ldots, n_k$ is published as a term of the frame.

As with the analysis we often take the liberty of ignoring the position information and consider cores as multisets of terms instead of pairs or triples. In these cases it always holds that $cores(M,S)$ is a subset of $\mathcal{A}(M,S)$ and that each $M|_w \in cores(M,S)$ is a subterm of M (per definition of revelation).

Example 3.2.5. Take the frame $\varphi = (\nu k_1, b, c)\{^{enc([b,enc(c,k_2{}^+)],k_1{}^+)}/_{x_1}, {}^{[d,k_1^-]}/_{x_2}\}$ which is based on the set S from the previous examples (again in the theory $\mathcal{E}_{pub}$). Then

$$\begin{aligned} cores(enc([b, enc(c, k_2{}^+)], k_1{}^+), S) &= \{b, enc(c, k_2{}^+)\} \\ cores([d, k_1^-], S) &= \{d, k_1^-\} \\ cores(x_1, \varphi) &= \{b, c\} \\ cores(x_2, \varphi) &= \{d, k_1^-\} \\ cores(\varphi) &= \{b, c, d, k_1^-, d, k_2\} \end{aligned}$$

We have that $c \in cores(x_1, \varphi)$ because the name k_2 is free. The key k_2^- is therefore known and can be used in the revelation of c from $enc(c, k_2)$. Note also that d is included twice in the last example. This is because we are working with multisets: d is both revealed from $[d, k_1^-]$ and is free in φ. For the same reason the free name k_2 is also included in the last example. □

The motivation for introducing cores is that these are essentially the terms which matter when deciding static equivalence between two frames φ and φ': two terms $M_i \in im(\varphi)$ and $M_i' \in im(\varphi')$ must be equal up to cores, and whenever two cores N_i, N_j in φ are related in a certain way (such as by equality or rewrite relations) then the corresponding cores N_i', N_j' in φ' must also be related in the same way.

But the basic definition of cores is insufficient for capturing this intuition. Take for instance the following simple frame:

$$\varphi \;\triangleq\; (\nu k)\{^{enc([a,b],k^+)}/_{x_1}, {}^{k^-}/_{x_2}\}$$

Then $cores(x_1, \varphi) = \{a, b\}$, but k^+ is not a core. First of all this means that $M = enc(a, k^+)$ cannot be written as a context over cores (since a context cannot contain the private name k). Second, disregarding k^+ altogether results in a loss of information – the fact that the equality $dec(M, k^-) =_{\mathcal{E}} [a, b]$ holds is lost. The term k^+ is inaccessible in the sense that it cannot be deduced by revelation (it is not in the analysis) but still plays an essential role in deciding static equivalence. Or put differently, if we were to replace k^+ with some other arbitrary term, this would change the analysis of φ.Let us make precise the notion of inaccessible terms.

Definition 17 (Inaccessible Terms). *Let φ be any frame and let $M \in im(\varphi)$. A sub-term $M|_w$ is inaccessible in M if each of the following conditions hold:*

1. $M|_w \notin \mathcal{A}(\varphi)$.
2. *for all* $N \in cores(\varphi)$ *and* $M^* \in st(M|_w)$ *it holds that* $M^* \notin st(N)$.

We have argued that basic cores as defined above are insufficient in the presence of inaccessible terms – the inaccessible terms should somehow be taken into account when deciding static equivalence. Inaccessible terms must still have a different status than cores though, for they cannot be "dug out" and used in new settings since they are not part of the analysis; they are stuck in their parent terms. So in addition to cores, we should also take appropriate analysis terms, which contain inaccessible terms, into account. This prompts us to define *extended cores* thus:

Definition 18 (Extended cores). *A term* $M|_w \in \mathcal{A}(M, S)$ *is a extended core, or ecore, with respect to the set* S *if either*

1. $(M|_w, w) \in core(M, S)$.
2. *There is no context* $C[x_1, \ldots, x_k]$ *and no terms* $M_1, \ldots, M_k \in \mathcal{A}(S)$ *such that* $M|_w = C[M_1, \ldots, M_k]$.

The set of extended cores of M *with respect to* S *is denoted by* $ecore(M, S)$*, and as for cores we define* $ecores(M, \varphi) = ecores(M, im(\varphi) \cup fn(\varphi))$ *and* $ecores(x, \varphi) = ecores(x\varphi, \varphi)$*. Finally, we define the ecores of a frame* $\varphi = (\nu\tilde{n})\{{}^{M_i}/_{x_i}\}_{i\in I}$ *as*

$$ecores(\varphi) = \bigcup_{i\in I}\{(M_i|_w, w, i) \mid (M_i|_w, w) \in ecores(M_i, \varphi)\} \cup \{(n, \_, \_) \mid n \in fn(\varphi)\}$$

Again we often disregard the position information in ecores. Note that if a term M does not have any inaccessible subterms then $ecores(M, \varphi) = cores(M, \varphi)$. Let us consider some examples.

Example 3.2.6. Take the frame $\varphi \triangleq (\nu*)\{{}^{[enc(a,{k_1}^+), enc(b,{k_2}^+)]}/_{x_1}, {}^{k_1^-}/_{x_2}\}$ in the theory $\mathcal{E}_{pub}$. Then

$$cores(x_1, \varphi) = \{a, enc(b, {k_2}^+)\}$$
$$ecores(x_1, \varphi) = \{enc(a, {k_1}^+), enc(b, {k_2}^+), a\}$$

Note how each term in $im(\varphi)$ can now be written as a context over ecores, and how the inaccessible terms are included in ecores while still located in their parent terms. Another way of putting it is that *the analysis can be reconstructed from extended cores* by applying appropriate function symbols! □

Example 3.2.7. Take the frame $\varphi \triangleq (\nu*)\{{}^{enc(enc(a,k^+),k^+)}/_{x_1}, {}^{k^-}/_{x_2}\}$ in the theory $\mathcal{E}_{pub}$. Then

$$cores(x_1, \varphi) = \{a\}$$
$$ecores(x_1, \varphi) = \{enc(enc(a, k^+), k^+), enc(a, k^+), a\}$$

Note that, in this particular example, $ecores(x_1, \varphi) = \mathcal{A}(x_1, \varphi)$. Also note that ecores may have other ecores as subterms, which is not the case for standard cores. □

In contrast to cores as defined in the Spi calculus, $ecores(x, \varphi)$ in our new definition is a set, which is natural when considering theories with e.g. pairing. But in the following development it will be necessary to impose a linear ordering on ecores for the purpose of comparing cores with the same index in different frames, which is exactly why we have gone through the trouble of including position information in the definition of ecores.

To define a linear ordering on ecores, first let $<$ be the standard less-than ordering on integers and let $\lll$ be the standard lexiographic ordering relation on positions (i.e. $w_1 \lll w_2$ iff w_1 is lexiographically less than w_2). Assume also an arbitrary but fixed ordering $\lll\!\!\ll$ on names. An ordering on ecores can then be defined as follows.

Definition 19 (Ordering of ecores). *Let $\varphi = (\nu\tilde{n})\{^{M_i}/_{x_i}\}_{i\in I}$ be any frame and let $(N_1, w_1, i_1), (N_2, w_2, i_2) \in ecores(\varphi)$. Define $N_1 \ll N_2$ iff either of the following hold:*

1. $i_1 < i_2$.
2. $i_1 = i_2$ *and* $w_1 \lll w_2$
3. $w_2 = i_2 = \_$ *and* $w_1 \neq \_$ *and* $i_1 \neq \_$
4. $w_1 = i_1 = w_2 = i_2 = \_$ *and* $N_1 \lll\!\!\ll N_2$

Informally, order between parent terms has highest priority, position within identical terms has second priority, and ordering on (free) names has third priority.

From now on we will consider ecores as sequences (ordered by $\ll$) expressed using the notation $ecores(\varphi) = N_1, \ldots, N_k$, or $ecores(\varphi) = (N)_{i\in J}$, but will still take the liberty of using set-theoretic notions such as membership. The specifics in the definition of $\ll$ will play no further role; it has been given above in order to demonstrate that an ordering is indeed possible and well-defined.

3.2.5 Correlated Variables

In the refined definition of strong static equivalence to be given in the next section, we shall consider not only appropriate relationships (such as equality and rewrites) between cores, but also relationships between *contexts over cores*. In order to limit the complexity of this refined definition and subsequently give a construction of finite characteristic formula, it is crucial that we only consider a restricted class of contexts. To that end we start by defining the notion of correlation between positions and variables in a context.

Definition 20 (Correlated positions and variables). *Let $L[\tilde{z}] > R[\tilde{z}]$ be a rewrite rule and let $C[\tilde{y}]$ be a context which is unifiable with $L[\tilde{z}]$. Let w_i and w_j be two positions in $C[\tilde{y}]$ and let w'_i and w'_j be the longest prefixes of w_i, respectively w_j, such that the position w'_i, respectively w'_j, exists in $L[\tilde{z}]$. We then say that w_i and w_j are strongly correlated, written $w_i \rightleftarrows w_j$, if $v(C[\tilde{z}]|_{w'_i}) \cap$*

$v(C[\tilde{z}]|_{w_j'}) \neq \emptyset$. If $z \in v(C[\tilde{z}]|_{w_i'}) \cap v(C[\tilde{z}]|_{w_j'})$ we further say that w_i and w_j are correlated through z and write $w_i \rightleftarrows_z w_j$.

Strong correlation is a reflexive and symmetric binary relation. Define weak correlation, $\rightleftharpoons$, to be the transitive closure of strong correlation and say that w_i, w_j are weakly correlated if $w_i \rightleftharpoons w_j$.

For each pair of variables $y_i, y_j \in v(C[\tilde{y}])$ let $\tilde{w_i}$ and $\tilde{w_j}$ be the positions of y_i and y_j in $C[\tilde{y}]$, respectively (there will be multiple positions if a variable has multiple occurrences). We then say that y_i and y_j are strongly (respectively weakly) correlated if every $w_i \in \tilde{w_i}$ is strongly (respectively weakly) correlated to every $w_j \in \tilde{w_j}$.

Correlation is always relative to both a context and a rewrite rule. The definition may seem a little technical at first, but it really just pins down the natural idea that correlated variables are the ones which depend on each other in rewrites. Let us consider an example.

Example 3.2.8. Consider the following extended public key rewrite rule from the theory $\mathcal{E}_{pub2}$ (introduced first in Example 2.2.4):

$$dec(enc(z_1, f(z_2{}^+, z_3{}^+)), f(z_2^-, z_3^-)) > z_1$$

and the context

$$dec(enc(y_1, y_2), f(y_3, y_4^-))$$

Then y_1 is neither strongly nor weakly correlated to any other variables. y_3 and y_4 are strongly correlated to y_2 but not to each other. They are however weakly correlated to each other.
Suppose now that the instance $dec(enc(N_1, N_2), f(N_3, N_4^-))$, where $N_1, \ldots, N_4$ are some cores, is an instance of the LHS of the above rewrite rule and hence is reducible. The fact that y_1 is not correlated to any other variables means that we can substitute N_1 for any other term N_1' and the the rewrite will still be possible. If instead we substitute N_2 for some N_2' the rewrite may no longer be possible, since correlation between y_2 and the other variables dictates a syntactic constraints on N_2 in relation to the other cores (for instance N_2 must contain N_3 and N_4 as subterms). □

The last example demonstrated how rewrites intuitively depend only on the instantiation of correlated variables. The notion of correlation can therefore be used to define the type of contexts which are relevant for the upcoming refined definition of static equivalence. The contexts which we are interested in for this purpose are the simplest contexts (in a certain precise sense) in which all variable positions are correlated; this intuition is formalised in the following definition of *partitioning contexts*.

Definition 21 (Partitioning Contexts). *A context $C^\perp[\tilde{y}]$ is a partitioning context if it is unifiable with the LHS of some rewrite rule $L[\tilde{z}] > R[\tilde{z}]$ and all of the following hold:*

1. *$y_i \rightleftharpoons y_j$ for all $y_i, y_j \in v(C^\perp[\tilde{y}])$.*

2. *$C^{\perp}[\tilde{y}]|_w = \perp$ implies $L[\tilde{z}]|_w = z$ for all w and some $z \in v(L[\tilde{z}])$.*

3. *$w \neq w_i$ for all positions w and w_i where $C^{\perp}[\tilde{y}]|_w = \perp$ and $C^{\perp}[\tilde{y}]|_{w_i} = y_i$ for some $y_i \in v(C^{\perp}[\tilde{y}])$.*

The first condition says that all variables in the context must be weakly correlated. The second condition says that all occurrences of $\perp$ must unify trivially with a variable in the rewrite rule. And the third condition says that no variables are correlated to any occurrence of $\perp$; this is a way of expressing that the context is maximal in the sense that it cannot be extended with further variables.

Example 3.2.9. Consider again the extended public key rewrite rule from $\mathcal{E}_{pub2}$:

$$dec(enc(z_1, f(z_2{}^{+}, z_3{}^{+})), f(z_2^{-}, z_3^{-})) > z_1$$

Then the following are examples of contexts which can easily verified to be partitioning:

- $dec(enc(y_1, \perp^{+}, \perp^{+})), f(\perp^{-}, \perp^{-}))$.
- $dec(enc(\perp, y_1), f(y_2, y_3^{-}))$.
- $dec(enc(\perp, f(\perp^{+}, y_1{}^{+})), f(\perp^{-}, f(y_2)^{-}))$.

The following are examples of contexts which are *not* partitioning:

- $dec(enc(y_1, y_2), f(y_3, y_4^{-}))$ is not partitioning because y_1 is not strongly correlated to any of the other variables (so condition 1 fails).
- $dec(enc(f(\perp), f(y_1{}^{+}, \perp^{+})), f(y_2, \perp^{-}))$ is not partitioning because the first position of $\perp$ does not match the position of z_1 (so condition 2 fails).
- $dec(enc(\perp, f(\perp^{+}, \perp^{+})), f(y_2, \perp^{-}))$ is not partitioning because there is a related context with more correlated variables which is partitioning, e.g. $dec(enc(\perp, f(y_1, \perp^{+})), f(y_2^{-}, \perp^{-}))$ (so condition 3 fails).

□

The term *partitioning context* is chosen because correlation is an equivalence relation and hence partitions variables into equivalence classes; all variables in a partitioning context are then in the same equivalence class. Moreover, each equivalence class gives rise to a *generated partitioning context*, as defined next.

Definition 22 (Generated Partitioning Contexts). *Let $C[\tilde{y}]$ be some context which unifies with the LHS of some rewrite rule $L[\tilde{z}] >_r R[\tilde{z}]$ and let $\tilde{y}_1, \ldots, \tilde{y}_k$ be the partition of $\tilde{y}$ under $\rightleftharpoons$.*

Then each equivalence class $\tilde{y}_i$ generates a (unique) partitioning context $C_i^{\perp}[\tilde{y}_i]$ defined as follows:

1. *$C_i^{\perp}[\tilde{y}_i]$ is a partitioning context with respect to the rewrite rule r.*

2. *If* $C^{\perp}[\tilde{y}]|_w = y$ *then* $C_i^{\perp}[\tilde{y}_i]|_w = y$ *for all* $y \in \tilde{y}_i$ *and for all positions* w.

Example 3.2.10. Take again the rewrite rule extended public key rewrite rule from $\mathcal{E}_{pub2}$

$$dec(enc(z_1, f(z_2{}^{+}, z_3{}^{+})), f(z_2^{-}, z_3^{-})) >_r z_1$$

and consider the following context which unifies with the LHS of this rule:

$$dec(enc(y_1, y_2), f(y_3, y_4^{-}))$$

As we saw in an earlier example there are two variable equivalence classes, $\tilde{y}_1 = \{y_1\}$ and $\tilde{y}_2 = \{y_2, y_3, y_4\}$. They generate the following two partitioning contexts:

$$\begin{aligned} C_1^{\perp}[\tilde{y}_1] &= dec(enc(y_1, f(\perp^{+}, \perp^{+})), f(\perp^{-}, \perp^{-})) \\ C_2^{\perp}[\tilde{y}_2] &= dec(enc(\perp, y_2), f(y_3, y_4^{-})) \end{aligned}$$

$\square$

In words, if $L[\tilde{z}] >_r R[\tilde{z}]$ is a rewrite rule and the context $C[\tilde{y}]$ unifies with $L[\tilde{z}]$, then the partitioning context generated from the equivalence class $\tilde{y}_i$ is obtained by preserving the part of $C[\tilde{y}]$ which contains $\tilde{y}_i$ and simplifying the rest with $\perp$ to match $L[\tilde{z}]$. So intuitively, any generated partitioning context will also unify with $L[\tilde{z}]$. Furthermore, if $C[\tilde{N}]$ (i.e. the context instantiated with cores) unifies with $L[\tilde{z}]$, then so will the instance $C_i[\tilde{N}]$ of every genererating partitioning context. This observation will play a key role in the proofs to be given later in the report.

3.3 The Refined Definition

In this section a refined definition of strong static equivalence which is amenable to the construction of characteristic formulae will be given based on the notions introduced in the last section.

First recall from Subsection 2.2.1 that a term M_1 is *less general* than a term M_2 if there is some substitution θ such that $M_1 = M_2\theta$; if this is the case, we write $M_1 \sqsubset M_2$ in the following. Similarly, we say that M_1 is *more general* than M_2, written $M_1 \sqsupset M_2$, if there exists some substitution θ such that $M_2 = M_1\theta$. Finally, we shall write $M_1 \square M_2$ to mean that M_1 and M_2 are unifiable but neither is more or less general than the other, i.e. there exists a unifying substitution θ such that $M_1\theta = M_2\theta$, $M_1 \not\sqsupset M_2$ and $M_1 \not\sqsubset M_2$.

We shall mainly be interested in asserting generality on contexts in relation to the LHS of some rewrite rule. For this purpose we also define $\sqsubset$, $\sqsupset$ and $\square$ as unary relation symbols, and write e.g. $C[\tilde{y}]\square$ if there exists a rewrite rule $L[\tilde{z}] >_r R[\tilde{z}]$ in the relevant TRS such that $C[\tilde{y}]\square L[\tilde{z}]$ (and similarly for $\sqsubset$ and $\sqsupset$). With these notions we are now ready to state the refined definition of strong static equivalence.

Definition 23 (Refined Strong Static Equivalence). *Let* $\varphi \equiv (\nu\tilde{n})\{{}^{M_i}/_{x_i}\}_{i\in I}$ *and* $\varphi' \equiv (\nu\tilde{n})\{{}^{M_i'}/_{x_i}\}_{i\in I}$ *be two frames with* $ecores(\varphi) = (N)_{j\in J}$, $ecores(\varphi') = (N')_{j\in J}$. *Then* φ *and* φ' *are strong statically equivalent, written* $\varphi \approx'_{ss} \varphi'$, *if* $dom(\varphi) = dom(\varphi')$ *and each of the following conditions hold:*

1. *For each* $i \in I$ *there is some context* $C[\tilde{y}]$ *such that*
   - $M_i = C[\tilde{N}]$
   - $M_i' = C[\tilde{N'}]$
2. *For any context* $C[\tilde{y}]$ *and for all* $j \in J$ *it holds that*
$$C[\tilde{N}] = N_j \Leftrightarrow C[\tilde{N'}] = N_j'$$
3. *For any partitioning context* $C_1^\perp[\tilde{y}]$ *where* $C_1^\perp[\tilde{y}] \sqsupset$ *or* $C^\perp[\tilde{y}]\Box$ *it holds for all* $C_2^\perp[\tilde{y}]$ *that*
$$C_1^\perp[\tilde{N}] >_r C_2^\perp[\tilde{N}] \Leftrightarrow C_1^\perp[\tilde{N'}] >_r C_2^\perp[\tilde{N'}]$$

Condition 1 requires that each $M_i \in im(\varphi)$ and $M_i' \in im(\varphi')$ are equal up to ecores. For this to be well defined, there must be at least one context $C[\tilde{y}]$ such that $M_i = C[\tilde{N}]$, i.e. all terms in a frame can be written as a context over ecores. This fact is established in the following easy lemma; indeed a slightly stronger result is proven, namely that any *analysis term* can be written as a context over ecores.

Lemma 1. *Let* φ *be any frame with* $ecores(\varphi) = (N)_{j\in J}$. *Then for any* $M \in \mathcal{A}(\varphi)$ *there exists a context* $C[\tilde{y}]$ *such that* $M = C[\tilde{N}]$.

Proof. By induction in the height of the parse tree of M.

Basis. Then $M = a$ for some name a. If $a \in bn(\varphi)$ then $a = N_j$ for some ecore N_j since a can only be included in the analysis by revelation and a cannot be further reduced by revelation. Hence the context is the trivial context $C_i[\tilde{y}] = y_j$. If $a \in fn(\varphi)$ then a is per definition an ecore, so again the context is the trivial context.

Step. If M is an ecore, i.e. $M = N_j$ for some $j \in J$, then the context is the trivial context $C_i[\tilde{y}] = y_j$. Otherwise it follows from the definition of ecores that there must be some context $C[x_1, \ldots, x_k]$ and terms $M_1, \ldots, M_k \in \mathcal{A}(\varphi)$ such that $M = C[M_1, \ldots, M_k]$. Applying the induction hypothesis to each M_i we get that $M_i = C_i[\tilde{N}]$ for some context $C_i[\tilde{y}]$ and $\tilde{N} \subseteq ecores(\varphi)$. Hence $M = C[C_1[\tilde{N}], \ldots, C_2[\tilde{N}]]$ which concludes the proof. ☐

This lemma, together with condition 2, gives that condition 1 in fact holds for *all* contexts.

One may fear that this refined definition $\approx'_{ss}$ is no better than $\approx_{ss}$, since conditions 2 and 3 contain universal quantifications over contexts. When constructing characteristic formulae for a frame φ it is necessary to express both which equalities from condition 2 on the form $C[\tilde{N}] = N_j$ *do* hold, but also which do *not* hold. The key point here is that there are only finitely many equalities on the form $C[\tilde{N}] = N_j$ which do hold since there are only a finite number of cores each of finite size. We will see in the next chapter how an appropriate semantics for quantifiers in our first-order logic of frames will allow us to give a finite formulae expressing all the negative equalities.

A similar concern arises in condition 3. There are only finitely many partitioning contexts which are more general than the LHS of some rewrite rule, but there are infinitely many contexts which are unifiable with, but neither less general than nor more general than, the LHS of some rewrite rule. Take for example the public key rewrite rule; then the following infinite sequence of partitioning contexts satisfy this:

$$\begin{aligned}
&dec(enc(\perp, y_1), y_2),\\
&dec(enc(\perp, {y_1}^+), y_2),\\
&dec(enc(\perp, f(y_1)^+), y_2),\\
&dec(enc(\perp, f(f(y_1))^+), y_2),\\
&\ldots
\end{aligned}$$

The key point is that because the contexts are partitioning contexts and because there are only finitely many many cores, there are a finite number of instantiations which are reducible. Quantification in the logic will then allow us to give finite formulae expressing all the instantiations which are not reducible.

If we had also included in condition 3 contexts which are less general than the LHS of some rewrite rule, then the situation would be different. For there would be infinitely many instantiations of this kind of context which *are* reducible as well as infinitely many instantiations which are *not* reducible. Luckily these contexts need not be taken into account explicitly in the definition, for as we shall see later a rewrite condition for this kind of context follows from condition 2.

3.4 Examples

The conditions in $\approx'_{ss}$ have been carefully devised with the aim that $\approx'_{ss}$ and $\approx_{ss}$ should coincide. That this is indeed the case is not trivially obvious. The next section develops a proof of coincidence between the two definitions, but first we proceed in this section with some motivating examples which shed some light onto the workings of the refined definition and gives us confidence in its sanity. The examples will be based on the theories of symmetric key encryption $\mathcal{E}_{sym}$, public key encryption $\mathcal{E}_{pub}$ and extended public key encryption $\mathcal{E}_{pub2}$ which were introduced in Chapter 2. Whenever we consider two frames φ and φ' we will let N_i range over $ecores(\varphi)$ and N'_i range over $ecores(\varphi')$.

Example 3.4.1. Consider the following frames in the theory $\mathcal{E}_{sym}$:

$$\varphi = (\nu a, k, c)\{^{enc(enc(a,k),c)}/_{x_1},\ ^{c}/_{x_2}\}$$
$$\varphi' = (\nu a, k, c)\{^{enc(enc(enc(a,k),k),c)}/_{x_1},\ ^{c}/_{x_2}\}$$

These are clearly strong statically equivalent, and indeed all the conditions of $\approx'_{ss}$ are satisfied. The ecores are as follows:

$$N_1 = enc(a,k) \qquad N'_1 = enc(enc(a,k),k)$$
$$N_2 = c \qquad N'_2 = c$$

This illustrates the intuition that the two frames are equal up to ecores (i.e. condition 1 holds) and that ecores cannot be distinguished by testing syntactic equality or by testing reductions. □

Example 3.4.2. Consider the following frames, again in the theory $\mathcal{E}_{sym}$, which are identical to the frames from the last example except an unknown key k in φ' has been replaced by a known key c:

$$\varphi = (\nu a, c, k)\{^{enc(enc(a,k),c)}/_{x_1},\ ^{c}/_{x_2}\}$$
$$\varphi\prime = (\nu a, c, k)\{^{enc(enc(enc(a,k),c),c)}/_{x_1},\ ^{c}/_{x_2}\}$$

This change results in the two frames not being strong statically equivalent because the equality

$$x_1 =_{\mathcal{E}} enc(dec(dec(x_1, x_2), x_2), x_2)$$

holds in φ' but not in φ. The ecores of the two frames are

$$N_1 = enc(a,k) \qquad N'_1 = enc(a,k)$$
$$N_2 = c \qquad N'_2 = c$$

Condition 1 of $\approx'_{ss}$ then captures the difference between the two frames because the first term from φ can only be written as $enc(N_1, N_2)$ while the first term from φ' can only be written as $enc(enc(N_1, N_2), N_2)$, and the relevant contexts are thus not the same. Note however that the cores in the two frames are exactly the same (namely a and c), so φ and φ' could not have been distinguished using only condition 2 and 3 in $\approx'_{ss}$. Hence condition 1 is strictly necessary. □

Example 3.4.3. Consider the following frames in the theory $\mathcal{E}_{pub}$, which are similar to the frames from the previous example except that they use public key encryption instead of symmetric key encryption:

$$\varphi = (\nu a, c, k)\{^{enc(enc(a,k^+),c^+)}/_{x_1},\ ^{c^-}/_{x_2}\}$$
$$\varphi' = (\nu a, c, k)\{^{enc(enc(enc(a,k^+),c^+),c^+)}/_{x_1},\ ^{c^-}/_{x_2}\}$$

This time the two frames should not be strong statically equivalent because the term

$$dec(dec(x_1, x_2), x_2)$$

can reduce in two steps in φ' but only in one step in φ. The public key k^+ is not in the analysis of the above frames, so the ecores are given as follows:

$$\begin{aligned}
N_1 &= enc(enc(a,k^+),c^+) & N_1' &= enc(enc(enc(a,k^+),c^+),c^+)\\
N_2 &= enc(a,k^+) & N_2' &= enc(enc(a,k^+),c^+)\\
& & N_3' &= enc(a,k^+)\\
N_4 &= c^- & N_4' &= c^-
\end{aligned}$$

(note that N_3 does not exist in φ). The two frames are then distinguished by condition 3 which fails because $dec(N_2', N_4') > N_3'$, while this does not hold for the corresponding cores from φ. Note that the context employed in this reduction is valid for use in condition 3 of $\approx_{ss}'$ (it is partitioning and less general than the LHS of the decryption rule).

Note also that condition 1 in $\approx_{ss}'$ cannot be used to distinguish the two frames as it could in the previous example. For the empty trivial context $C[\tilde{y}] = y_1$ satisfies condition 1: $M_1 = N_1$ and $M_1' = N_1'$.

This example thus illustrates the crucial role of extended cores and their contained inaccessible terms: if k^+ were not retained in an ecore then no rewrite could distinguish φ and φ'. □

Example 3.4.4. Consider the following very simple frames (in any theory with hash functions $f(\cdot)$ and $g(\cdot)$) where the name a is free:

$$\begin{aligned}
\varphi &= \{ {}^{f(a)}/_{x} \}\\
\varphi' &= \{ {}^{g(a)}/_{x} \}
\end{aligned}$$

These frames are not strong statically equivalent because the equality $f(a) = x$ holds in φ but not in φ'. An environment can test this equality because the name a is free.

Recall now that we have defined free names to be ecores and hence we have that

$$\begin{aligned}
N_1 &= f(a) & N_1' &= g(a)\\
N_2 &= a & N_2' &= a
\end{aligned}$$

and condition 2 can now be used to distinguish the two frames since $f(N_2) = N_1$ holds but $f(N_2') = N_1'$ does not.

This example hence illustrates the necessity of condition 2 and that taking contexts into account is indeed necessary. If condition 2 were weakened to require only that two cores N_i and N_j are equal if and only if N_i' and N_j' are equal (as in [9]), frames involving hash functions akin to φ and φ' would not be distinguished. This would have wide implications for any theories which may exhibit hash-like behaviour, such as public key encryption would do in cases where a public key is known but the corresponding private key is not. □

Example 3.4.5. Consider the following frames (in a theory with pairing, e.g. $\mathcal{E}_{sym}$):

$$\begin{aligned}\varphi &= (\nu a, b)\{\,{}^{[a,b]}/_{x_1},\ {}^{[a,b]}/_{x_2}\} \\ \varphi' &= (\nu a, b)\{\,{}^{[a,b]}/_{x_1},\ {}^{[b,a]}/_{x_2}\}\end{aligned}$$

These are not strong statically equivalent because $fst(x_1) =_{\mathcal{E}} fst(x_2)$ holds in φ but not in φ'. Again this is captured by the the second condition of $\approx'_{ss}$; we have that

$$\begin{aligned} N_1 &= a & N'_1 &= a \\ N_2 &= b & N'_2 &= b \\ N_3 &= a & N'_3 &= b \\ N_4 &= b & N'_4 &= a \end{aligned}$$

so $N_1 = N_3$ but $N'_1 \neq N'_3$. Hence, here the relevant context is just the trivial context y_1 instantiated to the core N_1. It follows that condition 2 also requires, as a special case, that whenever two cores in φ are equal, the corresponding cores in φ' must also be equal. This example also serves to illustrate than an ordering on cores is indeed necessary. $\square$

Example 3.4.6. Consider the following frames in a theory with hash functions $f(\cdot)$ and $g(\cdot)$:

$$\begin{aligned}\varphi &= (\nu a, b)\{\,{}^{f(a)}/_{x_1},\ {}^{g(b)}/_{x_2}\} \\ \varphi' &= (\nu a, b)\{\,{}^{a}/_{x_1},\ {}^{b}/_{x_2}\}\end{aligned}$$

These are clearly strong statically equivalent, and it is easy to see that all conditions of $\approx'_{ss}$ hold. Suppose however that we are working in a theory with the rewrite rule

$$f(z_1) + g(z_2) >_r f(z_1)$$

In that case $\varphi \not\approx_s \varphi'$ since the equality $x_1 + x_2 =_{\mathcal{E}} x_1$ holds in φ but not in φ'. This example serves to illustrate the need for condition 3 which captures this equality, and without which the two frames *would* be strong statically equivalent according to the new definition.

$\square$

3.5 Summary

We have given a refined version of strong static equivalence which goes some of the way towards a construction of characteristic formulae, although the refined definition still relies on quantification over a certain type of contexts (namely partitioning contexts).

In support of the refined definition, we have introduced a number of concepts: *revelation* is a binary relation which expresses when a term can reveal one of its subterms (e.g. by decryption with an appropriate key); the *analysis* of a frame is is the set of terms obtained by repeated revelation from the frame; *cores* are the terms from the analysis which are minimal under revelation; *ecores* are an extension of cores with additional information indispensable for deciding strong static equivalence; *correlated variables* and *partitioning contexts* are required to bound the number of contexts considered in the refined definition of strong static equivalence.

Finally, a number of examples have been given which have strengthened our belief in the refined definition of strong static equivalence.

Chapter 4

Results On The Refined Definition

The refined definition of strong static equivalence, $\approx'_{ss}$, presented in the previous chapter has been devised with the intention that it should coincide with $\approx_{ss}$. In this chapter we prove that this indeed is the case. Section 1 shows one direction, namely that $\approx'_{ss}$ implies $\approx_{ss}$ (or more precisely that $\approx'_{ss}\subseteq\approx_{ss}$), while Section 2 shows the other direction that $\approx_{ss}$ implies $\approx'_{ss}$ ($\approx_{ss}\subseteq\approx'_{ss}$).

We have already seen that $\approx_{ss}$ and $\approx_{s}$ do not coincide in general. But in theories in which inaccessible terms do not exist, including symmetric key theories and public key theories where public keys are always known, they *do* coincide. We show this result in Section 3.

4.1 $\approx'_{ss}$ implies $\approx_{ss}$

The main theorem of this section states that $\approx'_{ss}$ implies $\approx_{ss}$. In order to prove this result a number of preliminary lemmas are required. One of the key lemmas states that if $\varphi \approx'_{ss} \varphi$ then it holds for any contexts $C_1[\tilde{y}]$ and $C_2[\tilde{y}]$ that

$$C_1[\tilde{N}] =_{\mathcal{E}} C_2[\tilde{N}] \Leftrightarrow C_1[\tilde{N}'] =_{\mathcal{E}} C_2[\tilde{N}']$$

The proof of this in turn relies on a lemma stating that rewrites are preserved by substitution of cores:

$$C_1[\tilde{N}] > C_2[\tilde{N}] \Leftrightarrow C_1[\tilde{N}'] > C_2[\tilde{N}']$$

and a lemma stating that syntactic equality is preserved by substitution of cores:

$$C_1[\tilde{N}] = C_2[\tilde{N}] \Leftrightarrow C_1[\tilde{N}'] = C_2[\tilde{N}']$$

We start by attacking the latter result, which is expressed formally in the following lemma.

Figure 4.1: Illustration of the two cases and the terms involved.

$$C_1[- - - \overbrace{- - \underbrace{* * * * *}_{\text{discrepancy at } w} - -}^{C_1[\tilde{N}']|_{w'}=N'_j} - - -] \neq C_2[- - - \overbrace{- - \underbrace{* * * * *}_{\text{discrepancy at } w} - -}^{C_2[\tilde{N}']|_{w'}=C[\tilde{N}']} - - -]$$

(a) The first case.

$$C_1[- - - \overbrace{- - \underbrace{* * * * *}_{\text{discrepancy at } w} - -}^{C_1[\tilde{N}']|_{w'}=N'_j} - - -] \neq C_2[- - \overbrace{- - - \overbrace{- - \underbrace{* * * * *}_{\text{discrepancy at } w} - -}^{C_2[\tilde{N}']|_{w'} \in st(N'_i)}}^{C_2[\tilde{N}']|_{w''}=N'_i} - -]$$

(b) The second case.

Lemma 2. *Let φ and φ' be two frames with $\varphi \approx'_{ss} \varphi'$, $ecores(\varphi) = (N)_{j \in J}$ and $ecores(\varphi') = (N')_{j \in J}$. It then holds for any contexts $C_1[\tilde{y}]$ and $C_2[\tilde{y}]$ that*

$$C_1[\tilde{N}] = C_2[\tilde{N}] \Leftrightarrow C_1[\tilde{N}'] = C_2[\tilde{N}']$$

Proof. We show the direction from left to right; the converse is symmetric. Suppose that $C_1[\tilde{N}] = C_2[\tilde{N}]$ and suppose for a contradiction that $C_1[\tilde{N}'] \neq C_2[\tilde{N}']$. Then the two contexts must differ at some position, so let w be a position of greatest length such that $C_1[\tilde{N}']|_w \neq C_2[\tilde{N}']|_w$. Now there must be a prefix w' of w and an ecore N'_j such that either $C_1[\tilde{N}']|_{w'} = N'_j$ or $C_2[\tilde{N}']|_{w'} = N'_j$; for otherwise $C_1[\tilde{N}] \neq C_2[\tilde{N}]$, contrary to assumption. Suppose the former without loss of generality, i.e. N'_j is an ecore in the first context which captures the discrepancy between the LHS and the RHS. There are now two cases to consider – each is illustrated in Figure 4.1 and covered below.

1. There is some context $C[\tilde{y}]$ such that $C_2[\tilde{y}]|_{w'} = C[\tilde{y}]$. Since $C_1[\tilde{N}']|_{w'} \neq C_2[\tilde{N}']|_{w'}$ per choice of w' we then have that $N'_j \neq C[\tilde{N}']$. Condition 2 in the definition of $\approx'_{ss}$ gives that also $N_j \neq C[\tilde{N}]$, i.e. $C_1[\tilde{N}]|_{w'} \neq C_2[\tilde{N}]|_{w'}$. Hence $C_1[\tilde{N}] \neq C_2[\tilde{N}]$, giving the desired contradiction.

2. The position w' does not exist in $C_2[\tilde{y}]$, i.e. $C_2[\tilde{N}']|_{w'}$ is a subterm of some core N'_i in $C_2[\tilde{N}']$. Then there is some prefix w'' of w and context $C[\tilde{y}]$ such that $N'_i = C_2[\tilde{N}']|_{w''}$. Also, there is some context $C^*_1[\tilde{y}]$ such that $C_1[\tilde{N}']|_{w''} = C^*_1[\tilde{N}']$. We have that $C_1[\tilde{N}']|_{w''} \neq C_2[\tilde{N}']|_{w''}$ and hence that $C^*_1[\tilde{N}'] \neq N'_i$. The contradiction is then obtained in the same manner as for case 1.

□

Next we turn our attention to showing that arbitrary rewrites are preserved by substitution of ecores. This is accomplished in three steps:

1. Show that *primitive* rewrites on any *partitioning context* is preserved by substitution of ecores. Condition 3 in $\approx'_{ss}$ gives that this result holds for

any partitioning contexts which are *not* more specific than the LHS of some rewrite rule. We rely on condition 2 to show that substitution of ecores also works for partitioning contexts which *are* more specific than the LHS of some rewrite rule.

2. Show that *primitive* rewrites on *any* context (i.e. not necessarily partitioning) are preserved by substitution of ecores. For this we rely on the notion of generated partitioning contexts.

3. Show that *any* rewrites (i.e. not necessarily primitive) on *any* context are preserved by substitution of ecores.

Each of the three steps is covered by a separate lemma.

Lemma 3. *Let φ and φ' be two frames with $\varphi \approx'_{ss} \varphi'$, $ecores(\varphi) = (N)_{j\in J}$ and $ecores(\varphi') = (N')_{j\in J}$. It then holds for any partitioning context $C_1^{\perp}[\tilde{y}]$ and any context $C_2^{\perp}[\tilde{y}]$ that:*

$$C_1^{\perp}[\tilde{N}] >_r C_2^{\perp}[\tilde{N}] \Leftrightarrow C_1^{\perp}[\tilde{N}'] >_r C_2^{\perp}[\tilde{N}']$$

Proof. We show the direction from left to right; the converse is symmetric. Suppose that $C_1^{\perp}[\tilde{N}] >_r C_2^{\perp}[\tilde{N}]$ by virtue of rule $r : L[\tilde{z}] >_r R[\tilde{z}]$. If $C_1^{\perp}[\tilde{N}] \sqsupset L[\tilde{z}]$ or $C_1^{\perp}[\tilde{N}] \Box L[\tilde{z}]$ the result follows immediately from condition 3 in $\approx'_{ss}$, so suppose that $C_1^{\perp}[\tilde{N}] \sqsubset L[\tilde{z}]$

Per definition of primitive rewrites and the assumption that $C_1^{\perp}[\tilde{y}]$ is less general than $L[\tilde{z}]$, there exists a substitution $\theta = \{{}^{T_i}/_{z_i}\}_{i\in I}$ such that $C_1^{\perp}[\tilde{N}] = L[\tilde{z}]\theta$ and $C_2^{\perp}[\tilde{N}] = R[\tilde{z}]\theta$. It must be shown that there is some substitution θ' such that $C_1^{\perp}[\tilde{N}'] = L[\tilde{z}]\theta'$ and $C_2^{\perp}[\tilde{N}'] = R[\tilde{z}]\theta'$. Say that there are k occurrences of z_i in $L[\tilde{z}]$. Then since each position w_j of an occurrence of z_i (i.e. with $L[\tilde{z}]|_{w_j} = z_i$) also exists in $C_1^{\perp}[\tilde{y}]$, the subterm $C_1^{\perp}[\tilde{N}]|_{w_j}$ is in fact a context over ecores. Hence we have that

$$T_i = C_1^{\perp}[\tilde{N}]|_{w_1} =, \ldots, = C_1^{\perp}[\tilde{N}]|_{w_k}$$

and it follows by Lemma 2 that also

$$T'_i = C_1^{\perp}[\tilde{N}']|_{w_1} =, \ldots, = C_1^{\perp}[\tilde{N}']|_{w_k}$$

Hence $C_1^{\perp}[\tilde{N}']$ and $C_2^{\perp}[\tilde{N}']$ *are* instances of $L[\tilde{z}]$ and $R[\tilde{z}]$ respectively, with unifying substitution $\theta' = \{{}^{T'_i}/_{z_i}\}_{i\in I}$. ☐

Lemma 4. *Let φ and φ' be two frames with $\varphi \approx'_{ss} \varphi'$, $ecores(\varphi) = (N)_{j\in J}$ and $ecores(\varphi') = (N')_{j\in J}$. It then holds for any contexts $C_1[\tilde{y}]$ and $C_2[\tilde{y}]$ that*

$$C_1[\tilde{N}] >_r C_2[\tilde{N}] \Leftrightarrow C_1[\tilde{N}'] >_r C_2[\tilde{N}']$$

Proof. We show the direction from left to right; the converse is symmetric. Suppose that $C_1[\tilde{N}] >_r C_2[\tilde{N}]$ by virtue of some rewrite rule $L[\tilde{z}] >_r R[\tilde{z}]$, i.e. $C_1[\tilde{N}]$ is an instance of $L[\tilde{z}]$. This also means that $C_1[\tilde{y}]$ and $L[\tilde{z}]$ are unifiable with some unifying substitution $\theta = \{^{T_a}/_{z_a}\}_{a\in A}\{^{S_b}/_{y_b}\}_{b\in B}$. $C_1[\tilde{N}]$ satisfies the constraints in θ in the sense that if e.g. $\theta = \{^{f(y_1)}/_{y_2}\}$ then it must hold that $N_2 = f(N_1)$. In general, the substitution θ can be viewed as a set of constraints which must be satisfied in order for any instance of $C_1[\tilde{y}]$ to be reducible.

Let $\tilde{y}_1, \ldots, \tilde{y}_k$ be the partition into equivalence classes of $\tilde{y}$ with respect to $\rightleftharpoons$. The key observation is that whenever $C_1[\tilde{N}]$ is an instance of $L[\tilde{z}]$ it also holds for every generated partitioning context $C_i^\perp[\tilde{y}_i]$ that $C_i^\perp[\tilde{N}_i]$ is an instance of $L[\tilde{z}]$. To see this, let θ_i be the unifying substitution for $C_i^\perp[\tilde{y}]$ and $L[\tilde{y}]$ (a such exists per definition of partitioning contexts). Suppose for a contradiction that $C_i^\perp[\tilde{N}]$ does not satisfy the constraints in θ_i. Any variable $z \in \tilde{z}$ which does not join correlated variables unifies trivially with some $\perp$ per construction of partitioning contexts. So the only constraint which can fail is a constraint on some of the ecores in $C_i^\perp[\tilde{N}]$. But these constraints are inherited from θ and hence also fail for $C_1[\tilde{N}]$. This contradicts the assumption that $C_1[\tilde{N}]$ is an instance of $L[\tilde{]}$.

Now suppose for a contradiction that that $C_1[\tilde{N}'] \not>_r C_2[\tilde{N}']$, i.e. some constraints in θ fail for this instantiation. Then there must be a generated partitioning context $C_i^\perp[\tilde{y}]$ such that $C_i^\perp[\tilde{N}']$ fails some of the same constraints in θ, i.e. $C_i^\perp[\tilde{N}'] \not>_r$. By Lemma 3 it also holds that $C_i^\perp[\tilde{N}] \not>_r$, contrary to the observation above.

We conclude that also $C_1[\tilde{N}'] >_r C_2[\tilde{N}']$ as desired. ☐

Lemma 5. *Let φ and φ' be two frames with $\varphi \approx'_{ss} \varphi'$, ecores$(\varphi) = (N)_{j\in J}$ and ecores$(\varphi') = (N')_{j\in J}$. It then holds for any contexts $C_1[\tilde{y}]$ and $C_2[\tilde{y}]$ that*

$$C_1[\tilde{N}] > C_2[\tilde{N}] \Leftrightarrow C_1[\tilde{N}'] > C_2[\tilde{N}']$$

Proof. We show the direction from the left to right; the converse follows by symmetry. So suppose that $C_1[\tilde{N}] > C_2[\tilde{N}]$. Per definition of the rewrite relation, there is some subterm $C_1[\tilde{N}]|_w$ which matches the redex of a rewrite rule r and therefore rewrites primitively: $C_1[\tilde{N}]|_w >_r C_1[\tilde{N}]|_{w_2}$ (note that the redex must also be a context over ecores since we are working in subterm theories). Then $C_2[\tilde{N}] = C_1[\tilde{N}]\{^{C_1[\tilde{N}]|_{w_2}}/_{C_1[\tilde{N}]|_w}\}$. By Lemma 4 $C_1[\tilde{N}']|_w >_r C_1[\tilde{N}']|_{w_2}$ and by Lemma 2 also $C_2[\tilde{N}'] = C_1[\tilde{N}']\{^{C_1[\tilde{N}']|_{w_2}}/_{C_1[\tilde{N}']|_w}\}$. We then conclude, per definition of the reduction relation, that $C_1[\tilde{N}'] > C_2[\tilde{N}']$. ☐

We are now ready to state the main lemma of this section, namely that equality is preserved by substitution of ecores from statically equivalent frames.

Lemma 6. *Let φ and φ' be two frames with $\varphi \approx'_{ss} \varphi'$, ecores$(\varphi) = (N)_{j\in J}$ and ecores$(\varphi') = (N')_{j\in J}$. It then holds for any contexts $C_1[\tilde{y}]$ and $C_2[\tilde{y}]$ that*

$$C_1[\tilde{N}] =_{\mathcal{E}} C_2[\tilde{N}] \Leftrightarrow C_1[\tilde{N}'] =_{\mathcal{E}} C_2[\tilde{N}']$$

Proof. We show the implication from left to right (the converse then follows by symmetry), so suppose that $C_1[\tilde{N}] =_{\mathcal{E}} C_2[\tilde{N}]$. Since the relation $=_{\mathcal{E}}$ coincides with $<>^*$ (the reflexive, symmetric and transitive closure of the rewrite relation), we also have that $C_1[\tilde{N}] <>^* C_2[\tilde{N}]$. Since we are working with convergent theories there is a $C_3[\tilde{N}] = C_1[\tilde{N}] \downarrow = C_2[\tilde{N}] \downarrow$. Hence it holds that $C_1[\tilde{N}] >^{k_1} C_3[\tilde{N}]$ and $C_2[\tilde{N}] >^{k_2} C_3[\tilde{N}]$ for some $k_1, k_2 \in \mathbb{N}$. We will show that these reduction sequences are preserved by substitution of ecores from φ', i.e. that also $C_1[\tilde{N'}] >^{k_1} C_3[\tilde{N'}]$ and $C_2[\tilde{N'}] >^{k_2} C_3[\tilde{N'}]$.

We give a proof of the first case by induction in k_1. **Basis ($k_1 = 0$).** Then we have that $C_1[\tilde{N}] = C_3[\tilde{N}]$, and it follows by Lemma 2 that also $C_1[\tilde{N'}] = C_3[\tilde{N'}]$. **Step (assume for k_1, prove for $k_1 + 1$).** In this case we have that $C_1[\tilde{N}] > M >^{k_1} C_3[\tilde{N}]$ for some M. The crucial point is that M can be written as a context over ecores since it is a subterm of a context over ecores, i.e. $M = C_4[\tilde{N}]$ for some $C_4[\tilde{y}]$. So we have that $C_1[\tilde{N}] > C_4[\tilde{N}] >^{k_1} C_3[\tilde{N}]$. By Lemma 5 it also holds that $C_1[\tilde{N'}] > C_4[\tilde{N'}]$. The induction hypothesis gives that $C_4[\tilde{N'}] >^{k_1} C_3[\tilde{N'}]$. Hence $C_1[\tilde{N'}] >^{k_1+1} C_3[\tilde{N'}]$ as desired.

The proof that $C_2[\tilde{N'}] >^{k_2} C_3[\tilde{N'}]$ is identical. It follows immediately that $C_1[\tilde{N'}] \downarrow = C_2[\tilde{N'}] \downarrow$ and by assumption of convergent subterm theories that $C_1[\tilde{N'}] <>^* C_2[\tilde{N'}]$. Hence also $C_1[\tilde{N'}] =_{\mathcal{E}} C_2[\tilde{N'}]$, which concludes the proof. $\square$

The above proof relies crucially on confluence for working on $>^*$ instead of on $<>^*$. It would be impossible to carry out the induction on reduction sequences on the form $C_1[\tilde{N}] <>^* C_2[\tilde{N}]$. For suppose the first step in this sequence is $C_1[\tilde{N}] < M <>^* C_2[\tilde{N}]$. Then it would not necessarily be possible to write M as a context over ecores; for example we might have that $a < \mathit{fst}[a, b] <>^* \mathit{snd}[b, a]$ where b is private in φ and where $\mathit{fst}[a, b]$ therefore cannot be written as any context over ecores. Consequently neither Lemma 2 nor the induction hypothesis can be applied.

One might also attempt to give an alternative proof directly from the definition of $=_{\mathcal{E}}$; this would go by induction in the height of the proof tree of the equality $C_1[\tilde{N}] =_{\mathcal{E}} C_2[\tilde{N}]$. However this approach presents some difficulties. First of all, there is a universal quantification over substitutions in the SUBSTITUTION rule. This can be overcome by replacing the SUBSTITUTION and the REWRITE rule with the following new rule:

$$(\textsc{Primitive Rewrite})\frac{}{M_1 =_{\mathcal{E}} M_2} \text{ if } M_1 >_r M_2$$

It is easy to prove that the equational theory induced based on this new rule is the same as the theory induced by the standard definition. Intuitively, instead of instantiating variables at arbitrary positions in the proof of an equality, the variable could just as well have been instantiated at time of introduction.

However one problem with an induction in the height of the proof tree of $C_1[\tilde{N}] =_{\mathcal{E}} C_2[\tilde{N}]$ remains, namely the transitivity case. The premises would give that $C_1[\tilde{N}] =_{\mathcal{E}} M$ and $M =_{\mathcal{E}} C_2[\tilde{N}]$, but it is not necessarily possible to write M as a context over ecores and thus the induction hypothesis does not apply. There does not seem to be any easy solution to this problem, so a direct proof on $=_{\mathcal{E}}$ fails for this lemma.

We now arrive at the main result of this section:

Theorem 1. *$\approx'_{ss}$ implies $\approx_{ss}$.*

Proof. Let $\varphi \equiv (\nu\tilde{n})\{{}^{M_i}/_{x_i}\}_{i\in I}$ and $\varphi' \equiv (\nu\tilde{n})\{{}^{M'_i}/_{x_i}\}_{i\in I}$ be two frames with $\varphi \approx'_{ss} \varphi'$ and let $ecores(\varphi) = (N)_{j\in J}$ and $ecores(\varphi') = (N')_{j\in J}$. We must show that the two conditions in $\approx_{ss}$ both hold.

Condition 1 Let M_1 and M_2 be any two terms. We must show that $(M_1 =_{\mathcal{E}} M_2)\varphi \Leftrightarrow (M_1 =_{\mathcal{E}} M_2)\varphi'$; we show the implication from left to right, as the converse is symmetric. So suppose that $(M_1 =_{\mathcal{E}} M_2)\varphi$, i.e. $n(M_1, M_2) \cap bn(\varphi) = \emptyset$ and $M_1\varphi =_{\mathcal{E}} M_2\varphi$. This means that M_1 and M_2 are in fact context over variables from φ and some free names $\tilde{n}$, so we reason as follows:

$$\begin{array}{rcll} M_1\varphi & =_{\mathcal{E}} & M_2\varphi & \\ C_{M_1}[x_{a_1}, \ldots, x_{a_s}, \tilde{n}]\varphi & =_{\mathcal{E}} & C_{M_2}[x_{b_1}, \cdots, x_{b_t}, \tilde{n}]\varphi & \Updownarrow \\ C_{M_1}[x_{a_1}\varphi, \ldots, x_{a_s}\varphi, \tilde{n}] & =_{\mathcal{E}} & C_{M_2}[x_{b_1}\varphi, \cdots, x_{b_t}\varphi, \tilde{n}] & \Updownarrow \\ C_{M_1}[M_{a_1}, \ldots, M_{a_s}, \tilde{n}] & =_{\mathcal{E}} & C_{M_2}[M_{b_1}, \ldots, M_{b_t}, \tilde{n}] & \end{array}$$

Condition 1 in the definition of $\approx'_{ss}$ says that each $M \in im(\varphi)$ can be written as a context over ecores, and since each free name in φ is also an ecore we have that $\tilde{n} = \tilde{N}$ and hence:

$$C_{M_1}[C_{a_1}[\tilde{N}], \ldots, C_{a_s}[\tilde{N}], \tilde{N}] =_{\mathcal{E}} C_{M_2}[C_{b_1}[\tilde{N}], \cdots, C_{b_t}[\tilde{N}], \tilde{N}]$$

Next apply Lemma 6:

$$C_{M_1}[C_{a_1}[\tilde{N'}], \ldots, C_{a_s}[\tilde{N'}], \tilde{N'}] =_{\mathcal{E}} C_{M_2}[C_{b_1}[\tilde{N'}], \cdots, C_{b_t}[\tilde{N'}], \tilde{N'}]$$

By condition 1 of $\approx'_{ss}$ again, each context $C_{a_i}[\tilde{N'}]$ is syntactically equal to the corresponding term $M'_{a_i} \in im(\varphi')$. We also have that ecores which represent free names are identical in the two frames. Hence:

$$\begin{array}{rcll} C_{M_1}[M'_{a_1}, \ldots, M'_{a_s}, \tilde{n}] & =_{\mathcal{E}} & C_{M_2}[M'_{b_1}, \ldots, M'_{b_t}, \tilde{n}] & \\ C_{M_1}[x_{a_1}\varphi', \ldots, x_{a_s}\varphi', \tilde{n}] & =_{\mathcal{E}} & C_{M_2}[x_{b_1}\varphi', \cdots, x_{b_t}\varphi', \tilde{n}] & \Updownarrow \\ M_1\varphi' & =_{\mathcal{E}} & M_2\varphi' & \end{array}$$

Finally, since $bn(\varphi) = bn(\varphi')$ we get that $(M_1 =_{\mathcal{E}} M_2)\varphi'$ as desired, concluding the proof of condition 1 in $\approx_{ss}$.

Condition 2. Let M be any term. We must show that $(M >^k_\varphi) \Leftrightarrow (M >^k_{\varphi'})$; we do the implication from left to right as the converse follows by symmetry. So suppose that $(M >^k_\varphi)$, i.e. $n(M) \cap bn(\varphi) = \emptyset$ and there is some M^k such that $M\varphi >^k M^k$. In other words, there is a sequence of rewrites $M\varphi > M^1 > \cdots > M^k$. As in the proof of condition 1 it follows by the condition on names that M is in fact a context over terms from φ and free names, and we reason as follows:

$$\begin{aligned} M\varphi &= C_M[x_{a_1}, \cdots, x_{a_s}, \tilde{n}]\varphi \\ &= C_M[x_{a_1}\varphi, \cdots, x_{a_s}\varphi, \tilde{n}] \\ &= C_M[M_{a_1}, \cdots, M_{a_s}, \tilde{n}] \\ &= C_M[C_{a_1}[\tilde{N}], \cdots, C_{a_s}[\tilde{N}], \tilde{N}] \end{aligned}$$

Again the last step follows from condition 1 in the definition of $\approx'_{ss}$, which says that each $M \in im(\varphi)$ can be written as a context over ecores.

Since we are working in subterm theories, each of the reducts M^i are subterms of $M\varphi$ and hence can also be written as a context over ecores and free names. We then have the following sequence of rewrites:

$$C_M[C_{a_1}[\tilde{N}], \cdots, C_{a_s}[\tilde{N}], \tilde{N}] > C_1[\tilde{N}] > \cdots > C_k[\tilde{N}]$$

and by Lemma 5 the corresponding rewrites on ecores in φ' also hold:

$$C_M[C_{a_1}[\tilde{N}'], \cdots, C_{a_s}[\tilde{N}'], \tilde{N}'] > C_1[\tilde{N}'] > \cdots > C_k[\tilde{N}']$$

By condition 1 of $\approx'_{ss}$ again, each context $C_{a_i}[\tilde{N}']$ is syntactically equal to the corresponding term $M'_{a_i} \in im(\varphi')$, and the free names are the same in the two frames:

$$\begin{aligned} C_M[M'_{a_1}, \cdots, M'_{a_s}, \tilde{n}] &> C_1[\tilde{N}'] > \cdots > C_k[\tilde{N}'] \\ C_M[x_{a_1}, \cdots, x_{a_s}, \tilde{n}]\varphi' &> C_1[\tilde{N}'] > \cdots > C_k[\tilde{N}'] \qquad \Updownarrow \\ M\varphi' &> C_1[\tilde{N}'] > \cdots > C_k[\tilde{N}'] \end{aligned}$$

Hence $(M >^k_{\varphi'})$ as desired. $\square$

4.2 $\approx_{ss}$ implies $\approx'_{ss}$

The main theorem of this section states that $\approx_{ss}$ implies $\approx'_{ss}$. In order to prove this result a number of preliminary lemmas are again required. One of the key lemmas states that if $\varphi \approx_{ss} \varphi'$ then for each $N_j \in ecores(\varphi)$ there is a context $\mathcal{R}[\tilde{x}]$ over terms from $im(\varphi)$ such that $\mathcal{R}[\tilde{M}] >^k N_j$ and $\mathcal{R}[\tilde{M}'] >^k N'_j$. The context $\mathcal{R}[\tilde{M}]$ is coined the *recipe* of N_j since in records the details of how the ecore N_j is revealed from terms in φ.

To make these ideas precise we start by giving a definition of recipes.

Definition 24 (Analysis recipe). *Let φ be any frame and let $M|_w \in \mathcal{A}(M, \varphi)$ for some $M \in im(\varphi)$. The analysis recipe $\mathcal{R}[\tilde{x}]$ for $M|_w$ is a context which is defined inductively on the analysis level i as follows:*

- *$M|_w \in \mathcal{A}^0(M_i, \varphi)$. Then $M|_w = M$ and we define $\mathcal{R}[\tilde{x}]$ to be the trivial context x where $M = x\varphi$.*

- $M|_w \in \mathcal{A}^{i+1}(M_i, \varphi)$. *If also* $M|_w \in \mathcal{A}^i(M_i, \varphi)$ *then the analysis recipe is identical to the analysis recipe from level i. Otherwise, per definition of analysis, there is some* $M|_{w_2} \in \mathcal{A}^i(M, \varphi)$ *such that* $M|_{w_2} \succ_{\bigcup_{M \in S} \mathcal{A}^i(M,S)} > M|_w$ *(where* $S = im(\varphi) \cup fn(\varphi)$*), which per definition of revelation means that there are* $L_1, \ldots, L_k$ *each in* $\mathcal{A}^i(M, \varphi)$ *for some* $M \in im(\varphi)$*, and a destructor context* $D[x, x_1, \ldots, x_k]$*, such that* $D[M|_{w_2}, L_1, \ldots, L_k] > M|_w$*. Choose* $D^*[x, x_1, \ldots, x_k]$ *to be any such context for which all subterms are irreducible, i.e. only a primitive reduction is possible (such context will always exist!). Let* $\mathcal{R}[\tilde{x}]$ *be the analysis recipe for* $M|_{w_2}$ *and let* $\mathcal{R}_j[\tilde{x}]$ *be the analysis recipe for each* L_j *(these recipes are well-defined because* $M|_{w_2}$ *and the* L_j*s are in* $\mathcal{A}^i(M_i, \varphi)$*). Then* $D^*[\mathcal{R}[\tilde{x}], \mathcal{R}_1[\tilde{x}], \ldots, \mathcal{R}_k[\tilde{x}]]$ *is defined to be the analysis recipe for* $M|_w$*.*

Analysis recipes are not necessarily unique since there may be two analysis terms which are syntactically equal and hence both can be used for revelation in the analysis. But this will not be important for our purpose. The next lemma states that corresponding analysis terms from statically equivalent frames have the same analysis recipes.

Lemma 7. *Let* $\varphi \equiv (\nu\tilde{n})\{{}^{M_i}/_{x_i}\}_{i\in I}$*,* $\varphi' \equiv (\nu\tilde{n'})\{{}^{M'_i}/_{x_i}\}_{i\in I}$ *with* $\varphi \approx_{ss} \varphi'$*. If* $\mathcal{R}[\tilde{x}]$ *is an analysis recipe for* $M_i|_w \in \mathcal{A}(M_i, \varphi)$ *then it is also an analysis recipe for* $M'_i|_w \in \mathcal{A}(M'_i, \varphi')$ *and there is a* $k \in \mathbb{N}$ *such that*

$$\mathcal{R}[\tilde{M}] >^k M_i|_w \not>$$
$$\mathcal{R}[\tilde{M'}] >^k M'_i|_w \not>$$

Proof. By induction in the definition of analysis recipes.

Basis ($M_i|_w \in \mathcal{A}^0(M_i, \varphi)$**).** Trivial.

Step ($M_i|_w \in \mathcal{A}^{i+1}(M_i, \varphi)$**).** Per definition of analysis there is some $M_i|_{w_2} \in \mathcal{A}^i(M_i, \varphi)$ such that $M_i|_{w_2} \succ_{\bigcup_{T \in S} \mathcal{A}^i(T,S)} > M_i|_w$ (where $S = im(\varphi) \cup fn(\varphi)$), which per definition of revelation means that there are $L_1, \ldots, L_s$ each in $\mathcal{A}^i(M, \varphi)$ for some $M \in im(\varphi)$, and a destructor context $D[x, x_1, \ldots, x_s]$, such that $D[M_i|_{w_2}, L_1, \ldots, L_s] >^{M_i|_{w_2}} M_i|_w$.

Let $\mathcal{R}[\tilde{x}]$ be the analysis recipe for $M_i|_{w_2}$ and let $\mathcal{R}_j[\tilde{x}]$ be the analysis recipe for each L_j. Then $D[\mathcal{R}[\tilde{x}], \mathcal{R}_1[\tilde{x}], \ldots, \mathcal{R}_s[\tilde{x}]]$ is per definition an analysis recipe for $M_i|_w$.

Per induction hypothesis there is a $k \in \mathbb{N}$ such that $\mathcal{R}[\tilde{M}] >^k M_i|_{w_2}$ and $\mathcal{R}[\tilde{M'}] >^k M'|_{w_2}$. Similarly there is a $k_j \in \mathbb{N}$ such that $\mathcal{R}_j[\tilde{M}] >^{k_j} L_j$ and $\mathcal{R}_j[\tilde{M'}] >^{k_j} L'_j$ for each j. Hence we get that

$$D[\mathcal{R}[\tilde{M}], \mathcal{R}_1[\tilde{M}], \ldots, \mathcal{R}_s[\tilde{M}]] >^l D[M_i|_{w_2}, L_1, \ldots, L_s]$$
$$D[\mathcal{R}[\tilde{M'}], \mathcal{R}_1[\tilde{M'}], \ldots, \mathcal{R}_s[\tilde{M'}]] >^l D[M'_i|_{w_2}, L'_1, \ldots, L'_s]$$

where $l = k+k_1+\cdots+k_s$. Per definition of analysis recipes, exactly one further rewrite is possible for the former context $D[M_i|_{w_2}, L_1, \ldots, L_s]$, namely the primitive rewrite. By condition 2 in $\approx_{ss}$, exactly one rewrite is also possible for the latter context $D[M'_i|_{w_2}, L'_1, \ldots, L'_s]$, and per definition of analysis recipes this must be also be a primitive rewrite. Since $M_i|_{w_2}$ and $M'_i|_{w_2}$ are the revelators for the respective reductions and the reduct in the first case is $M_i|_w$, the reduct for the second case must correspondingly be $M_i|_w$.

Finally, the induction hypothesis gives that $\mathcal{R}[\tilde{x'}]$ is also the analysis recipe for $M'|_{w_2}$ and $\mathcal{R}_j[\tilde{M'}]$ is the analysis recipe for M'_j for each j. Hence the context

$$D[\mathcal{R}[\tilde{x}], \mathcal{R}_1[\tilde{x}], \ldots, \mathcal{R}_s[\tilde{x}]]$$

is per definition also the analysis recipe for $M'_i|_w$. This concludes the proof. $\square$

The above result could be re-phrased in terms of "revelation bisimulation"; informally, any two corresponding analysis terms from statically equivalent frames can exhibit the same revelations and the revealed terms are bisimilar (i.e. corresponding revelated terms can exhibit the same further revelations). We will not pursue this direction further since it is not necessary for our purposes, but the idea of revelation bisimulation does yield some insight into the nature of static equivalence.

The next lemma is an immediate corollary of Lemma 7:

Lemma 8. *Let φ and φ' be two frames with $\varphi \approx_{ss} \varphi'$, $ecores(\varphi) = (N)_{j\in J}$ and $ecores(\varphi) = (N')_{j\in J}$. Then for any contexts $C_1[\tilde{y}]$ and $C_2[\tilde{y}]$ it holds that*

$$C_1[\tilde{N}] =_{\mathcal{E}} C_2[\tilde{N}] \Leftrightarrow C_1[\tilde{N'}] =_{\mathcal{E}} C_2[\tilde{N'}]$$

Next we give two important results on the kind of contexts $C^{\perp}[\tilde{y}]$ considered in condition 3 of $\approx'_{ss}$. Informally, the first lemma says that every variable in $C^{\perp}[\tilde{y}]$ is strongly correlated to at least one variable at a position which exists in the relevant rewrite rule. This is used in the proof of the second lemma, which says that only primitive rewrites on $C^{\perp}[\tilde{N}]$ are possible.

Lemma 9. *Let φ and φ' be two frames with $\varphi \approx_{ss} \varphi'$, $ecores(\varphi) = (N)_{j\in J}$ and $ecores(\varphi) = (N')_{j\in J}$. Let $C_1[\tilde{y}]$ be any partitioning context with $C_1[\tilde{y}] \sqsupset L[\tilde{z}]$ or $C_1[\tilde{y}] \square L[\tilde{z}]$ for some rewrite rule $L[\tilde{z}] >_r R[\tilde{z}]$. Then any variable position w in $C[\tilde{y}]$ (i.e. a position such that $C[\tilde{y}]|_w$ is a variable) is strongly correlated to some variable position w' which exists in $L[\tilde{z}]$.*

Proof. Let w be any position in $C[\tilde{y}]$. There are two cases to consider.

1. w is weakly correlated to all other positions w' in $C[\tilde{y}]$ only through a single variable z in $v(L[\tilde{z}])$, i.e. $w \rightleftarrows_z^k w'$ for some natural number k. Hence also $w \rightleftarrows w'$, i.e. w and w' are also strongly correlated. By the assumption that $C_1[\tilde{y}] \sqsupset L[\tilde{z}]$ or $C_1[\tilde{y}] \square L[\tilde{z}]$, at least one of the positions w' exist in $L[\tilde{z}]$, which concludes the first case.

2. w is weakly correlated to some w' through at least two variables z_1 and z_2 in $v(L[\tilde{z}])$. In this case we have for some w'' that $w \rightleftarrows^k_{z_1} w'' \rightleftarrows_{z_2} w'$. Again this implies that $w \rightleftarrows w'$. Since w'' correlates through both z_1 and z_2 this position must exist in $L[\tilde{z}]$, which concludes the second case.

□

Lemma 10. *Let φ be any frame with ecores$(\varphi) = (N)_{j \in J}$. It then holds for any partitioning context $C_1^{\perp}[\tilde{y}]$ with $C_1^{\perp}[\tilde{y}] \sqsupset$ or $C_1^{\perp}[\tilde{y}]\Box$ and any context $C_2^{\perp}[\tilde{y}]$ that*

$$C_1^{\perp}[\tilde{N}] > C_2^{\perp}[\tilde{N}] \Leftrightarrow C_1^{\perp}[\tilde{N}] >_r C_2^{\perp}[\tilde{N}]$$

Proof. Let $L[\tilde{z}] >_r R[\tilde{z}]$ be the rewrite rule such that $C_1[\tilde{y}] \sqsupset L[\tilde{z}]$ or $C_1[\tilde{y}]\Box L[\tilde{z}]$. The direction from right to left is immediate from the definition of the rewrite relation. For the converse first observe that any subterm $C_1[\tilde{N}]|_w$ at a position w which exists in $L[\tilde{z}]$ is irreducible by the assumption of independent rewrite systems (i.e. no destructor functions occur internally in rewrite rules). Consequently any possible internal reduction must be at position w which exist in $C_1[\tilde{y}]$ but not in $L[\tilde{z}]$. But by Lemma 9 the position w is strongly correlated to a position w' which exists in $L[\tilde{z}]$. Therefore $C_1[\tilde{N}]|_w$ must be a subterm of some ecore and is hence irreducible. □

We are now ready to show that $\approx_{ss}$ implies condition 3 in $\approx'_{ss}$.

Lemma 11. *Let $\varphi \equiv (\nu\tilde{n})\{^{M_i}/_{x_i}\}_{i \in I}$, $\varphi' \equiv (\nu\tilde{n'})\{^{M'_i}/_{x_i}\}_{i \in I}$, ecores$(\varphi) = (N)_{j \in J}$, ecores$(\varphi) = (N')_{j \in J}$ and $\varphi \approx_{ss} \varphi'$. Let $C_1^{\perp}[\tilde{y}]$ be a partitioning context with $C_1^{\perp}[\tilde{y}] \sqsupset$ or $C_1[\tilde{y}]\Box$. It then holds for any context $C_2^{\perp}[\tilde{y}]$ that*

$$C_1^{\perp}[\tilde{N}] >_r C_2^{\perp}[\tilde{N}] \Leftrightarrow C_1^{\perp}[\tilde{N}'] >_r C_2^{\perp}[\tilde{N}']$$

Proof. We show the direction from left to right; the converse is symmetric. Suppose that the rewrite $C_1^{\perp}[\tilde{N}] >_r C_2^{\perp}[\tilde{N}]$ is possible by virtue of some rule $L[\tilde{z}] >_r R[\tilde{z}]$. We then have that $C_1^{\perp}[\tilde{N}] =_{\mathcal{E}} C_2^{\perp}[\tilde{N}]$ and by Lemma 8 also $C_1^{\perp}[\tilde{N}'] =_{\mathcal{E}} C_2^{\perp}[\tilde{N}']$. The challenge is to show that the equality is in fact a primitive rewrite, which is not immediately apparent. For it may be that the equality is concluded by transitivity, i.e. multiple reduction steps are necessary. Such multiple rewrites may also be *possible* in $C_1[\tilde{N}]$ even though they are not *necessary* for establishing the equality, so we cannot use this to give a proof by contradiction. We must take another approach.

Each ecore N_j and N'_j is an analysis term so by Lemma 7 there is an analysis recipe $\mathcal{R}_j[\tilde{x}]$ such that $\mathcal{R}_j[\tilde{M}] >^{k_j} N_j$ and $\mathcal{R}_j[\tilde{M}'] >^{k_j} N'_j$. As in the proof of Lemma 7 we get that there is a context $C^{\perp}[\tilde{x}]$ such that $C^{\perp}[\tilde{M}] >^l C_1^{\perp}[\tilde{N}]$ and $C^{\perp}[\tilde{M}'] >^l C_1^{\perp}[\tilde{N}']$. $C_2^{\perp}[\tilde{N}]$ is irreducible by Lemma 10, so $C^{\perp}[\tilde{M}]$ normalises in $l+1$ reduction steps. By condition 2 in $\approx_{ss}$ exactly one further rewrite is possible for $C^{\perp}[\tilde{N}']$, for otherwise $C^{\perp}[\tilde{M}']$ would have a different number of reductions than $C^{\perp}[\tilde{M}]$. Finally we have by Lemma 10 that this must be a primitive rewrite.

Technically one must require in the above argument that the reduction $C^{\perp}[\tilde{M}] >^{l} C_1^{\perp}[\tilde{N}]$ in l steps is *maximal*, i.e. that there is no longer reduction sequence yielding the same reduct. Any maximal reduction must be a *bottom-up* reduction in the sense that subterms are always reduced first. It follows that any maximal reduction of $C^{\perp}[\tilde{M}]$ must be through $C_1^{\perp}[\tilde{N}]$ and correspondingly that any maximal reduction of $C^{\perp}[\tilde{M'}]$ must be through $C_1^{\perp}[\tilde{N'}]$, which is necessary for the soundness of the proof. $\square$

The next lemma states that $\approx_{ss}$ implies condition 2 in $\approx'_{ss}$.

Lemma 12. *Let* $\varphi \equiv (\nu\tilde{n})\{{}^{M_i}/_{x_i}\}_{i\in I}$, $\varphi' \equiv (\nu\tilde{n'})\{{}^{M'_i}/_{x_i}\}_{i\in I}$, $ecores(\varphi) = (N)_{j\in J}$, $ecores(\varphi) = (N')_{j\in J}$ *and* $\varphi \approx_{ss} \varphi'$. *It then holds for any context* $C[\tilde{y}]$ *and any* $j \in J$ *that*

$$C[\tilde{N}] = N_j \Leftrightarrow C[\tilde{N'}] = N'_j$$

Proof. Suppose that $C[N_1, \ldots, N_s] = N_j$. Each N_i has a recipe $\mathcal{R}_i[\tilde{x}]$, i.e. $\mathcal{R}_i[\tilde{M}] >^{k_i} N_i$ where k_i maximal. Hence

$$C[\mathcal{R}_1[\tilde{M}], \ldots, \mathcal{R}_s[\tilde{M}]] =_{\mathcal{E}} \mathcal{R}_j[\tilde{M}]$$

and by condition 1 in $\approx_{ss}$ also

$$C[\mathcal{R}_1[\tilde{M'}], \ldots, \mathcal{R}_s[\tilde{M'}]] =_{\mathcal{E}} \mathcal{R}_j[\tilde{M'}]$$

By Lemma 7 the recipe for each N'_i is $\mathcal{R}_i[\tilde{x}]$, so $\mathcal{R}_i[\tilde{M'}] >^{k_i} N_i$ where k_i maximal and we have that

$$C[N'_1, \ldots, N'_k] =_{\mathcal{E}} N'_j$$

The right hand side, N'_j, is irreducible since it is an ecore. We then propose a **Claim:** The left hand side is also irreducible. Given this claim, the equality is syntactic by confluence, which concludes the proof.

Proof of claim. Suppose for a contradiction that the left hand side, $C[N'_1, \ldots, N'_s]$, is not irreducible, i.e. we have for some M' that:

$$C[\mathcal{R}_1[\tilde{M'}], \ldots, \mathcal{R}_s[\tilde{M'}]] >^{l} C[N'_1, \ldots, N'_k] > M'$$

where $l = k_1 + \ldots k_s$ is maximal. Condition 2 in $\approx_{ss}$ then gives that also

$$C[\mathcal{R}_1[\tilde{M}], \ldots, \mathcal{R}_s[\tilde{M}]] >^{l} C[N_1, \ldots, N_k] > M$$

for some M, contradicting that $C[N_1, \ldots, N_k]$ is syntactically equal to an ecore (and hence irreducible). $\square$

Finally we arrive at the main result of this section.

Theorem 2. $\approx_{ss}$ *implies* $\approx'_{ss}$

Proof. Let $\varphi \equiv (\nu\tilde{n})\{^{M_i}/_{x_i}\}_{i\in I}$, $\varphi' \equiv (\nu\tilde{n'})\{^{M'_i}/_{x_i}\}_{i\in I}$, $ecores(\varphi) = (N)_{j\in J}$, $ecores(\varphi) = (N')_{j\in J}$ and $\varphi \approx_{ss} \varphi'$. We must show that conditions 1, 2 and 3 in $\approx'_{ss}$ hold.

Condition 1. By Lemma 1 any $M_i \in im(\varphi)$ can be written as a context over ecores, i.e. at least one context $C_i[\tilde{y}]$ exists such that $M_i = C_i[\tilde{N}]$. Then it must also hold that $M'_i = C_i[\tilde{N}']$; for otherwise there would be ecores N_j and N'_j in M_i and M'_i respectively such that N_j and N'_j do not have the same analysis recipes, contradicting Lemma 7.

Condition 2. This is Lemma 12.

Condition 3. This is Lemma 11. □

4.3 Conditions under which $\approx_s$ and $\approx_{ss}$ Coincide

In Chapter 2 we demonstrated that $\approx_s$ and $\approx_{ss}$ do not generally coincide and gave the following counter-example in the theory of public key encryption:

$$\begin{aligned}\varphi &\triangleq (\nu a, c, k)\{^{enc(a,k^+)}/_{x_1}, {}^{k^-}/_{x_2}\}\\ \varphi' &\triangleq (\nu a, c, k)\{^{enc(a,k^+)}/_{x_1}, {}^{c^-}/_{x_2}\}\end{aligned}$$

It is easy to verify that $\varphi \approx_s \varphi'$ while $\varphi \not\approx_{ss} \varphi'$. Hence allowing tests on reductions as well as tests on equality increases the distinguishing power of an observer process. However, there are theories where tests on reductions do not yield increased distinguishing power. Take for example the symmetric-key counterpart of the above frames:

$$\begin{aligned}\varphi_2 &\triangleq (\nu a, c, k)\{^{enc(a,k)}/_{x_1}, {}^{k}/_{x_2}\}\\ \varphi'_2 &\triangleq (\nu a, c, k)\{^{enc(a,k)}/_{x_1}, {}^{c}/_{x_2}\}\end{aligned}$$

We still have that $\varphi_2 \not\approx_{ss} \varphi'_2$ because $dec(M_1, M_2) >$ while $dec(M'_1, M'_2) \not>$, so condition 2 in $\approx_{ss}$ fails. But suddenly we also have that $\varphi_2 \not\approx_s \varphi'_2$ because the following equality holds in φ_2 but not in φ'_2:

$$x_1 =_{\mathcal{E}} enc(dec(x_1, x_2), x_2)$$

Hence in this case the standard static equivalence agrees with the strong version. The reason for this is intuitively that, for the above equality to hold in φ_2, a reduction on $dec(M_1, M_2)$ is forced. For if no reduction is possible, and since M_1 is irreducible, we would have by confluence that

$$M_1 = enc(dec(M_1, M_2), M_2)$$

This syntactic equality is impossible since M_1 cannot be a proper subterm of itself.

The question now is in which situations a reduction can be forced by an appropriate equality as in the example above – and thus where condition 2 in $\approx_{ss}$ is superfluous. The equality in the above example can be constructed because the term $enc(a, k)$ in φ_2 can be written as a *non-trivial* context over other analysis terms. In contrast it was not possible to write $enc(a, k^+)$ in φ as a nontrivial context over analysis terms since k^+ is not in the analysis (we have that $enc(a, k^+)$ is an ecore but not a core).

More generally, any analysis term L which is not an ecore can be written as a non-trivial context over other analysis terms exactly when $cores(\varphi) = ecores(\varphi)$. For otherwise L would per definition be an ecore, which is impossible since ecores are per assumption cores (and L was assumed not to be a core).

We now have the intuition to give a formal proof of the main result in this section, namely that $\approx_{ss}$ implies $\approx'_{ss}$ whenever $cores(\varphi) = ecores(\varphi)$. We start with a preliminary lemma, namely the counterpart of Lemma 7 which says that corresponding terms from statically equivalent frames have the same analysis recipes. We cannot rely on the original proof of this lemma since it assumed $\approx_{ss}$ and relied on condition 2. Hence we employ the idea set forth in the above example that equalities can force rewrites.

Lemma 13 (cf. Lemma 7). *Let $\varphi \equiv (\nu\tilde{n})\{^{M_i}/_{x_i}\}_{i\in I}$, $\varphi' \equiv (\nu\tilde{n'})\{^{M'_i}/_{x_i}\}_{i\in I}$ with $\varphi \approx_s \varphi'$ and suppose that $cores(\varphi) = ecores(\varphi)$. If $\mathcal{R}[\tilde{x}]$ is an analysis recipe for $M_i|_w \in \mathcal{A}(M_i, \varphi)$ then it is also an analysis recipe for $M'_i|_w \in \mathcal{A}(M'_i, \varphi')$ and there is a $k \in \mathbb{N}$ such that*

$$\mathcal{R}[\tilde{M}] >^k M_i|_w \not>$$
$$\mathcal{R}[\tilde{M'}] >^k M'_i|_w \not>$$

Proof. By induction in the definition of analysis recipes.

Basis ($M_i|_w \in \mathcal{A}^0(M_i, \varphi)$). Trivial.

Step ($M_i|_w \in \mathcal{A}^{i+1}(M_i, \varphi)$). We shall benefit from some notational short hands: for any tuple $\tilde{L} \subseteq \mathcal{A}(\varphi)$ where $\tilde{L} = L_1, \ldots, L_s$ and $\mathcal{R}_1[\tilde{x}], \ldots, \mathcal{R}_s[\tilde{x}]$ are the respective analysis recipes, we let $\tilde{\mathcal{R}}_L[\tilde{x}] \triangleq \mathcal{R}_1[\tilde{x}], \ldots, \mathcal{R}_s[\tilde{x}]$.

Now per definition of analysis there is some $M_i|_{w_2} \in \mathcal{A}^i(M_i, \varphi)$ such that $M_i|_{w_2} \succ_{\bigcup_{M\in S} \mathcal{A}^i(M,S)} M_i|_w$ (where $S = im(\varphi) \cup fn(\varphi)$), which per definition of revelation means that there are terms $L_1, \ldots, L_s \subseteq \bigcup_{M\in S} \mathcal{A}^i(M, S)$ and a destructor context $D[x, x_1, \ldots,]$ such that $D[M_i|_{w_2}, \tilde{L}] >^{M_i|_{w_2}} M_i|_w$. Let $\mathcal{R}[\tilde{x}]$ be the analysis recipe for $M_i|_{w_2}$ and let $\tilde{\mathcal{R}}_L[\tilde{x}]$ be the tuple of analysis recipes for $\tilde{L}$ as defined above. Then $D[\mathcal{R}[\tilde{x}], \tilde{\mathcal{R}}_L[\tilde{x}]]$ is per definition an analysis recipe for $M_i|_w$.

Per induction hypothesis there is a k such that $\mathcal{R}[\tilde{M}] >^k M_i|_{w_2}$ and $\mathcal{R}[\tilde{M'}] >^k M'|_{w_2}$. Similarly there is a k_j such that $\mathcal{R}_j[\tilde{M}] >^{k_j} L_j$ and $\mathcal{R}_j[\tilde{M'}] >^{k_j} L'_j$ for each $\mathcal{R}_j[\tilde{x}]$ in $\tilde{\mathcal{R}}_L[\tilde{x}]$. Hence we get that

$$D[\mathcal{R}[\tilde{M}], \tilde{\mathcal{R}}_L[\tilde{M}] >^l D[M_i|_{w_2}, \tilde{L}]$$
$$D[\mathcal{R}[\tilde{M'}], \tilde{\mathcal{R}}_L[\tilde{M'}]] >^l D[M'_i|_{w_2}, \tilde{L'}]$$

where $l = k + k_1 + \cdots + k_s$. Per definition of analysis recipes, exactly one further rewrite is possible for the former context $D[M_i|_{w_2}, \tilde{L}]$, namely the primitive rewrite:

$$D[M_i|_{w_2}, \tilde{L}] >_r M_i|_w$$

The challenge is now to show that a reduction is possible on the latter context, $D[M_i'|_{w_2}, \tilde{L}']$. In the proof of Lemma 7 we simply relied on condition 2 in $\approx_{ss}$, but now this condition is not available because we are assuming the weaker static equivalence, $\approx_s$. So a more cunning approach is called for.

The assumption that $cores(\varphi) = ecores(\varphi)$ is the key here. We then get that any analysis term L which is not a core is syntactically equal to a *non-trivial* context over other analysis terms. For otherwise L would per definition be an ecore which is impossible since ecores are now cores (and L was assumed not to be a core). Hence for some terms $\tilde{T} \subseteq \mathcal{A}(\varphi)$ we can write

$$M_i|_{w_2} = C[M_i|_w, \tilde{T}]$$

Each $T_j \in \tilde{T}$ has some analysis recipe $\mathcal{R}_j[\tilde{x}]$, i.e. $\mathcal{R}_j[\tilde{M}] > T_j$; strictly speaking this should be shown in a separate induction, since the induction hypothesis does not necessarily apply to T_j, but the result should come as no surprise. Noting that

$$D[M_i|_{w_2}, \tilde{L}] =_\mathcal{E} M_i|_w$$

and using that any analysis term is equal to the instantiation of its analysis recipe, we reason as follows:

$$\begin{array}{ll}
M_i|_{w_2} = C[M_i|_w, \tilde{T}] & \\
M_i|_{w_2} =_\mathcal{E} C[D[M_i|_{w_2}, \tilde{L}], \tilde{T}] & \Updownarrow \\
\mathcal{R}[\tilde{M}] =_\mathcal{E} C[D[\mathcal{R}[\tilde{M}], \tilde{\mathcal{R}}_L[\tilde{M}]], \tilde{\mathcal{R}}_T[\tilde{M}]] & \Updownarrow \\
(\mathcal{R}[\tilde{x}] =_\mathcal{E} C[D[\mathcal{R}[\tilde{x}], \tilde{\mathcal{R}}_L[\tilde{x}]], \tilde{\mathcal{R}}_T[\tilde{x}]])\varphi &
\end{array}$$

Per definition of $\approx_s$ we may continue thus:

$$\begin{array}{ll}
(\mathcal{R}[\tilde{x}] =_\mathcal{E} C[D[\mathcal{R}[\tilde{x}], \tilde{\mathcal{R}}_L[\tilde{x}]], \tilde{\mathcal{R}}_T[\tilde{x}]])\varphi' & \\
\mathcal{R}[\tilde{M'}] =_\mathcal{E} C[D[\mathcal{R}[\tilde{M'}], \tilde{\mathcal{R}}_L[\tilde{M'}]], \tilde{\mathcal{R}}_T[\tilde{M'}]] & \Updownarrow \\
M_i'|_{w_2} =_\mathcal{E} C[D[M_i'|_{w_2}, \tilde{L}'], \tilde{T}'] &
\end{array}$$

The last step follows from the induction hypothesis: since $\mathcal{R}[\tilde{x}]$ is the analysis recipe for $M_i|_{w_2} \in \mathcal{A}^i(M_i, \varphi)$ we have that $\mathcal{R}[\tilde{M}] >^k M_i|_{w_2}$ and $\mathcal{R}[\tilde{M'}] >^k M_i'|_{w_2}$. The tuples $\tilde{L}'$ and $\tilde{T}'$ simply contain the normalised recipes from $\tilde{\mathcal{R}}_L[\tilde{M'}]$ and $\tilde{\mathcal{R}}_T[\tilde{M'}]$, respectively. Since these recipes are not for analysis terms in level i, the induction hypothesis does not apply to these. Hence we do not know that each $L_j' \in \tilde{L}'$ corresponds to the analysis term $L_j \in \tilde{L}$, and indeed we do not even know that L_j' is an analysis term. However this is unimportant – all that matters for our purpose is that each L_j' is irreducible.

From the last equality we deduce that $D[M'_i|_{w_2}, \tilde{L}']$ must be reducible. For suppose not towards a contradiction. The LHS of the last equality is irreducible because it is a subterm of M_i and we assume each $M_i \in im(\varphi)$ to be irreducible. Therefore, by confluence the normal form of the RHS is syntactically equal to $M_i|_{w_2}$. But this is impossible since the normal form of the RHS then contains $M_i|_{w_2}$ as a proper subterm.

Per definition of analysis recipes, only one reduction is possible on $D[M_i|_{w_2}, \tilde{L}']$, and since $M_i|_{w_2}$ is the revelator it must hold that

$$D[M_i|_{w_2}, \tilde{L}'] >_r M_i|_w$$

and we conclude as desired that

$$D[\mathcal{R}[\tilde{M}], \tilde{\mathcal{R}}_L[\tilde{M}] >^l D[M_i|_{w_2}, \tilde{L}] > M_i|_w$$
$$D[\mathcal{R}[\tilde{M}'], \tilde{\mathcal{R}}_L[\tilde{M}']] >^l D[M'_i|_{w_2}, \tilde{L}'] > M'_i|_w$$

Finally, the induction hypothesis gives that $\mathcal{R}[\tilde{x}]$ is also the analysis recipe for $M|_{w_2}$ and $\mathcal{R}_j[\tilde{x}]$ is the analysis recipe for L_j for each $L_j \in \tilde{L}$. Hence the context

$$D[\mathcal{R}[\tilde{x}], \tilde{\mathcal{R}}_L[\tilde{x}]]$$

is per definition also the analysis recipe for $M'_i|_w$. This concludes the proof. □

Lemma 14 (cf. Lemma 8). *Let φ and φ' be two frames with $\varphi \approx_s \varphi'$ and suppose that $cores(\varphi) = ecores(\varphi)$. Then it holds for any contexts $C_1[\tilde{y}]$ and $C_2[\tilde{y}]$ that*

$$C_1[\tilde{N}] =_{\mathcal{E}} C_2[\tilde{N}] \Leftrightarrow C_1[\tilde{N}'] =_{\mathcal{E}} C_2[\tilde{N}']$$

Proof. This is a corollary of Lemma 13. □

Lemma 15. *Let φ and φ' be two frames with $\varphi \approx_s \varphi'$ and suppose that $cores(\varphi) = ecores(\varphi)$. Then it holds for any contexts $C_1[\tilde{y}]$ and $C_2[\tilde{y}]$ where $C_1[\tilde{y}] \sqsupset$ or $C_1[\tilde{y}]\square$ that*

$$C_1[\tilde{N}] >_r C_2[\tilde{N}] \Leftrightarrow C_1[\tilde{N}'] >_r C_2[\tilde{N}']$$

Proof. We show the direction from left to right; the converse is symmetric. So suppose that

$$C_1[\tilde{N}] >_r C_2[\tilde{N}]$$

We must show that this rewrite is preserved when substituting cores from φ'. Since the rewrite is primitive $C_1[\tilde{y}]$ unifies with the LHS of some rewrite rule, i.e. there is a destructor context $D[x, \tilde{x}]$ and some other context $C[\tilde{y}]$ such that

$$C_1[\tilde{N}] = D[C[\tilde{N}], \tilde{N}] > C_2[\tilde{N}]$$

where $C[\tilde{N}]$ is the revelator. Since we are working with subterm theories, $C_2[\tilde{N}]$ is a subterm of $C[\tilde{N}]$. We now employ the same trick as in the proof Lemma

13: the assumption that $cores(\varphi) = ecores(\varphi)$ gives us that any analysis term which is not itself a core can be written as a non-trivial context over cores. Therefore there is some context $C_3[x, \tilde{x}]$ such that

$$\begin{aligned} C[\tilde{N}] &= C_3[C_2[\tilde{N}], \tilde{N}] \\ C[\tilde{N}] &=_{\mathcal{E}} C_3[D[C[\tilde{N}], \tilde{N}], \tilde{N}] && \Updownarrow \\ C[\tilde{N'}] &=_{\mathcal{E}} C_3[D[C[\tilde{N'}], \tilde{N'}], \tilde{N'}] \end{aligned}$$

The last step followed by Lemma 14. As before we conclude that $D[C[\tilde{N'}], \tilde{N'}]$ must be primitively reducible, although this time we call upon Lemma 10 to get that the LHS $C[\tilde{N'}]$ is irreducible (this lemma applies because we have assumed that $C_1[\tilde{y}] \sqsupset$ or $C_1[\tilde{y}]\Box$). Since $C[\tilde{N'}]$ is the revelator we then conclude as desired that

$$C_1[\tilde{N'}] = D[C[\tilde{N'}], \tilde{N'}] >_r C_2[\tilde{N'}]$$

$\square$

Theorem 3. *If $ecores(\varphi) = cores(\varphi)$ then $\varphi \approx_{ss} \varphi' \Leftrightarrow \varphi \approx_s \varphi'$.*

Proof. The direction from left to right is immediate from the definitions of $\approx_{ss}$ and $\approx_s$.

For the other direction it must be shown that, under the given assumptions, condition 2 in $\approx_{ss}$ is a consequence of condition 1 – i.e. we must show that

$$(M >^k)\varphi \Leftrightarrow (M >^k)\varphi'$$

for all terms M and all $k \in \mathbb{N}$.

First observe that any two corresponding terms $M_i \in im(\varphi)$ and $M'_i \in im(\varphi')$ can be written as the same context over cores. For otherwise there would be cores $N_j \in ecores(M_i, \varphi)$ and $N'_j \in ecores(M'_i, \varphi')$ which do not have the same analysis recipe, contradicting Lemma 13.

So suppose that $(M >^k)\varphi$ and let $ecores(\varphi) = (N)_{j \in J}$ and $ecores(\varphi') = (N')_{j \in J}$. Then $n(M) \cap bn(\varphi) = \emptyset$ and $M\varphi >^k$. Since every free name in φ is an ecore, we have for some context $C_1[\tilde{y}]$ that

$$\begin{aligned} M\varphi &= C_1[\tilde{N}] \\ M\varphi' &= C_1[\tilde{N'}] \end{aligned}$$

The proof now proceeds by induction in k in order to show that $C_1[\tilde{N}] >^k \Rightarrow C_1[\tilde{N'}] >^k$.

Basis (k = 0). This is trivial (use reflexivity).

Step (assume for k, prove for $k + 1$). We then have that

$$C_1[\tilde{N}] > C_2[\tilde{N}] >^k$$

for some $C_2[\tilde{y}]$. Per definition of the rewrite relation there is some subterm $C_1[\tilde{N}]|_{w_1}$ of $C_1[\tilde{N}]$ which rewrites primitively, i.e.

$$C_1[\tilde{N}]|_w >_r C_2^*[\tilde{N}]|_{w_2}$$

Apply Lemma 15 to get that also

$$C_1[\tilde{N'}]|_{w_1} >_r C_2[\tilde{N'}]|_{w_2}$$

As in the proof of Lemma 5 we then get that

$$C_1[\tilde{N'}] > C_2[\tilde{N'}]$$

Since $C_2[\tilde{N}] >^k$ it follows by the induction hypothesis that also $C_2[\tilde{N'}] >^k$. Hence we have that

$$M\varphi' = C_1[\tilde{N'}] >^{k+1}$$

which per definition means that $(M >^k)\varphi'$. □

Theorem 3 above can be rephrased to state that $\approx_{ss}$ and $\approx_s$ coincide in any theory where inaccessible terms are impossible. For if there are no inaccessible terms then any analysis term which is not an ecore can be written as contexts over other analysis terms, so it follows per definition of ecores that $ecores(\varphi) = cores(\varphi)$.

The condition that no inaccessible terms are possible is easy to check from the definition of a given equational theory. Take for example any analysis term term $enc(M_1, M_2)$ in the theory of symmetric key encryption. If $enc(M_1, M_2)$ is not an ecore then neither M_1 nor M_2 can be inaccessible. For M_1 is revealed from $enc(M_1, M_2)$ and hence is in the analysis. And in order for this to be possible, the symmetric key M_2 must also be in the analysis. Hence any frame φ in a symmetric key theory satisfies that $ecores(\varphi) = cores(\varphi)$ and so by Theorem 3 $\approx_{ss}$ and $\approx_s$ coincide.

In contrast, a term $enc(M_1, M_2)$ in a public key theory might have the public key M_2 inaccessible. For it may be that $enc(M_1, M_2)$ can be decrypted even though M_2 is not in the analysis; all that is required is for the corresponding private key to be in the analysis.

However, we can make the assumption that any public key is always known by the environment, i.e. any public key will always be in the analysis of a frame. Indeed, this is one of the main attractions of public-key cryptography: a principal may publish its public key for use by any interested parties. Under this assumption we *do* have that no inaccessible terms are possible, and hence that $\approx_{ss}$ and $\approx_s$ coincide. On the other hand, this assumption is not valid for public-key *signatures* where we would have the following rewrite rule – the only difference from the public-key decryption rule is that the order of public and private keys has been swapped:

$$dec(enc(z_1, z_2^-), z_2{}^+) >_r z_1$$

For here we would have to assume that the private key is always known to an environment, which obviously is not sound.

4.4 Summary

In this chapter we have shown two important results, namely that $\approx_{ss}$ and $\approx'_{ss}$ coincide and that $\approx_s$ and $\approx_{ss}$ coincide for frames where $cores(\varphi) = ecores(\varphi)$ – or, more generally, in theories with no inaccessible terms. These include theories with symmetric key encryption and theories with public key encryption, under the assumption that public keys are always known to the environment.

Chapter 5

A Logic for Frames

This chapter sets out to design a logic for frames which characterises strong static equivalence and which is amenable to construction of characteristic formulae. The refined definition of strong static equivalence, $\approx'_{ss}$, presented in Chapter 3 was derived exactly with this purpose in mind, so we will base our developments on $\approx'_{ss}$ instead of $\approx_{ss}$ whenever convenient. We established in Chapter 4 that the two definitions of strong static equivalence coincide, so the results in this section apply to both $\approx'_{ss}$ and $\approx_{ss}$.

The resulting first order logic for frames can be used to reason directly about the terms in the frames using propositions for equality, syntactic equality and reductions. Quantification ranges over the *synthesis* of the frame, which intuitively is the set of terms that an environment can deduce by constructing new terms from the ecores of the frame. Quantification thus allows indirect reasoning about the knowledge of an environment, and assertions such as *there exists a term y (known by the environment) such that x_1 can be decrypted with y to obtain x_2* can be made.

Section 1 gives an informal discussion of possible constructs for inclusion in the logic and Section 2 formalises the resulting syntax and semantics. In Section 3 it is shown that the logic characterises strong static equivalence; the proof of this result relies on lemmas from Chapter 4. Finally section 4 shows how characteristic formulae can be constructed. The construction is based on a further refinement of strong static equivalence which does not exhibit the problems of universal quantification over contexts which were still present in $\approx'_{ss}$.

5.1 Motivation

This section presents some examples and informal ideas on which the syntax and semantics for the logic for frames are based. The objective is to answer the question of which kind of assertions one would like to express about frames. We base our discussion on applications in cryptographic protocols but will still maintain a certain level of generality. Simplicity of the logic will have high priority for the sake of subsequent theoretical developments.

A key objective is to develop a logic which characterises strong static equivalence, i.e. whenever two frames are statically equivalent they should satisfy the same set of formulae. Another aim is to make characteristic formulae possible, i.e. for every frame φ there should exist a formula $\mathcal{C}_\varphi$ s.t. $\varphi \approx'_{ss} \varphi'$ if and only if $\varphi' \vDash \mathcal{C}_\varphi$ for all frames φ'. A starting point for achieving this is to include the operators featuring in the definition of $\approx'_{ss}$ in the logic.

We start by considering ideas for the basic propositional equational logic in the first subsection and then continue with a discussion of a first order logic (i.e. a logic with quantification over terms) in the second subsection.

5.1.1 Propositional Equational Logic

A formula in propositional logic is generally built from atomic propositions p_i, logical connectives (such as $\vee$, $\neg$, $\wedge$, $\rightarrow$ and $\leftrightarrow$), together with suitable bracketing; an example formula could thus be $(p_1 \vee p_2) \wedge \neg p_3$. In ordinary propositional logic, the atomic propositions are boolean variables whose truth values must be assigned by a *valuation function* in order to decide the truth of any given formula. So an atomic proposition (and hence a formula) is true with respect to some valuation. In the case of a logic for frames, the truth of an atomic proposition (and hence of a formula) should be relative to a frame, and (some of) the atomic propositions will be equalities, giving rise to an *equational* logic. In the following we will loosely write $\varphi \vDash A$ when the formula A is true in the frame φ; the precise definition of this satisfaction relation will be given in the next section.

First we must decide what kind of atomic propositions are meaningful and useful for reasoning about frames. We base our discussion on two statically equivalent frames in the public key theory $\mathcal{E}_{pub}$:

$$\varphi = (\nu a, b, c)\{{}^{enc(c^-,k^+)}/_{x_1},\, {}^{enc(b,c^+)}/_{x_2},\, {}^{b}/_{x_3}\}$$
$$\varphi' = (\nu a, b, c)\{{}^{enc(f(c)^-,k^+)}/_{x_1},\, {}^{enc(f(b),f(c)^+)}/_{x_2},\, {}^{f(b)}/_{x_3}\}$$

Equality of Terms in Frames

Equality between terms in frames immediately comes to mind as a possible atomic proposition. For example one would expect that the equality proposition $dec(x_2, dec(x_1, k^-)) =_{\mathcal{E}} x_3$ holds in the frame φ, written $\varphi \vDash dec(x_2, dec(x_1, k^-)) =_{\mathcal{E}} x_3$. On the other hand, the proposition $dec(x_1, k^-) =_{\mathcal{E}} c^-$ should not be true in φ since c is private. If it were true the logic would not characterise static equivalence since the proposition is certainly not true in φ'. Hence a meaningful semantics for the equality proposition is captured precisely in the definition of equality of terms in frames (Definition 5), i.e. $\varphi \vDash M_1 =_{\mathcal{E}} M_2$ if and only if $(M_1 =_{\mathcal{E}} M_2)\varphi$.

As a bonus feature, the formula $a =_{\mathcal{E}} a$ can be used to check that the name a is free in a frame, since the only reason why this formula may not hold in a frame φ is that $a \in bn(\varphi)$.

The equality proposition together with negation would be sufficient to characterise the weak version of static equivalence. However it does not yield much insight into the nature of a frame and is therefore uninteresting in itself. Furthermore, it would not suffice for constructing characteristic formulae, so we shall need to consider additional atomic propositions.

Reduction

Reduction propositions on the form $dec(x_2, dec(x_1, k^-)) > x_3$ will be required in characteristic formulae in order to express condition 3 in the definition of $\approx'_{ss}$. Intuitively this proposition would be true in φ and φ' (although technically the reduction requires two steps). The intended meaning is that $\varphi \vDash M_1 > M_2$ if and only if $M_1\varphi > M_2\varphi$. For the same reason as with equality, we must require that $n(M_1, M_2) \cap bn(\varphi) = \emptyset$.

Reduction propositions may not appear useful at a first glance because of our inability to express that e.g. $dec(x_1, k^+) > c^-$ (where c is private). But in combination with existential quantification (to be considered in the next subsection), the reduction proposition will give rise to significant expressive power.

Syntactic equality

Propositions with syntactic equality between terms on the form $M_1 = M_2$ is necessary to express condition 1 and 2 in the definition of $\approx'_{ss}$ in a characteristic formulae. The intended meaning of such propositions is, not surprisingly, that $\varphi \vDash M_1 = M_2$ if and only if $M_1\varphi = M_2\varphi$, and again we must require that $n(M_1, M_2) \cap bn(\varphi) = \emptyset$.

Other candidates

Ecores form another essential ingredient in the definition of $\approx'_{ss}$, and hence one might considering including in the logic a proposition $core(x, M)$ stating that the term M is an ecore in x. With a naive interpretation of the core proposition, we would have that for example $core(x_3, b)$ is true in the frame φ. However it would not be true in φ' (where the ecore of x_3 is $f(b)$) so the resulting logic would fail to characterise static equivalence. It turns out that an appropriate ecore proposition can be expressed using existential quantification and the reduction proposition, so we do not need to explicitly include it in our logic.

As another idea for atomic propositions, recall that a frame represents messages sent by some process on a public channel, and hence represents the knowledge of an environment. It would therefore be nice to directly assert that the environment can deduce a given term, i.e. that the term c^- can be deduced from φ (namely by decrypting x_1 with k^- since k is a free name). But c^- cannot be deduced from φ' – $f(c^-)$ can be deduced, but f is a one-way hash function. It follows that a deduction proposition would not characterise static equivalence and hence we do not include it in the logic. However, we shall see how to obtain similar expressiveness using first order quantification in the next subsection.

5.1.2 First-order Logic

Adding the quantifier $\exists$ yields a much more interesting logic with a resulting increase in expressiveness. It is however not obvious how the semantics for this quantifier should be defined; in particular, the domain over which a quantification ranges is of much significance. The intuition is that a frame φ should satisfy a formula $\exists x A$ (where A is another formula with the variable x free) if there is some term M such that φ satisfies $A\{^M/_x\}$. A slightly nicer way of expressing this is that $\varphi\{^M/_x\}$ must satisfy A. The question is now which domain we allow the term M to be drawn from.

A first idea may be to let quantification range over arbitrary terms. However this would result in characterisation failing; for example, φ' defined in the previous subsection would satisfy $\exists x(x_3 = f(x))$ (choose $x = b$) while φ clearly would not. The problem is intuitively that the quantified variable is bound to a term occurring on sub-core level in the frame; this is problematic since subterms of cores do not have any effect on static equivalence.

The solution is to let quantification range over the set of terms which can already be deduced from the frame. For example, the terms $f(b)$, $f(f(b))$ and $f(enc(f(b), k))$ can be deduced from φ' but b cannot (we formally define deduction in terms of a synthesis from ecores in the next section).

Existential quantification together with the atomic propositions can now be used to express many interesting properties about frames. For example we may express that the core b in $ecores(x_2, \varphi)$ has analysis recipe $dec(x_2, dec(x_1, k^-))$ as follows:

$$\exists y_1 \exists y_2 (dec(x_2, dec(x_1, k^-)) > y_1 \wedge y_1 > y_2 \wedge \neg \exists y_3 (y_2 > y_3))$$

because this formula is satisfied exactly if the term in question is the analysis recipe for the core b. Notice how the formula is satisfied by both φ and φ'. We shall rely on this idea when expressing ecores in the construction of characteristic formulae later in this chapter.

Existential quantification can also be used to express that x_2 can be decrypted using *some* private key known to the environment:

$$\exists y_1 \exists y_2 (dec(x_2, y_1) > y_2)$$

Again this formulae is true in both φ and φ' (for the former choose $y_1 = c^-$ and $y_2 = b$, and for the latter choose $y_1 = f(c)^-$ and $y_2 = f(b)$). Alternatively we can also use syntactic equality to express that the encrypted term in x_1 is known to the environment thus:

$$\exists y (x_1 = enc(y, k^+))$$

5.2 Syntax and Semantics

Formal syntax and semantics for the logic of frames will now be given. Having discussed the motivation in the last section, we will be brief.

5.2.1 Syntax

The syntax for the chosen first order logic for frames, $\mathcal{LF}$, is defined as follows, where the M_i range over terms and x ranges over variables:

$$\begin{aligned} A ::= & M_1 =_{\mathcal{E}} M_2 \mid M_1 > M_2 \mid M_1 = M_2 \mid A_1 \vee A_2 \\ & \mid \neg A_1 \mid (A_1) \mid \exists x(A_1) \end{aligned}$$

The full set of logical connectives is defined from $\neg$,$\vee$ and $\exists$ in the usual way:

$$\begin{aligned} A_1 \wedge A_2 &\triangleq \neg(\neg A_1 \vee \neg A_2) \\ A_1 \rightarrow A_2 &\triangleq A_2 \vee \neg A_1 \\ A_1 \leftrightarrow A_2 &\triangleq A_1 \rightarrow A_2 \wedge A_2 \rightarrow A_1 \\ \forall x(A_1) &\triangleq \neg \exists x(\neg A_1) \\ \mathbf{true} &\triangleq x =_{\mathcal{E}} x \\ \mathbf{false} &\triangleq \neg \mathbf{true} \end{aligned}$$

We assume that propositional connectives associate from the left i.e. $A_1 \vee A_2 \wedge A_3$ is interpreted as $(A_1 \vee A_2) \wedge A_3$, but parenthesis can always be used to explicitly override this. When there is no ambiguity we may omit parenthesis in existential quantification (and write e.g. $\exists y_1 \exists y_2 M_1 = M_2$). Free and bound variables in formulae are defined as expected, i.e. quantification is the only binder.

Example 5.2.1. The following is an example of a formula generated from the chosen grammar:

$$\exists y_1 \exists y_2 (dec(x_2, y_1) > y_2 \wedge \neg \exists y_3 \exists y_4 (dec(y_2, y_3) > y_4))$$

This expresses that x_2 can be decrypted using some known key and that the resulting term cannot be further decrypted. □

5.2.2 Semantics

Common for all three atomic propositions is the requirement that terms do not contain private names. Therefore we start by defining the notions of *reduction in frames* and *syntactic equality in frames* along the same line as equality in frames.

Definition 25 (Reduction in frame). *M_1 reduces to M_2 in φ, written $(M_1 > M_2)\varphi$, if and only if $n(M_1, M_2) \cap bn(\varphi) = \emptyset$ and $M_1\varphi > M_2\varphi$.*

Definition 26 (Syntactic equality in frame). *M_1 is syntactically equal to M_2 in φ, written $(M_1 = M_2)\varphi$, if and only if $n(M_1, M_2) \cap bn(\varphi) = \emptyset$ and $M_1\varphi = M_2\varphi$.*

We can now define the satisfaction relation $\models$ for the propositional logic:

$$\begin{aligned}
\varphi \models M_1 =_{\mathcal{E}} M_2 \quad &\triangleq \quad (M_1 =_{\mathcal{E}} M_2)\varphi \\
\varphi \models M_1 > M_2 \quad &\triangleq \quad (M_1 > M_2)\varphi \\
\varphi \models M_1 = M_2 \quad &\triangleq \quad (M_1 = M_2)\varphi \\
\varphi \models A_1 \vee A_2 \quad &\triangleq \quad \varphi \models A_1 \text{ or } \varphi \models A_2 \\
\varphi \models \neg A \quad &\triangleq \quad \varphi \not\models A
\end{aligned}$$

As discussed in the previous section, quantification should be restricted to the universe of terms which the environment can deduce from a given frame. Intuitively these are the terms which can be constructed by applying function symbols to terms from the ecores of the frame.

Definition 27 (Frame synthesis). *The set of terms deducible from a frame φ is called the synthesis of φ and is defined inductively as follows:*

$$\begin{aligned}
\mathcal{S}^0(\varphi) \quad &\triangleq \quad ecores(\varphi) \\
\mathcal{S}^{i+1}(\varphi) \quad &\triangleq \quad \mathcal{S}^i \cup \{f(T_1, \ldots, T_k) \mid f \in \Sigma^{(k)} \text{ and } T_1, \ldots, T_k \in \mathcal{S}^i(\varphi)\}
\end{aligned}$$

It follows that the synthesis is infinite while the analysis and ecores are finite. Since any analysis term can be written as a context over ecores, the containment hierarchy between the three sets is

$$ecores(\varphi) \subseteq \mathcal{A}(\varphi) \subset \mathcal{S}(\varphi)$$

Note that the synthesis can alternatively be defined as the set of all contexts over ecores.

Our definition of synthesis is similar to that of deduction from frames by Abadi et al. in [2, Section 2.3], but there are two differences: first, deduction includes terms with arbitrary fresh names (i.e. non-private names which do not occur in the frame). Second, deduction also includes terms M with arbitrary bound names, as long as bound names in $M\!\downarrow$ occur only as subterms of analysis terms. For example, the term $M = \mathit{fst}([a, b])$ can be deduced from a frame where b is private and not in the analysis, but M would not be in the synthesis of such frame. The differences between deduction and synthesis are not significant for our purpose though; we adopt synthesis for technical convenience since any synthesis term per definition can be written as a primitive context over ecores. Also there does not seem to be any reason why terms with fresh names should be of interest in quantification.

The definition of satisfaction can now be completed with the case for existentially quantified formulae:

$$\varphi \models \exists x(A_1) \quad \triangleq \quad \varphi\{{}^{M}\!/\!_{x}\} \models A_1 \text{ for some term } M \in \mathcal{S}(\varphi)$$

Observe how bindings of quantified variables are represented in a natural way by extending the frame with an additional substitution. In cases where $x \in dom(\varphi)$ we assume alpha-conversion of x to some $x' \notin dom(\varphi)$, since otherwise a quantification may overwrite existing terms in φ.

5.3 Characterisation

In this section we show that $\mathcal{LF}$ characterises $\approx_{ss}$ and hence $\approx'_{ss}$. In order to do this, we need a result which says that two strong statically equivalent frames φ and φ' can be extended with appropriate terms M and M' while maintaining strong static equivalence. This will be expressed in the following two *extension lemmas*

Lemma 16 (Extension Lemma 1). *Let* $\varphi = (\nu\tilde{n})\{{}^{M_i}/_{x_i}\}_{i \in I}$ *and* $\varphi' = (\nu\tilde{n})\{{}^{M'_i}/_{x_i}\}_{i \in I}$ *be two frames with* $\varphi \approx_{ss} \varphi'$*, and let* $ecores(\varphi) = (N)_{j \in J}$ *and* $ecores(\varphi') = (N')_{j \in J}$*. Then for any* $j \in J$ *it holds that*

$$\varphi\{{}^{N_j}/_{x_s}\} \approx_{ss} \varphi'\{{}^{N'_j}/_{x_s}\}$$

(where s *is any index with* $s \notin I$*).*

Proof. It must be shown that both conditions of $\approx_{ss}$ hold for the extended frames. For this we rely on Lemma 7 which says that N_j and N'_j have the same analysis recipes.

Condition 1. Suppose that $(M_1 =_{\mathcal{E}} M_2)\varphi\{{}^{N_j}/_{y}\}$, i.e. $n(M_1, M_2) \cap bn(\varphi\{{}^{N_j}/_{y}\}) = \emptyset$ and $M_1\varphi\{{}^{N_j}/_{y}\} =_{\mathcal{E}} M_2\varphi\{{}^{N_j}/_{y}\}$. If $x_s \notin v(M_1, M_2)$ we are done, so suppose that $x_s \in v(M_1, M_2)$. Suppose further for notational convenience and without loss of generality that both $x_s \in v(M_1)$ and $x_s \in v(M_2)$. M_1 and M_2 are in fact contexts, so we reason as follows:

$$\begin{aligned} M_1\varphi\{{}^{N_j}/_{y}\}) &=_{\mathcal{E}} M_2\varphi\{{}^{N_j}/_{y}\} \\ C_1[\tilde{x}, x_s]\varphi\{{}^{N_j}/_{x_s}\} &=_{\mathcal{E}} C_2[\tilde{x}, x_s]\varphi\{{}^{N_j}/_{x_s}\} && \Updownarrow \\ C_1[\tilde{M}, N_j] &=_{\mathcal{E}} C_2[\tilde{M}, N_j] \end{aligned}$$

Now let $\mathcal{R}[\tilde{x}]$ be the analysis recipe for N_j. By Lemma 7 on p. 52, $\mathcal{R}[\tilde{M}] >^k N_j$ and hence also $\mathcal{R}[\tilde{M}] =_{\mathcal{E}} N_j$. Using that $\varphi \approx_{ss} \varphi'$ and condition 1 of $\approx_{ss}$ we continue thus:

$$\begin{aligned} C_1[\tilde{M}, \mathcal{R}[\tilde{M}]] &=_{\mathcal{E}} C_2[\tilde{M}, \mathcal{R}[\tilde{M}]] \\ (C_1[\tilde{x}, \mathcal{R}[\tilde{x}]] &=_{\mathcal{E}} C_2[\tilde{x}, \mathcal{R}[\tilde{x}]])\varphi && \Updownarrow \\ (C_1[\tilde{x}, \mathcal{R}[\tilde{x}]] &=_{\mathcal{E}} C_2[\tilde{x}, \mathcal{R}[\tilde{x}]])\varphi' && \Updownarrow \\ C_1[\tilde{M}', \mathcal{R}[\tilde{M}']] &=_{\mathcal{E}} C_2[\tilde{M}', \mathcal{R}[\tilde{M}']] \end{aligned}$$

By Lemma 7 again, $\mathcal{R}[\tilde{x}]$ is also the analysis recipe for N'_j and $\mathcal{R}[\tilde{M}] >^k N'_j$. The proof of condition 1 can then be completed thus:

$$\begin{array}{ll} C_1[\tilde{M}', N'_j] =_{\mathcal{E}} C_2[\tilde{M}', N'_j] & \\ (C_1[\tilde{x}, x_s] =_{\mathcal{E}} C_2[\tilde{x}, x_s])\varphi'\{{}^{N'_j}/_{x_s}\} & \Updownarrow \\ (M_1 =_{\mathcal{E}} M_2)\varphi'\{{}^{N'_j}/_{x_s}\} & \end{array}$$

Condition 2. This is somewhat similar to the proof of condition 1. If we again have that $\mathcal{R}[\tilde{M}] >^k N_j$ we use that $\varphi \approx_{ss} \varphi'$, and condition 2 in $\approx_{ss}$ then allows us to reason as follows

$$\begin{array}{ll} (M >^c)\varphi\{{}^{N_j}/_{x_s}\} & \\ (C[\tilde{x}, x_s] >^c)\varphi\{{}^{N_j}/_{x_s}\} & \Updownarrow \\ C[\tilde{M}, N_j] >^c & \Updownarrow \\ C[\tilde{M}, \mathcal{R}[\tilde{M}]] >^{c+k} & \Updownarrow \\ (C[\tilde{x}, \mathcal{R}[\tilde{x}]] >^{c+k})\varphi & \Updownarrow \\ (C[\tilde{x}, \mathcal{R}[\tilde{x}]] >^{c+k})\varphi' & \Updownarrow \\ C[\tilde{M}', \mathcal{R}[\tilde{M}']] >^{c+k} & \Updownarrow \\ C[\tilde{M}', N'_j]] >^c & \Updownarrow \\ (C[\tilde{x}, x_s] >^c)\varphi'\{{}^{N'_j}/_{x_s}\} & \end{array}$$

In contrast to the proof of condition 1, the above line of reasoning relies crucially on Lemma 7 saying that $\mathcal{R}[\tilde{M}] >^k N_j$ and $\mathcal{R}[\tilde{M}'] >^k N'_j$, i.e. that the recipe instances reduce in exactly the same number of steps to the respective cores. $\square$

Lemma 17 (Extension Lemma 2). *Let $\varphi = (\nu\tilde{n})\{{}^{M_i}/_{x_i}\}_{i\in I}$ and $\varphi' = (\nu\tilde{n})\{{}^{M'_i}/_{x_i}\}_{i\in I}$ be two frames with $\varphi \approx_{ss} \varphi'$, and let $ecores(\varphi) = (N)_{j\in J}$ and $ecores(\varphi') = (N')_{j\in J}$. Then for any $C[\tilde{y}]$ it holds that*

$$\varphi\{{}^{C[\tilde{N}]}/_{x_s}\} \approx_{ss} \varphi'\{{}^{C[\tilde{N}']}/_{x_s}\}$$

(where s is any index with $s \notin I$).

Proof. It must be shown that both conditions of $\approx_{ss}$ hold for the extended frames. We just show condition 1 as condition 2 is similar.

Condition 1. Suppose that $(M_1 =_{\mathcal{E}} M_2)\varphi\{{}^{C[\tilde{N}]}/_y\})$, i.e. $n(M_1, M_2) \cap bn(\varphi\{{}^{C[\tilde{N}]}/_y\}) = \emptyset$ and $M_1\varphi\{{}^{C[\tilde{N}]}/_y\} =_{\mathcal{E}} M_2\varphi\{{}^{C[\tilde{N}]}/_y\}$. If $x_s \notin v(M_1, M_2)$ we are done, so suppose that $x_s \in n(M_1, M_2)$. Suppose further for notational convenience and without loss of generality that both $x_s \in v(M_1)$ and $x_s \in v(M_2)$.

First apply Lemma 16 iteratively to $\varphi \approx_{ss} \varphi'$ to get that

$$\varphi\{{}^{N_j}/_{y_j}\}_{j\in J} \approx_{ss} \varphi'\{{}^{N'_j}/_{y_j}\}_{j\in J}$$

We then use this to reason as follows.

$$
\begin{array}{rcll}
C_1[\tilde{x}, y]\varphi\{{}^{C[\tilde{N}]}/_{y}\} & =_{\mathcal{E}} & C_2[\tilde{x}, y]\varphi\{{}^{C[\tilde{N}]}/_{y}\} & \\
C_1[\tilde{M}, C[\tilde{N}]] & =_{\mathcal{E}} & C_2[\tilde{M}, C[\tilde{N}]] & \Updownarrow \\
C_1[\tilde{x}, C[\tilde{y}]]\varphi\{{}^{N_j}/_{y_j}\}_{j\in J} & =_{\mathcal{E}} & C_2[\tilde{x}, C[\tilde{y}]]\varphi\{{}^{N_j}/_{y_j}\}_{j\in J} & \Updownarrow \\
C_1[\tilde{x}, C[\tilde{y}]]\varphi'\{{}^{N'_j}/_{y_j}\}_{j\in J} & =_{\mathcal{E}} & C_2[\tilde{x}, C[\tilde{y}]]\varphi'\{{}^{N'_j}/_{y_j}\}_{j\in J} & \Updownarrow \\
C_1[\tilde{M}', C[\tilde{N}']] & =_{\mathcal{E}} & C_2[\tilde{M}', C[\tilde{N}']] & \Updownarrow \\
C_1[\tilde{x}, y]\varphi'\{{}^{C[\tilde{N}']}/_{y}\} & =_{\mathcal{E}} & C_2[\tilde{x}, y]\varphi'\{{}^{C[\tilde{N}']}/_{y}\} & \Updownarrow \\
M_1\varphi'\{{}^{C[\tilde{N}']}/_{y}\} & =_{\mathcal{E}} & M_2\varphi'\{{}^{C[\tilde{N}']}/_{y}\} &
\end{array}
$$

□

Technically we could also have given a direct proof of Lemma 17 above, but the presentation and notation would have been more involved. Note that static equivalence is not preserved by extension with arbitrary terms – take for example the following two frames in the theory $\mathcal{E}_{pub}$ (where we only use hash functions):

$$
\begin{array}{rcl}
\varphi & \triangleq & (\nu a)\{{}^{f(a)}/_{x_1}\} \\
\varphi' & \triangleq & (\nu a)\{{}^{a}/_{x_1}\}
\end{array}
$$

Then $\varphi \approx_{ss} \varphi'$. But $\varphi\{{}^{f(a)}/_{x_2}\} \not\approx_{ss} \varphi'\{{}^{f(a)}/_{x_2}\}$ because the equality $x_1 =_{\mathcal{E}} x_2$ holds in the extended φ but not in the extended φ'. So even though all the equalities which hold in the original frames also hold in the extended frames, there may be new equalities which hold in only one of the extended frames. Hence it is essential that the two frames are extended with the same context over their respective cores.

With the second extension lemma in our toolbox we now arrive at the main result of this section, namely that the logic $\mathcal{LF}$ characterises $\approx_{ss}$. For stating the characterisation theorem we find it convenient to introduce the notion of a *logical theory*:

Definition 28. *The logical theory of frame φ in the logic $\mathcal{LF}$ is the class of formulae in $\mathcal{LF}$ which are true in φ:*

$$
Th_{\mathcal{LF}}(\varphi) \;\triangleq\; \{A \in \mathcal{LF} \mid \varphi \vDash A\}
$$

Theorem 4 ($\mathcal{LF}$ characterises $\approx_{ss}$). $\varphi \approx_{ss} \varphi' \Leftrightarrow Th_{\mathcal{LF}}(\varphi) = Th_{\mathcal{LF}}(\varphi')$.

Proof. Let $\varphi \equiv (\nu\tilde{n})\{{}^{M_i}/_{x_i}\}_{i\in I}$ and $\varphi' \equiv (\nu\tilde{n})\{{}^{M'_i}/_{x_i}\}_{i\in I}$ be any frames with $ecores(\varphi) = (N)_{j\in J}$ and $ecores(\varphi') = (N')_{j\in J}$. There are two directions to prove.

($\Leftarrow$) Suppose that $Th_{\mathcal{LF}}(\varphi) = Th_{\mathcal{LF}}(\varphi')$, i.e. $(\varphi \vDash A \Leftrightarrow \varphi' \vDash A)$ for all formulae $A \in \mathcal{LF}$. In particular this holds for the class of formulae $M_1 =_{\mathcal{E}} M_2$, i.e. ($\varphi \vDash M_1 =_{\mathcal{E}} M_2 \Leftrightarrow \varphi' \vDash M_1 =_{\mathcal{E}} M_2$). Per definition of satisfaction, $(M_1 =_{\mathcal{E}} M_2)\varphi \Leftrightarrow (M_1 =_{\mathcal{E}} M_2)\varphi'$, which proves condition 1.

Condition 2 in $\approx_{ss}$ states that $(M >^k)\varphi \Leftrightarrow (M >^k)\varphi'$; we show the direction from left to right (the converse is symmetric). So suppose that $(M >^k)\varphi$, which per definition of rewrites in frames means that $n(M) \cap bn(\varphi) = \emptyset$ and $M >^k M^k$ for some M^k. Expanding on the transitive closure relation, there are terms $M^1, \ldots, M^{k-1}$ such that $M > M^1 > \cdots > M^{k-1} > M^k$. Each of these terms are clearly in $\mathcal{S}(\varphi)$, so it holds that:

$$\varphi \vDash \exists y_1 \exists y_2 \ldots \exists y_k (M > y_1 \wedge y_1 > y_2 \wedge \cdots \wedge y_{k-1} > y_k)$$

and per assumption also

$$\varphi' \vDash \exists y_1 \exists y_2 \ldots \exists y_k (M > y_1 \wedge y_1 > y_2 \wedge \cdots \wedge y_{k-1} > y_k)$$

which per definition of satisfaction means that there are terms $M^1, \ldots, M^k \in \mathcal{S}(\varphi')$ such that $M > M^1 > \cdots > M^{k-1} > M^k$. Hence $(M >^k)\varphi'$.

($\Rightarrow$) Suppose that $\varphi \approx_{ss} \varphi'$. We show by structural induction on formula A that ($\varphi \vDash A \Rightarrow \varphi' \vDash A$) holds for all formulae A; the proof of the converse is identical.

Basis *Case:* $A = (M_1 =_{\mathcal{E}} M_2)$. Per definition of satisfaction, $(M_1 =_{\mathcal{E}} M_2)\varphi$. Per assumption of static equivalence also $(M_1 =_{\mathcal{E}} M_2)\varphi'$, which again per definition of satisfaction means that $\varphi' \vDash M_1 =_{\mathcal{E}} M_2$.

Case: $A = (M_1 > M_2)$. Per definition of satisfaction, $M_1\varphi > M_2\varphi$ and $n(M_1, M_2) \cap bn(\varphi) = \emptyset$. We then have that M_1 and M_2 are in fact contexts over variables and free names $\tilde{n}$, and as in the proof of Theorem 1 we get that

$$C_{M_1}[M_{a_1}, \ldots M_{a_s}, \tilde{n}] > C_{M_2}[M_{b_1}, \ldots M_{b_s}, \tilde{n}]$$

Since $\approx_{ss}$ implies $\approx'_{ss}$, we can rely on the definition of $\approx'_{ss}$ and the results from Subsection 4.1. Condition 1 in $\approx'_{ss}$ gives that each $M \in im(\varphi)$ can be written as a context over ecores, and since any free name is also an ecore we have that

$$C_{M_1}[C_{a_1}[\tilde{N}], \ldots C_{a_s}[\tilde{N}], \tilde{N}] > C_{M_2}[C_{b_1}[\tilde{N}], \ldots C_{b_s}[\tilde{N}], \tilde{N}]$$

Applying Lemma 5 (p. 48) gives:

$$C_{M_1}[C_{a_1}[\tilde{N'}], \ldots C_{a_s}[\tilde{N'}], \tilde{N}] > C_{M_2}[C_{b_1}[\tilde{N'}], \ldots C_{b_s}[\tilde{N'}], \tilde{N}]$$

By condition 1 of $\approx'_{ss}$ again, each context $C_{a_i}[\tilde{N}]$ is syntactically equal to the corresponding term $M'_{a_1} \in im(\varphi')$, and since φ and φ' have the same free names we get

$$C_{M_1}[M'_{a_1}, \ldots M'_{a_s}, \tilde{n}] > C_{M_2}[M'_{b_1}, \ldots M'_{b_s}, \tilde{n}]$$

Hence $M_1\varphi' > M_2\varphi'$, which for the terms M_1 and M_2 is equivalent to $(M_1 > M_2)\varphi'$.

Case: $A = (M_1 = M_2)$. Per definition of satisfaction, $M_1\varphi = M_2\varphi$ and $n(M_1, M_2) \cap bn(\varphi) = \emptyset$. An argument similar to the proof above applies, except that we call on Lemma 2 (p. 46) instead of Lemma 5. This gives that $M_1\varphi' = M_2\varphi'$, i.e. $\varphi' \vDash M_1 = M_2$.

Step *Case:* $A = \neg A_1$. By the induction hypothesis and the assumption $\varphi \approx_{ss} \varphi'$ we have that $(\varphi \vDash A_1) \Leftrightarrow (\varphi' \vDash A_1)$, which is equivalent to $(\varphi \nvDash A_1) \Leftrightarrow (\varphi' \nvDash A_1)$. This per definition of satisfaction implies that $(\varphi \vDash \neg A_1) \Leftrightarrow (\varphi' \vDash \neg A_1)$.

Case: $A = A_1 \vee A_2$. By the induction hypothesis we have that $(\varphi \vDash A_i) \Leftrightarrow (\varphi' \vDash A_i)$ for $i = 1, 2$. There are three sub-cases to consider: 1) φ and φ' both satisfy A_1 and A_2 and hence both satisfy $A_1 \vee A_2$. 2) Neither φ nor φ' satisfy A_1 and A_2 and hence neither satisfy $A_1 \vee A_2$. 3) φ and φ' both satisfy exactly one of A_1 and A_2 and hence they both satisfy $A_1 \vee A_2$. Hence $(\varphi \vDash A_1 \vee A_2) \Leftrightarrow (\varphi' \vDash A_1 \vee A_2)$

Case: $A = \exists x A_1$. Per definition of satisfaction, $\varphi \vDash \exists x(A_1)$ means that $\varphi\{{}^{M}/_{x}\} \vDash A_1$ for some $M \in \mathcal{S}(\varphi)$. Any synthesis term can be written as a context over ecores, i.e. $M = C[\tilde{N}]$ for some context $C[\tilde{y}]$. Let $M' = C[\tilde{N'}]$. The second extension lemma (Lemma 17) then says that a $\varphi\{{}^{M}/_{x}\} \approx_{ss} \varphi'\{{}^{M'}/_{x}\}$. Applying the induction hypothesis to this gives that also $\varphi'\{{}^{M'}/_{x}\} \vDash A_1$. Now any context over cores is a synthesis term, so per definition of satisfaction we get that $\varphi' \vDash \exists x A_1$. ☐

We have now shown that the first order logic $\mathcal{LF}$ characterises $\approx_{ss}$ (and hence also $\approx'_{ss}$). One may ask if the propositional logic alone, without existential quantification, would also characterise $\approx_{ss}$. The answer is probably no. For existential quantification was used in the above proof to show condition 2 of $\approx_{ss}$. Condition 2 states, as a special case, that whenever a term $M_i \in im(\varphi)$ can reveal a subterm $M_i|_w$ then the corresponding term $M'_i \in im(\varphi')$ can reveal the corresponding subterm $M'_i|_w$. Hence condition 2 indirectly involves analysis terms. But the basic propositions in $\mathcal{LF}$ cannot be used to reason about arbitrary analysis terms because they cannot include terms with private names. Hence it would seem that existential quantification over the synthesis (and hence also over the analysis) is strictly necessary for characterisation.

5.4 Characteristic Formulae

We have now given syntax and semantics for a logic of frames and shown that it characterises $\approx'_{ss}$. The development of $\approx'_{ss}$, and the choice of atomic propositions for the logic, has been guided by our desire to find characteristic formulae for frames. But $\approx'_{ss}$ is still too high-level to yield a direct encoding into characteristic formula. The problem has been pointed out earlier, namely that condition 2 and 3 in $\approx'_{ss}$ involve infinitely many contexts. In the first subsection we therefore set out to give a further refinement of strong static equivalence which does not exhibit this problem. In subsections 2 and 3, encodings of revelation and ecores into the logic are given. Subsection 4 gives an encoding of each of the conditions in the further refined definition of strong static equivalence, and subsection 5 collects the efforts to give a unified construction of characteristic formula.

5.4.1 A Further Refinement of Strong Static Equivalence

Recall from Section 3.3 that condition 2 in the definition of $\approx'_{ss}$ generally involves infinitely many partitioning contexts which are unifiable with, but neither more general than or less general than, the LHS of some rewrite rule. We gave an example with the public key rewrite rule which gives rise to the following infinite sequence of partitioning contexts:

$$\begin{aligned}
&dec(enc(\bot, y_1), y_2),\\
&dec(enc(\bot, {y_1}^+), y_2),\\
&dec(enc(\bot, f(y_1)^+), y_2),\\
&dec(enc(\bot, f(f(y_1))^+), y_2),\\
&\ldots
\end{aligned}$$

A characteristic formula must capture all the instantiations of partitioning contexts which are not reducible, e.g. that

$$\begin{aligned}
&dec(enc(\bot, N_1), N_2) \not>,\\
&dec(enc(\bot, {N_1}^+), N_2) \not>,\\
&dec(enc(\bot, f(N_1)^+), N_2) \not>,\\
&dec(enc(\bot, f(f(N_1))^+), N_2) \not>,\\
&\ldots
\end{aligned}$$

But expressing this directly in the logic would give rise to infinite conjunction. Instead we just choose the first of the contexts and use existential quantification to express that

$$\neg\exists y^*.dec(enc(\bot, y^*), N_2) >$$

This works because existential quantification ranges over the synthesis of a frame and because every synthesis term can be written as a context over cores. The chosen context is what we refer to as a *minimised context*, the definition of which is given next.

Definition 29 (Minimised Contexts). *A minimised context $C[\tilde{y}, \tilde{y^*}]$ is a context with $C[\tilde{y}, \tilde{y^*}] \sqsupset$ such that every superterm of $y^* \in \tilde{y^*}$ contains $\bot$ or some $y \in \tilde{y}$; formally, whenever $C[\tilde{y}]|_w = y^* \in \tilde{y^*}$ it holds for every proper prefix w' of w that $st(C[\tilde{y}]|_{w'}) \cap (\{\bot\} \cup \tilde{y}) \neq \emptyset$.*

$\tilde{y^*}$ is a distinguished set of variables, referred to as the minimised variables, which are intended to be bound to synthesis terms rather than ecores.

Since minimised contexts are per definition more general than some rewrite rule, there are only finitely many of them. This enables us to give a further refinement of strong static equivalence using existential quantification and involving only finitely many contexts in the rewrite conditions. The problem with infinitely many contexts in condition 2 of $\approx'_{ss}$ is resolved in a similar (but more straight forward) manner.

Definition 30 (Further Refined Strong Static Equivalence). *Let* $\varphi \equiv (\nu\tilde{n})\{^{M_i}/_{x_i}\}_{i\in I}$ *and* $\varphi' \equiv (\nu\tilde{n})\{^{M'_i}/_{x_i}\}_{i\in I}$ *be two frames with* $ecores(\varphi) = (N)_{j\in J}$, $ecores(\varphi') = (N')_{j\in J}$. *Then define* $\varphi \approx''_{ss} \varphi'$ *iff* $dom(\varphi) = dom(\varphi')$ *and for each* $i \in I$ *the following conditions hold:*

1 *For some context* $C[\tilde{y}]$ *it holds that*

- $M_i = C[\tilde{N}]$
- $M'_i = C[\tilde{N}']$

2a *For all* $i, j \in J$ *it holds that*

$$N_i = N_j \Leftrightarrow N'_i = N'_j$$

2b *For all contexts* $C[\tilde{y}]$ *and for all* $j \in J$ *it holds that*

$$C[\tilde{N}] = N_j \Rightarrow C[\tilde{N}'] = N'_j$$

2c *If there there is no context* $C[\tilde{y}]$ *and* $j \in J$ *s.t.* $C[\tilde{N}] = N_j$*, then it must hold for all* $f(x_1, \ldots, x_k) \in \Sigma^{(k)}$ *that:*

$$\neg\exists M_1, \ldots, M_k \in \mathcal{S}(\varphi').f(M_1, \ldots, M_k) = N'_j$$

3a *For any partitioning context* $C_1^{\perp}[\tilde{y}]$ *where* $C_1^{\perp}[\tilde{y}] \sqsupset$ *it holds for all* $C_2^{\perp}[\tilde{y}]$ *that*

$$C_1^{\perp}[\tilde{N}] > C_2^{\perp}[\tilde{N}] \Leftrightarrow C_1^{\perp}[\tilde{N}'] > C_2^{\perp}[\tilde{N}']$$

3b *For any partitioning context* $C_1^{\perp}[\tilde{y}]$ *where* $C_1^{\perp}[\tilde{y}]\Box$ *it holds for all* $C_2^{\perp}[\tilde{y}]$ *that*

$$C_1[\tilde{N}] > C_2[\tilde{N}] \Rightarrow C_1[\tilde{N}'] > C_2[\tilde{N}']$$

3c *For any minimised partitioning context* $C_1^{\perp}[\tilde{y}, \tilde{y}^*]$ *where* $C_1^{\perp}[\tilde{y}, \tilde{y}^*]\Box$ *it holds for all* $C_2^{\perp}[\tilde{y}]$ *that*

$$\neg\exists T_1, \ldots, T_k \in \mathcal{S}(\varphi).C_1[\tilde{N}, T_1, \ldots, T_k] >_r C_2[\tilde{N}] \Rightarrow \\ \neg\exists T'_1, \ldots, T'_k \in \mathcal{S}(\varphi').C_1[\tilde{N}', T'_1, \ldots, T'_k] >_r C_2[\tilde{N}']$$

Lemma 18. $\approx'_{ss}$ *and* $\approx''_{ss}$ *coincide.*

Proof. In other words, we must show for all frames that $\varphi \approx'_{ss} \varphi' \Leftrightarrow \varphi \approx''_{ss} \varphi'$. So there are two directions to prove; we start with the easiest one.

(⇒). Suppose that $\varphi \approx'_{ss} \varphi'$. We must show that all conditions in $\approx''_{ss}$ are satisfied.

Condition 1. This is the same as condition 1 in $\approx'_{ss}$.

Conditions $2a$ **and** $2b$. These are both special cases of condition 2 in $\approx'_{ss}$ (to get condition $2a$ just choose $C[\tilde{N}]$ to be the trivial context y_j).

Condition $2c$. Suppose for a contradiction that condition $2c$ fails, i.e. there is no context $C[\tilde{y}]$ and $j \in J$ such that $C[\tilde{N}] = N_j$, but there *is* some function symbol $f(x_1, \ldots, x_k) \in \Sigma^{(k)}$ and some terms $T'_1, \ldots, T'_k \in \mathcal{S}(\varphi')$ such that $f(T'_1, \ldots, T'_k) = N'_j$. Per definition of synthesis, each T'_i can be written as a context over cores, i.e. $T'_i = C_i[\tilde{N}']$. This gives that $f(C_1[\tilde{N}'], \ldots, C_k[\tilde{N}']) = N'_j$. By condition 2 in the definition of $\approx'_{ss}$ it also holds that $f(C_1[\tilde{N}], \ldots, C_k[\tilde{N}]) = N_j$, which contradicts the assumption that no such context exists.

Conditions $3a$ **and** $3b$. These are both special cases of condition 3 in $\approx''_{ss}$

Condition $3c$. We show the contraposition of condition $3c$ (the idea is similar to that employed for condition $2c$). So suppose that there *are* terms $T'_1, \ldots, T'_k \in \mathcal{S}(\varphi')$ such that $C_1[\tilde{N}', T'_1, \ldots, T'_k] > C_2[\tilde{N}']$. Per definition of synthesis, each T'_i can be written as a context over cores, i.e. $T'_i = C_{T_i}[\tilde{N}']$. This gives that $C_1[\tilde{N}', C_{T_1}[\tilde{N}'], \ldots, C_{T_k}[\tilde{N}']] > C_2[\tilde{N}'_j]$. By condition 2 in the definition of $\approx'_{ss}$ it also holds that $C_1[\tilde{N}, C_{T_1}[\tilde{N}], \ldots, C_{T_k}[\tilde{N}]] > C_2[\tilde{N}]$. Hence there *are* $T_1, \ldots, T_k \in \mathcal{S}(\varphi)$ such that $C_1[\tilde{N}, T_1, \ldots, T_k] > C_2[\tilde{N}]$ (namely $T_i = C_{T_i}[\tilde{N}]$).

(⇐). Suppose that $\varphi \approx''_{ss} \varphi'$. We must show that all three conditions in $\approx'_{ss}$ are satisfied.

Condition 1. This is the same as condition 1 in $\approx''_{ss}$.

Condition 2. It must be shown that $C[\tilde{N}] = N_j \Leftrightarrow C[\tilde{N}'] = N'_j$. The direction from left to right is immediate from condition $2b$. For the other direction we assume $C[\tilde{N}'] = N'_j$ and show that also $C[\tilde{N}] = N_j$. This goes by induction in the height h of the parse tree of the term N'_j.

Basis (h = 0). Then N' is a name and we get that $C[\tilde{y}]$ is the trivial context y_j for some j. The result then follows from condition $2a$.

Step (assume for h, prove for $h+1$). If $C[\tilde{y}]$ is the trivial context, i.e. $C[\tilde{y}] = y_i$ for some i, we have that $N'_i = N'_j$ and the result follows immediately from condition $2a$.

Otherwise we assume for notational simplicity that $C[\tilde{N}'] = f(C_2[\tilde{N}']) = N'_j$ for some context $C_2[\tilde{y}]$, i.e. the unary function symbol f is the outer-most function symbol of N'_j (the generalisation is straight forward). We first show that

$$f(C_3[\tilde{N}]) = N_j \textit{ for some context } C_3[\tilde{y}]$$

i.e. that f is also the outer-most function symbol of N_j. To see this observe that there must be some context $C_4[\tilde{y}]$ such that $C_4[\tilde{N}] = N_j$. For otherwise condition 2d gives that there is no function symbol $f(x_1, \ldots, x_k) \in \Sigma^{(k)}$ and no terms $T_1, \ldots, T_k \in \mathcal{S}(\varphi')$ such that $f(T_1, \ldots, T_k) = N'_j$. But this is impossible because each core N' is in $\mathcal{S}(\varphi)$ and $C[\tilde{N}'] = N'_j$. Now condition 2c gives that also $C_4[\tilde{N}'] = N'_j$, and hence $C_4[\tilde{N}'] = f(C_2[\tilde{N}'])$. So we conclude that $C_4[\tilde{N}] = f(C_3[\tilde{N}]) = N_j$ for some context $C_3[\tilde{N}]$.

Next we show that

$$C_3[\tilde{N}] = C_2[\tilde{N}]$$

It follows from $f(C_3[\tilde{N}]) = N_j$ and condition $2b$ that also $f(C_3[\tilde{N}']) = N'_j$. We also have that $f(C_2[\tilde{N}']) = N'_j$, so it must hold that $C_3[\tilde{N}'] = C_2[\tilde{N}']$. The induction hypothesis gives that condition 2 in the definition of $\approx'_{ss}$ holds for cores with parse trees of height h or less, and thus Lemma 2 (p. 46) applies for terms with parse trees of height h or less. Hence the lemma applies to the present equality, and we may conclude that $C_3[\tilde{N}] = C_2[\tilde{N}]$. This in turn establishes that $C[\tilde{N}] = N_j$, which completes the inductive proof of condition 2.

Condition 3. We must show that $C_1[\tilde{N}] >_r C_2[\tilde{N}] \Leftrightarrow C_1[\tilde{N}'] >_r C_2[\tilde{N}']$. The direction from left to right is immediate from $3b$. For the other direction suppose that $C_1[\tilde{N}'] >_r C_2[\tilde{N}']$. In order to keep the notation simple, we just show this result for a specific rewrite rule and a specific context. We choose a rewrite rule which is sufficiently complex to exhibit the essence of the proof, namely the extended public key encryption rule in $\mathcal{E}_{pub2}-$ it should be clear as we proceed that the arguments can be lifted to the general case:

$$\text{Rewrite rule: } dec(enc(z_1, f(z_2{}^+, z_3{}^+)), f(z_2^-, z_3^-) >_r z_1$$
$$\text{Context } C_1[\tilde{N}'] = dec(enc(\perp, y_1), f(g(y_2)^-, y_3)$$

Now, in this example we would have that

$$dec(enc(\perp, N'_1), f(g(N'_2)^-, N'_3) > \perp$$

and hence there is a $T' \in \mathcal{S}(\varphi')$ (namely $T' = g(N'_2)^-$) such that

$$dec(enc(\perp, N'_1), f(T', N'_3) > \perp$$

The relevant context, namely

$$dec(enc(\perp, y_1), f(y_2^*, y_3)$$

is a minimised context and hence condition $3c$ applies. From contraposition of $3c$ it follows that there is a $T \in \mathcal{S}(\varphi)$ such that

$$dec(enc(\perp, N_1), f(T, N_3) > \perp$$

Since any synthesis term can be written as a context over cores, there is a context $C[\tilde{y}]$ such that $T = C[\tilde{N}]$, i.e.

$$dec(enc(\perp, N_1), f(C[\tilde{N}], N_3) > \perp$$

By condition *3b* it also holds that

$$dec(enc(\perp, N_1'), f(C[\tilde{N'}], N_3') > \perp$$

This implies that $C[\tilde{N'}] = g(N_2')^-$ because y_2^* is strongly correlated to y_1 (in general, Lemma 9 on p. 53 gives that minimised variables will always be strongly correlated to some variable akin to y_1). Hence also $C[\tilde{N}] = g(N_2)^-$ by Lemma 2 (p. 46). This concludes the proof, since we now get that

$$dec(enc(\perp, N_1), f(g(N_2)^-, N_3) > \perp$$

$\square$

5.4.2 Encoding Revelation

Recall that we defined $M \succ_{\mathcal{A}(\varphi)} M|_w$ to express that there are terms $T_1, \ldots, T_k \in \mathcal{A}(\varphi)$ and a destructor context $D[y, y_1, \ldots, y_k]$ such that $D[M, T_1, \ldots, T_k] >_r^M M|_w$. Observe that there are always infinitely many potential destructor contexts which can be used for a given revelation, again because it always holds that $D[y, y_1, \ldots, y_k]\square$. In the theory of public key encryption, the following is an example of an infinite sequence of destructor context:

$$\begin{aligned} dec(y_1, y_2) \\ dec(y_1, y_2{}^+) \\ dec(y_1, f(y_2)^+) \\ dec(y_1, f(f(y_2))^+) \\ \ldots \end{aligned}$$

As in the previous subsection we rely on minimised contexts to avoid infinite conjunctions. The key observation is the following: whenever a revelation from M is possible by $D[M, \tilde{T}] > M|_w$ where $\tilde{T} \subseteq \mathcal{A}(\varphi)$, the same revelation is also possible using some minimised context $D^*[y, \tilde{y^*}]$ and terms from the synthesis, i.e. $D^*[M, \tilde{T^*}] > M|_w$ where $\tilde{T^*} \subset \mathcal{S}(\varphi)$. This is an immediate consequence of the definition of minimised contexts and the fact that any context over analysis terms is a synthesis term. We illustrate this fact with an example.

Example 5.4.1. Consider a frame φ and a term $M = enc(a, k^+)$ in the usual public key theory $\mathcal{E}_{pub}$ and suppose that $T = k \in \mathcal{A}(\varphi)$.Then $M_1 \succ_{\mathcal{A}(\varphi)} a$ because the rewrite $dec(M, T^+) > a$ is possible. The relevant context, namely $dec(y_1, y_2^-)$, can be minimised to obtain $dec(y_1, y_2^*)$. The term $T^* = k^-$ is in the synthesis of φ (because k is in the analysis), and we get the reduction $dec(M, T^*) > a$. So the revelation of a from M is possible using the minimised context and synthesis terms instead of analysis terms. $\square$

Since there finitely many minimised contexts, the following set is finite:

$$\mathcal{S}_{destruct} = \{D[y, \tilde{y^*}] \mid D[y, \tilde{y^*}] \text{ is a minimised destructor context}\}$$

We are now in a position to encode the revelation relation in our logic:

$$M \succ T \triangleq \bigvee_{D[y,y_1^*,\ldots,y_k^*]\in\mathcal{S}_{destruct}} \exists y_1^* \ldots \exists y_k^* (D[M, y_1^*, \ldots, y_k^*] > T)$$

5.4.3 Encoding Ecores

Let $\varphi \equiv (\nu\tilde{n})\{^{M_i}/_{x_i}\}_{i\in I}$ be any frame, let $ecores(\varphi) = N_1, \ldots, N_s$, let $\mathcal{R}_j[\tilde{x}]$ be the analysis recipe for each N_j and let $k_j \in \mathbb{N}$ be such that $\mathcal{R}_j[\tilde{M}] >^{k_j} N_j$. For each ecore N_j we can construct a formula $\psi_{ecore-j}(y)$ with a free variable y which says that y is the ecore N_j:

$$\begin{aligned}\psi_{ecore-j}(y) \triangleq\ & \exists y_1 \ldots \exists y_{k_j}(\mathcal{R}_j[\tilde{x}] > y_1 \wedge y_2 > y_3 \wedge \cdots \wedge y_{k_j-1} > y) \wedge \\ & [\neg\exists y_2(y \succ y_2) \vee \\ & \bigwedge_{f(y_1,\ldots,y_s)\in\Sigma^{(s)}} \forall y_1, \ldots, y_s \neg(y = f(y_1, \ldots, y_s))]\end{aligned}$$

The first part of this formula says that the recipe instance for N_j must reduce to y in exactly k_j steps. The second part is a disjunction which expresses that either y is a pure core (the first disjunct) or y is an extended core (the second disjunct).

The characteristic formulae must first of all bind each ecore N_j to some variable y_j. No "let" construct is available in the logic, but existential quantification can serve a similar purpose (this is sound since every core is a synthesis term). Below is given the first step in the construction of a characteristic formula for φ. The sub-formula $\mathcal{C}'_\varphi$ encodes each of the conditions in $\approx''_{ss}$ and will be defined in the subsequent sections.

$$\mathcal{C}_\varphi = \exists y_1 \ldots \exists y_k(\psi_{ecore-1}(y_1) \wedge \cdots \wedge \psi_{ecore-k}(y_k) \wedge \mathcal{C}'_\varphi)$$

5.4.4 Encoding The Conditions

Let $\varphi \equiv (\nu\tilde{n})\{^{M_i}/_{x_i}\}_{i\in I}$ and let $ecores(\varphi) = N_1, \ldots, N_s$. We continue the construction of the characteristic formula $\mathcal{C}_\varphi$ for φ by encoding each of the conditions in $\approx''_{ss}$ into a formulae ψ_{cond}, each of which will feature as a conjunct in $\mathcal{C}'_\varphi$. The encodings will refer to the ecores bound to the variables $\tilde{y}$ above.

Condition 1

Each M_i can be written as a context over ecores – let $C_i[\tilde{y}]$ be any such context. Condition 1 in $\approx''_{ss}$ is then expressed in the following formulae:

$$\psi_{cond-1} \triangleq \bigwedge_{i\in I} x_i = C_i[\tilde{y}]$$

Condition $2a$

Define the sets

$$\begin{aligned} \mathcal{S}_{2a} &\triangleq \{(i,j) \mid N_i = N_j\} \\ \bar{\mathcal{S}}_{2a} &\triangleq \{(i,j) \mid N_i \neq N_j\} \end{aligned}$$

The following formula encodes condition $2a$:

$$\psi_{cond-2a} \triangleq \bigwedge_{(i,j)\in\mathcal{S}_{2a}} y_i = y_j \wedge \bigwedge_{(i,j)\in\bar{\mathcal{S}}_{2a}} y_i \neq y_j$$

Condition $2b$

Define the set

$$\mathcal{S}_{2b} \triangleq \{(C[\tilde{y}], j) \mid C[\tilde{N}] = N_j\}$$

The following formula encodes condition $2b$:

$$\psi_{cond-2b} \triangleq \bigwedge_{(C[\tilde{y}],j)\in\mathcal{S}_{2b}} C[\tilde{y}] = y_j$$

Condition $2c$

Define the set

$$\mathcal{S}_{2c} \triangleq \{j \mid \text{ there is no context } C[\tilde{y}] \text{ such that } C[\tilde{N}] = N_j\}$$

The following formula encodes condition $2c$:

$$\psi_{cond-2c} \triangleq \bigwedge_{j\in\mathcal{S}_{2c}} \bigwedge_{f(z_1,\ldots,z_k)\in\Sigma^k} \forall z_1 \ldots \forall z_k (\neg f(z_1,\ldots,z_k) = y_j)$$

Condition $3a$

Define the sets

$$\begin{aligned} \mathcal{S}_{3a} &\triangleq \{(C_1^{\perp}[\tilde{y}], C_2^{\perp}[\tilde{y}]) \mid C_1[\tilde{y}] \text{ is partitioning } \wedge C_1^{\perp}[\tilde{N}] \sqsupset \wedge C_1^{\perp}[\tilde{N}] > C_2^{\perp}[\tilde{N}]\} \\ \bar{\mathcal{S}}_{3a} &\triangleq \{(C_1^{\perp}[\tilde{y}], C_2^{\perp}[\tilde{y}]) \mid C_1[\tilde{y}] \text{ is partitioning } \wedge C_1^{\perp}[\tilde{N}] \sqsupset \wedge C_1^{\perp}[\tilde{N}] \not> C_2^{\perp}[\tilde{N}]\} \end{aligned}$$

The following formula encodes condition $3a$:

$$\psi_{cond-3a} \triangleq \bigwedge_{(C_1^{\perp}[\tilde{y}],C_2^{\perp}[\tilde{y}])\in\mathcal{S}_{3a}} C_1^{\perp}[\tilde{y}] > C_2^{\perp}[\tilde{y}] \wedge \bigwedge_{(C_1^{\perp}[\tilde{y}],C_2^{\perp}[\tilde{y}])\in\bar{\mathcal{S}}_{3a}} \neg C_1^{\perp}[\tilde{y}] > C_2^{\perp}[\tilde{y}]$$

Condition $3b$

Define the set

$$\mathcal{S}_{3b} \triangleq \{(C_1^{\perp}[\tilde{y}], C_2^{\perp}[\tilde{y}]) \mid C_1[\tilde{y}] \text{ is partitioning } \wedge C_1^{\perp}[\tilde{N}]\square \wedge C_1^{\perp}[\tilde{N}] > C_2^{\perp}[\tilde{N}]\}$$

The following formula encodes condition $3b$:

$$\psi_{cond-3b} \triangleq \bigwedge_{(C_1^{\perp}[\tilde{y}], C_2^{\perp}[\tilde{y}]) \in \mathcal{S}_{3b}} C_1^{\perp}[\tilde{y}] > C_2^{\perp}[\tilde{y}]$$

Condition $3c$

Define the set

$$\begin{aligned}\mathcal{S}_{3c} \triangleq \{C_1^{\perp}[\tilde{y}, z_1, \ldots, z_t] \mid & \; C_1^{\perp}[\tilde{y}, z_1, \ldots, z_t] \text{ is a minimised partitioning context } \wedge \\ & \neg\exists T_1, \ldots, T_k \in \mathcal{S}(\varphi).C_1[\tilde{N}, T_1, \ldots, T_k] >\}\end{aligned}$$

The following formula encodes condition $3c$:

$$\psi_{cond-3c} \triangleq \bigwedge_{C_1^{\perp}[\tilde{y}, z_1, \ldots, z_t] \in \mathcal{S}_{3c}} \forall z_1 \ldots \forall z_k \forall z (\neg C_1^{\perp}[\tilde{y}, z_1, \ldots, z_k] > z)$$

5.4.5 Putting it Together

The construction of a characteristic formula $\mathcal{C}_\varphi$ was started in Subsection 5.4.3 and Subsection 5.4.4 gave an encoding of the conditions in $\approx''_{ss}$, each of which should feature as a conjunct in $\mathcal{C}_\varphi$. All that remains now is to put it all together.

Definition 31 (Characteristic Formula). *Let $\varphi \equiv (\nu\tilde{n})\{^{M_i}/_{x_i}\}_{i \in I}$ be any frame, let $ecores(\varphi) = N_1, \ldots, N_s$, let $\mathcal{R}_j[\tilde{x}]$ be the analysis recipe for each N_j and let $k_j \in \mathbb{N}$ be such that $\mathcal{R}_j[\tilde{M}] >^{k_j} N_j$. We now rely on the constructions in the previous subsections to give a definition of the characteristic formula for φ, $\mathcal{C}_\varphi$:*

$$\begin{aligned}\mathcal{C}_\varphi \triangleq \; & \exists y_1 \ldots \exists y_k (\\ & \psi_{ecore-1}(y_1) \wedge \cdots \wedge \psi_{ecore-k}(y_k) \wedge \\ & \psi_{cond-1} \wedge \\ & \psi_{cond-2a} \wedge \psi_{cond-2b} \wedge \psi_{cond-2c} \\ & \psi_{cond-3a} \wedge \psi_{cond-3b} \wedge \psi_{cond-3c})\end{aligned}$$

Theorem 5. *For any frames φ and φ' with $dom(\varphi) = dom(\varphi')$ it holds that*

$$\varphi \approx_{ss} \varphi' \Leftrightarrow \varphi' \models \mathcal{C}_\varphi$$

Proof. It is clear from the construction of $\mathcal{C}_\varphi$ that

$$\varphi \approx''_{ss} \varphi' \Leftrightarrow \varphi' \vDash \mathcal{C}_\varphi$$

By Lemma 18, $\approx''_{ss}$ and $\approx'_{ss}$ coincide, and by Theorem 1 and Theorem 2, $\approx'_{ss}$ and $\approx_{ss}$ coincide. □

5.5 Summary

We have given a first order logic for frames, $\mathcal{LF}$, with atomic propositions for asserting equality, syntactic equality and reductions, which enable direct reasoning about the terms contained in a frame. Quantification ranges over the synthesis of a frame which intuitively is the set of terms that can be constructed by applying function symbols to ecores. Quantification hence enables indirect reasoning about the knowledge represented by a frame, which is essential for expressing e.g. that there is *some* term y_1 (known to the environment) that can be used to decrypt x_1 and obtain *some* analysis term y_2.

It has been shown that $\mathcal{LF}$ characterises strong static equivalence. Furthermore, we have given a construction of characteristic formulae. This construction relies on a further refinement of strong static equivalence which does not involve any universal quantification over contexts. The existence of characteristic formulae is an important result which shows the expressive power of the logic: $\mathcal{LF}$ is sufficiently powerful to describe all the essential features of a frame (with respect to static equivalence) in a single finite formula.

Chapter 6

A Logic for Applied π

In the previous chapter a logic for frames was developed. It was shown that the logic characterises static equivalence on frames and a construction of characteristic formulae was given. In this chapter we build upon the logic for frames and extend it with Hennessy-Milner modalities, yielding a logic for Applied π processes which characterises labelled bisimilarity. The resulting logic can be used to reason about the dynamic behaviour of a process as well as the static knowledge represented by the frame of the process, allowing assertions such as *process P can perform an input of an encrypted term M_1 followed by some output, after which the decryption key for the term M_1 is known by the environment* to be expressed.

Section 1 presents the syntax of the Applied π logic and Section 2 gives the semantics in terms of a satisfaction relation. The choice of syntax and semantics is to some extent predetermined by our intention for the logic to characterise labelled bisimilarity, so we will be brief with the presentation. Section 3 discusses two additional logical construct which can be defined from the basic logic, namely match and the matching input modality. Finally section 4 contains a proof that the logic characterises labelled bisimilarity; this relies on the results from the previous chapter which say that the underlying logic of frames characterises strong static equivalence.

6.1 Syntax

The definition of labelled bisimilarity between processes includes conditions on internal reduction, output transitions, bound output transitions and input transitions. The syntax of the logic $\mathcal{LA}$ for Applied π must therefore include modalities for each of these transitions in addition to the propositional connectives and the constructs from the logic of frames:

$$
\begin{aligned}
A ::= & M_1 =_{\mathcal{E}} M_2 \mid M_1 > M_2 \mid M_1 = M_2 \mid \exists x(A_1) \\
& \mid \neg A_1 \mid A_1 \vee A_2 \mid (A_1) \\
& \mid \langle \tau \rangle A_1 \\
& \mid \langle \bar{a} \rangle A_1 \\
& \mid \langle \nu \bar{a} \rangle A_1 \\
& \mid \langle a(u) \rangle A_1
\end{aligned}
$$

For example, the modal formula $\langle a(x) \rangle A_1$ expresses that a process can do an input on channel a, bound to the variable x, after which the process satisfies A_1. The full range of propositional and first order connectives can be defined in the same manner as for $\mathcal{LF}$. Similarly, the dual modalities (necessity) can be defined as usual; for example the dual modality of $\langle \tau \rangle$ is defined thus:

$$[\tau] A_1 \quad \triangleq \quad \neg \langle \tau \rangle \neg A_1$$

Note that constructs from the logic of frames may be freely mixed with the modalities. This opens up the possibility to reason about environment knowledge over time. Take for example the following formula:

$$\exists x_1 (\langle \bar{a} \rangle \langle a(y) \rangle \exists x_2 (dec(y, x_1) > x_2))$$

This formula expresses that a decryption key x_1 for the term which is input at some later stage is known up front; more specifically, it says that there is a known decryption key x_1 such that, after outputting some arbitrary term and inputting y, y can be decrypted with x_1 to obtain some x_2. A bigger example will be given in the next chapter where the logic is applied to reasoning about a security protocol.

6.2 Semantics

Semantics for the modalities is guided by our intention for $\mathcal{LA}$ to characterise labelled bisimilarity. Since this is a weak bisimilarity, the modalities must also be weak, i.e. input and output actions may be preceded and followed by any number of internal actions. Let $\vDash_{\mathcal{LF}}$ be the satisfaction relation for $\mathcal{LF}$. The satisfaction relation for $\mathcal{LA}$ can then be defined as follows:

$$
\begin{array}{lll}
P \vDash M_1 =_{\mathcal{E}} M_2 & \triangleq & \varphi(P) \vDash_{\mathcal{LF}} M_1 =_{\mathcal{E}} M_2 \\
P \vDash M_1 = M_2 & \triangleq & \varphi(P) \vDash_{\mathcal{LF}} M_1 = M_2 \\
P \vDash M_1 > M_2 & \triangleq & \varphi(P) \vDash_{\mathcal{LF}} M_1 > M_2 \\
P \vDash \exists x(A_1) & \triangleq & \text{there exists } M \in \mathcal{S}(\varphi(P)) \text{ s.t. } (P \mid \{{}^{M}/_{x}\}) \vDash A_1 \\
P \vDash \neg A_1 & \triangleq & P \nvDash A_1 \\
P \vDash A_1 \vee A_2 & \triangleq & P \vDash A_1 \text{ or } P \vDash A_2 \\
P \vDash (A_1) & \triangleq & P \vDash A_1 \\
P \vDash \langle\tau\rangle A_1 & \triangleq & \text{there exists } P' \text{ s.t. } P \xrightarrow{\tau}{}^{*} P' \text{ and } P' \vDash A_1 \\
P \vDash \langle\bar{a}\rangle A_1 & \triangleq & \text{there exists } u, P' \text{ s.t. } P \xrightarrow{\tau}{}^{*} \xrightarrow{\overline{a}\langle u\rangle} \xrightarrow{\tau}{}^{*} P' \text{ and } P' \vDash A_1 \\
P \vDash \langle\nu\bar{a}\rangle A_1 & \triangleq & \text{there exists } u, P' \text{ s.t. } P \xrightarrow{\tau}{}^{*} \xrightarrow{(\nu u)\overline{a}\langle u\rangle} \xrightarrow{\tau}{}^{*} P' \text{ and } P' \vDash A_1 \\
P \vDash \langle a(x)\rangle A_1 & \triangleq & \text{there exists } M, P' \text{ s.t. } P \xrightarrow{\tau}{}^{*} \xrightarrow{a(M)} \xrightarrow{\tau}{}^{*} P' \text{ and } P' \vDash A_1\{{}^{M}/_{x}\}
\end{array}
$$

6.3 Match and Matching Input Modality

Syntax and semantics for the Applied π logic have now been given. In this section we consider two additional operators which are not part of the basic logic. The first is match ($[M_1 =_{\mathcal{E}} M_2]A$) which asserts that whenever the match predicate holds, the formula A must be true. The second is the matching input modality ($\langle a({}^{M}/_{x})\rangle A$) which requires that M be input on a; this is in contrast to the modality $\langle a(x)\rangle A$ where any term can be input on a.

We discover that a naive adoption of the match operator from the π calculus logics [23] and the Spi logics [13] yields a logic which does not characterise labelled bisimilarity. A weaker match operator can however be defined from the equality predicate in the underlying frames logic, and this is also the key to defining the matching input modality from the existing input modality.

6.3.1 The Problem with Match

As in the π calculus, Applied π includes the process construct

$$\text{if } M_1 =_{\mathcal{E}} M_2 \text{ then } P \text{ else } Q$$

for testing equivalence between two terms and, based on the outcome, continuing either as process P or Q. One might therefore expect that a match operator $[M_1 =_{\mathcal{E}} M_2]A$ must be included in the logic in order for the logic to characterise labelled bisimilarity. Since tests on arbitrary terms (possibly containing private names) is allowed in the calculus, one might further expect that the match operator in the logic can take arbitrary terms and that satisfaction could then be defined as follows:

$$P \models [M_1 =_{\mathcal{E}} M_2]A \stackrel{\Delta}{=} \text{ if } M_1\varphi =_{\mathcal{E}} M_2\varphi \text{ then } P \models A$$

Note how this definition does *not* rely on equality in frames (i.e. we just require that $M_1\varphi =_{\mathcal{E}} M_2\varphi$ instead of $(M_1 =_{\mathcal{E}} M_2)\varphi)$, reflecting the intuition that an equality may hold even when M_1 and M_2 contain private names as in the conditional process construct. As we shall see in the next chapter, this definition of the match operator would be useful in applications. However, the resulting logic would not characterise labelled bisimilarity. To illustrate this fact consider the following two processes which are vacuously labelled bisimilar because they are statically equivalent and have no dynamic behaviour:

$$P \stackrel{\Delta}{=} (\nu k)(\{^{k}/_{x}\})$$
$$Q \stackrel{\Delta}{=} (\nu k)(\{^{f(k)}/_{x}\})$$

Take the match formula $A \stackrel{\Delta}{=} \neg[x =_{\mathcal{E}} k]\mathbf{false}$ which expresses that $x =_{\mathcal{E}} k$. Then $P \models A$ while $Q \not\models A$, which serves to show that a logic with arbitrary match does not characterise labelled bisimilarity. The reason is, intuitively, that even though processes have the power to test for this kind of equalities, they might not do so – and hence their behaviour may be independent of the outcome of the test.

The Spi logic includes a match predicate similar to the one above, and an example similar to the one above suggests that the Spi logic does *not* characterise the relevant bisimulation. Hence it would seem that the Spi characterisation theorem [13, Theorem 1 p. 8] does not hold; the proof of this theorem, given in [14], does not resolve the doubt since the case for match is omitted.

What is needed is a weaker test operator which cannot be used to distinguish statically equivalent processes, and the natural choice is to define the match operator in terms of *equality in frames*:

$$P \models [M_1 =_{\mathcal{E}} M_2]A \stackrel{\Delta}{=} \text{ if } (M_1 =_{\mathcal{E}} M_2)\varphi \text{ then } P \models A$$

However this operator is redundant since the test proposition $M_1 =_{\mathcal{E}} M_2$ is already part of the logic. We can then use this together with the propositional connectives to give an equivalent definition of the above weaker version of the match operator, thus:

$$P \models [M_1 =_{\mathcal{E}} M_2]A \stackrel{\Delta}{=} (M_1 =_{\mathcal{E}} M_2 \to A)$$

Similar test operators can be defined for syntactic equality and for reductions.

6.3.2 Matching Input Modality

The basic input modality $\langle a(x)\rangle A$ asserts that a process can afford a transition labelled $a(M)$ for *some* term M to a derivative which satisfies $A\{^{M}/_{x}\}$. We also wish to consider a stronger *matching input* modality, $\langle a({}^{M}\!/_{x})\rangle A$, which asserts that a process can afford the transitions labelled $a(M)$ for *exactly* the term M, to a derivative which satisfies $A\{{}^{M}\!/_{x}\}$. Matching input modality is strictly necessary to prove that the logic characterises labelled bisimilarity in the next section. Satisfaction can be defined as follows:

$$P \vDash \langle a({}^{M}\!/_{x})\rangle A \quad \triangleq \quad \text{there exists } P' \text{ s.t. } P \xrightarrow{\tau}{}^{*} \xrightarrow{a(M)} \xrightarrow{\tau}{}^{*} P' \text{ and } P' \vDash A\{{}^{M}\!/_{x}\}$$

As with the match operator, it turns out that the matching input modality is redundant; it can be defined from the standard input modality together with syntactic equality, thus:

$$\langle a({}^{M}\!/_{x})\rangle A \ \triangleq \ \langle a(x)\rangle(x = M \wedge A)$$

Informally, this definition says that a process satisfies $\langle a({}^{M}\!/_{x})\rangle A$ if it can input *some* term bound to variable x *and* if this term is syntactically equal to M. This seems sensible; however there is a catch, for the semantics for $x = M$ is defined in terms of syntactic equality in frames, i.e. a process P satisfies $x = M$ if and only if $(x = M)\varphi(P)$. For this to be true, it must first of all hold that $n(M) \cap bn(\varphi(P)) = \emptyset$. The following lemma establishes that this is indeed the case for any input M, and it follows immediately that the above definition of the matching input modality in terms of the standard input modality and syntactic equality captures the intended meaning.

Lemma 19. *For any transition $P \xrightarrow{a(M)} P'$ where M occurs free (i.e. with no names under restriction) in some sub-process of P' it holds that $n(M) \subseteq fn(\varphi(P'))$.*

Proof. By transition induction, i.e. induction in the height of the derivation tree of $P \xrightarrow{a(M)} P'$ with case distinction on the last rule applied.

Case: transition concluded by In . Then $P = a(x).P'$ and by definition of (extended) processes no active substitution can occur within the scope of an input. Hence $\varphi(P') = (\nu\emptyset)\emptyset$ so contains no private names, and the result holds vacuously.

Case: transition concluded by Scope . Then $P = (\nu n)Q$, $Q \xrightarrow{a(M)} Q'$ and $P' = (\nu n)Q'$. The induction hypothesis gives that $n(M) \cap bn(\varphi(Q')) = \emptyset$ and the side condition in the Scope rule gives that n does not occur in M. Since the frame of P is just the frame of Q with an additional restriction on n, we have that also $n(M) \cap bn(\varphi(Q')) = \emptyset$ and hence that $n(M) \subseteq fn(\varphi(Q'))$

Case: transition concluded by Par . Then we have that $P = Q \mid Q'$, $Q \xrightarrow{a(M)} Q''$ and $P' = Q'' \mid Q'$. Now recall how to obtain the frame of P': first obtain $\varphi(Q'') = \tilde{n}_1\sigma_1$ and $\varphi(Q') = \tilde{n}_2\sigma_2$ such that $\tilde{n}_1 \cap n(\sigma_2) = \tilde{n}_2 \cap n(\sigma_1) = \emptyset$ by α-converting as necessary. By induction hypothesis $n(M) \cap \tilde{n}_1 = \emptyset$. Per assumption we also have that $n(M) \subseteq n(Q'')$ and therefore $n(M) \cap \tilde{n}_2 = \emptyset$. Hence $n(M) \cap (\tilde{n}_1 \cup \tilde{n}_2) = \emptyset$ giving that $n(M) \cap bn(\varphi(P')) = \emptyset$. Since M does occur in P' we conclude that $n(M) \subseteq fn(\varphi(P'))$.

Case: transition concluded by STRUCT . Then $P \equiv Q$, $Q \xrightarrow{a(M)} Q'$ and $Q' \equiv P'$. Since M occurs free in a subcomponent of P it follows by inspection of the rules for structural congruence that M also occurs free as a subcomponent of Q. By induction hypothesis $n(M) \subseteq fn(\varphi(Q'))$, and again by inspection of rules for structural congruence it holds that $n(M) \subseteq fn(\varphi(P'))$. $\square$

Note that it does *not* necessarily hold that the names of an input are free in the derived *process*. Here is an example:

$$P \triangleq (\nu k)(\{^{k}/_{x}\}) \mid a(y).\overline{b}\langle y\rangle.\mathbf{0}$$

Then P affords the input transition $P \xrightarrow{a(k)} P'$ where

$$P' = (\nu k)(\{^{k}/_{x}\}) \mid \overline{b}\langle k\rangle.\mathbf{0}$$

and hence $k \in bn(P')$. However, the frame of the derivative P' above is obtained by α-converting the bound k thus:

$$\varphi(P') = (\nu c)(\{^{c}/_{x}\} \mid \overline{b}\langle k\rangle.\mathbf{0})$$

and then we have that $k \notin bn(\varphi(P'))$. Hence it is strictly necessary to speak only about the *frame* of the derived process in Lemma 19.

We have now introduced the matching input modality and shown how this can be defined from the basic input modality and equality. As a final remark we note that a *matching output* modality $\langle\overline{a}\langle M\rangle\rangle$ would result in a logic which does not characterise labelled bisimilarity. Because for P and Q to be labelled bisimilar, the output condition requires only that whenever P can perform an output action $\overline{a}\langle x\rangle$, then Q can also perform this output action – there is no requirement for x to be bound to the same term in the two processes.

6.4 Characterisation

In this section we show the expected result that $\mathcal{LA}$ characterises labelled bisimilarity. Existential quantification can now have any formulae in its scope (and not just $\mathcal{LF}$ formulae), so we must start by proving a new extension lemma saying that labelled bisimilarity between processes is preserved by extension with certain active substitutions.

Lemma 20 (Process extension lemma). *Let P and Q be two processes with $P \approx_l Q$. Let $ecores(\varphi(P)) = (N)_{j\in J}$ and $ecores(\varphi(P)) = (N')_{j\in J}$. Then for any context $C[\tilde{y}]$ and any variable x fresh in P and Q it holds that*

$$P \mid \{^{C[\tilde{N}]}/_{x}\} \approx_l Q \mid \{^{C[\tilde{N}']}/_{x}\}$$

Proof. Suppose that $P \approx_l Q$. By condition 1 of labelled bisimilarity we have that $\varphi(P) \approx_{ss} \varphi(Q)$. By the second extension lemma for frames (Lemma 17) it also holds that

$$\varphi(P)\{^{C[\tilde{N}]}/_{x}\} \approx_{ss} \varphi(Q)\{^{C[\tilde{N}']}/_{x}\}$$

which is equivalent to

$$\varphi(P \mid \{^{C[\tilde{N}]}/_{x}\}) \approx_{ss} \varphi(Q \mid \{^{C[\tilde{N}']}/_{x}\})$$

This establishes condition 1 of labelled bisimilarity for the extended processes.

For the other two conditions it suffices to show that P can perform exactly the same actions as the extended process $P \mid \{^{C[\tilde{N}]}/_{x}\}$. This amounts to showing that any action which P can perform can also be performed by the extended process, and conversely that any action that the extended process can perform can also be performed by P.

Suppose that P can perform some action. If the action is an internal reduction then the extended process can perform the same action since internal reduction is closed by application of evaluation contexts. If P can perform a labelled action, the PAR rule gives that the extended process can perform the same action (here we use that the variable x does not occur anywhere in P, so the premise of the rule is satisfied).

For the converse suppose that the extended process can perform some action. If the action is an internal reduction, this can only be done by process P since active substitutions have neither match or communication capabilities. Note that a reduction using the THEN or ELSE rules is independent of the active substitution since the variable x per assumption does not occur in the match predicate. If the extended process can perform a labelled action, then this action can only be concluded using the PAR rule where the process P performs the same action. □

We are now ready to show the main characterisation result. The proof is fairly standard and follows the structure in e.g. [23, Proof of Theorem 1, p. 24]. As is common for this kind of characterisation theorem we assume that processes are image finite: it must hold for any process P that the set $\{P' \mid P \xrightarrow{\alpha} P'\}$ is finite. In the parts of the proof concerning the frame logic we rely on the above extension lemma together with the results from the previous chapter saying that the logic of frames characterises strong static equivalence.

Theorem 6. *For image-finite processes, $\mathcal{LA}$ characterises labelled bisimilarity (based on strong static equivalence):*

$$P \approx_l Q \Leftrightarrow Th_{\mathcal{LA}}(P) = Th_{\mathcal{LA}}(Q)$$

Proof. There are two directions to show.

($\Rightarrow$) Suppose that $P \approx_l Q$ and $P \vDash A$. We show by structural induction in A that also $Q \vDash A$. We only show the base cases, the step for existential quantification, and the step for the input modalities; the remaining cases are similar or standard.

Basis.

Case: $A = M_1 \circ M_2$ where $\circ$ is either $=_{\mathcal{E}}$, $=$ or $>$. Per definition of satisfaction, $\varphi(P) \vDash_{\mathcal{LF}} M_1 \circ M_2$. From the assumption that $P \approx_l Q$ and condition 1 of labelled bisimilarity, we have that $\varphi(P) \approx_{ss} \varphi(Q)$. By Theorem 4 also $\varphi(Q) \vDash_{\mathcal{LF}} M_1 \circ M_2$, i.e. $Q \vDash M_1 \circ M_2$.

Step.

Case: $A = \exists x(A_1)$. Per definition of satisfaction there is a $M \in \mathcal{S}(\varphi(P))$ such that $(P \mid \{^M/_x\}) \vDash A_1$. We then proceed as in the corresponding case in the proof of Theorem 4. Any synthesis term can be written as a context over cores, i.e. $M = C[\tilde{N}]$ for some context $C[\tilde{y}]$. From the assumption that $P \approx_l Q$ and the extension lemma for processes (Lemma 20), we have that also $P \mid \{^M/_x\} \approx_l Q \mid \{^{M'}/_x\}$ where $M' = C[\tilde{N'}]$. Therefore the induction hypothesis gives that $Q \mid \{^{M'}/_x\} \vDash A_1$, and since $M' \in \mathcal{S}(\varphi(Q))$ we conclude from the definition of satisfaction that $Q \vDash \exists x(A_1)$.

Case: $A = \langle a(x)\rangle A_1$**.** Per definition of satisfaction there exists a term M, and processes P',P'' and P''' such that

$$P \xrightarrow{\tau}{}^{*} P' \xrightarrow{a(M)} P'' \xrightarrow{\tau}{}^{*} P''' \text{ and } P''' \vDash A_1\{^M/_x\}$$

From the assumption that $P \approx_l Q$ and by inductively using condition 2 in the definition of labelled bisimilarity, there is a process Q' such that $Q \xrightarrow{\tau}{}^{*} Q'$ and $P' \approx_l Q'$. By condition 3 there is a process Q'' such that $Q' \xrightarrow{a(M)} Q''$ with $P'' \approx_l Q''$. Using condition 2 inductively again we conclude that there is a process Q''' such that

$$Q \xrightarrow{\tau}{}^{*} Q' \xrightarrow{a(M)} Q'' \xrightarrow{\tau}{}^{*} Q'''$$

where $P''' \approx_l Q'''$. The induction hypothesis gives that $Q''' \vDash A_1\{^M/_x\}$, and hence we conclude from the definition of satisfaction that $Q \vDash \langle a(x)\rangle A_1$.

($\Leftarrow$) For the other direction suppose that $Th_{\mathcal{LA}}(P) = Th_{\mathcal{LA}}(Q)$, i.e. P and Q satisfy the same formulae. We must show that P and Q are related by some labelled bisimulation. Define the relation $\mathcal{L}$ where $P\mathcal{L}Q$ if and only if $Th_{\mathcal{LA}}(P) = Th_{\mathcal{LA}}(Q)$. Then clearly $P\mathcal{L}Q$; we show that $\mathcal{L}$ is a labelled bisimulation, i.e. that it satisfies the three conditions in the definition of $\approx_l$.

P and Q satisfy the same formulae, and in particular they satisfy the same frame formulae. It follows by Theorem 4 that $\varphi(P) \approx_{ss} \varphi(Q)$ which establishes condition 1 in the definition of labelled bisimilarity.

Next we show condition 3 for the case where $\alpha = a(M)$. So suppose that $P \xrightarrow{a(M)} P'$ and let $(Q)_{i\in I}$ be an enumeration of the set $\{Q' \mid Q \xrightarrow{a(M)} Q'\}$. Suppose for a contradiction that Q cannot simulate P, i.e. that $(P', Q_i) \notin \mathcal{L}$ for all $i \in I$. Then P' and Q_i do not satisfy the same formulae (this follows by contraposition of the direction proven above), i.e. for each $i \in I$ there is a formulae $A_i \in Th_{\mathcal{LA}}(P') - Th_{\mathcal{LA}}(Q_i)$. Per assumption of image-finiteness the enumeration $(Q)_{i\in I}$ is finite, so the formula $A = \langle a(^M/_x)\rangle \bigwedge_{i\in I} A_i$ is well defined (if $I = \emptyset$ we define $\bigwedge_{i \in I} A_i$ to be **true**). Then by construction $A \in Th_{\mathcal{LA}}(P) - Th_{\mathcal{LA}}(Q)$, contradicting that $P \approx_l Q$.

The proofs for the other labels in condition 3, and the proof for condition 2, are similar. □

Note how the input modality $\langle a(^{M}/_{x})\rangle$, which was defined from $\langle a(x)\rangle$ and equality, plays an essential role in the proof of the input case in condition 3. For if we only had $\langle a(x)\rangle$ available we could conclude only that Q can match P's transition with *some* input on a, but not necessarily with an input of M on a; this would not be strong enough to prove bisimilarity.

6.5 Summary

We have extended the logic of frames with four modalities, namely an internal action modality, two output modalities and an input modality. An additional input matching input modality has been introduced and shown to be definable from the basic logic. We have also argued that a generic match operator (where private names may occur in terms) would result in the logic not characterising labelled bisimilarity, but a weaker match operator can be defined from the equality proposition inherited from the frame logic.

The result is a logic $\mathcal{LA}$ for Applied π which characterises labelled bisimilarity. The logic allows intermixing frame formulae from $\mathcal{LF}$ (including existential quantification) with modalities, which opens up the possibility of reasoning about knowledge over time.

Chapter 7

An Application to the Needham-Schroeder Public Key Protocol

In this chapter we demonstrate how the logic for Applied π, based on the logic of frames, can be used to reason about a security protocol. The Needham-Schroeder Public Key Protocol [24] has been chosen for this purpose because its specification in Applied π is fairly simple and because it affords a well known attack [20] which can be expressed in the logic.

We start by explaining the Needham-Schroeder Public Key Protocol and a known attack in Section 1, after which we present an Applied π specification of the protocol in Section 2. In Section 3 we show how to express security properties in the logic, including a property which captures the attack on the protocol.

7.1 The Needham-Schroeder Public Key Protocol

The basic Needham-Schroeder Public Key Protocol is presented in the first subsection followed by an explanation of the attack in the second subsection.

7.1.1 The Protocol

The Needham-Schroeder Public Key Protocol can be used to authenticate two principals A and B to each other using public key cryptography, where we let A^+ and B^+ denote the public keys of A and B respectively. The protocol also employs nonces N_a and N_b which are random messages, generated by A and B respectively, intended to be used only in a single run of the protocol.

The complete Needham-Schroeder Public Key Protocol includes steps for obtaining public keys from a trusted key server. For simplicity we omit these steps and assume that the public keys are already available to the participating principals; following [20], the core protocol can then be described informally thus:

$$\begin{array}{lll}\text{Message 1} & A \rightarrow B: & A, B, \{N_a, A\}_{B+} \\ \text{Message 2} & B \rightarrow A: & B, A, \{N_a, N_b\}_{A+} \\ \text{Message 3} & A \rightarrow B: & A, B, \{N_b\}_{B+}\end{array}$$

Principal A initiates the protocol by sending a message to B in step 1. The first two parts of the message simply express the fact that A wishes to establish a session with B. The third part of the message contains the nonce N_a and the identity A encrypted under B's public key. Upon receiving this message, B uses its private key for decryption and obtains the nonce N_a. B then returns this nonce, together with a freshly generated nonce N_b, encrypted under A's public key. When A receives this message, A believes that the message must originate from B because only B would be able to decrypt the contents of the first message to obtain the nonce N_a. Hence A happily returns the nonce N_b to B, again encrypted with B's public key, in the belief that the nonce is a shared secret between the two principals. Similarly, B believes that the messages originated from A and that N_b is their shared secret, since only A should be able to decrypt the second message.

Following the three steps outlined above, A and B could proceed to generate a secret symmetric encryption key from the nonce N_b, which could be used for encryption in the remainder of the session. Symmetric key cryptography offers the advantage of speed compared to public key cryptography, so after the initial authentication using public keys, symmetric keys are employed in many practical applications such as the Secure Shell (ssh) program.

7.1.2 An Attack

An attack on the protocol which compromises the secrecy of the nonce N_b (and potentially subsequent symmetric encryption keys) is revealed in [20]. This is a classical example of how formal methods can be applied to reveal weaknesses in a protocol which for many years was believed to be secure. The attack involves an intruder I acting as a "man in the middle": If A initiates a session with I, I can establish a new session with B impersonating A. B will then believe that it is carrying out a session with A instead of I. Hence the attack involves interleaving two runs of the protocol; we denote these two runs by α and β and write e.g. $\alpha.2$ to refer to step two in run α. We also write $I(A)$ to indicate that I is impersonating A. The attack on the protocol can then be described as follows:

Message $\alpha.1$	$A \rightarrow I:$	$A, I, \{N_a, A\}_{I^+}$
Message $\beta.1$	$I(A) \rightarrow B:$	$A, B, \{N_a, A\}_{B^+}$
Message $\beta.2$	$B \rightarrow I(A):$	$B, A, \{N_a, N_b\}_{A^+}$
Message $\alpha.2$	$I \rightarrow A:$	$I, A, \{N_a, N_b\}_{A^+}$
Message $\alpha.3$	$A \rightarrow I:$	$A, I, \{N_b\}_{I^+}$
Message $\beta.3$	$I(A) \rightarrow B:$	$A, B, \{N_b\}_{B^+}$

In step $\beta.2$ the intruder receives the nonce N_b, but it is encrypted under A's public key and hence I cannot decrypt it. Instead the intruder uses A as an oracle by replaying this message in $\alpha.2$, and in message $\alpha.3$ I receives the nonce N_b encrypted under I's own public key; it can thus use its private key to obtain N_b. Note that B still believes that it has successfully carried out the authentication session with A and that N_b is a shared secret between A and B.

The protocol can be fixed simply by including the identity of the responder B in the encrypted part of message 2. A can then test if the message in step 3 originates from the expected principal.

7.2 The Specification

In this section we model the Needham-Schroeder Public Key Protocol in Applied π. We start in Subsection 1 by outlining some general assumptions, known as the Dolev-Yao Model. In Subsection 2 we present the equational theory used in the specification, which is based on public key encryption and lists. Subsection 3 discusses how to represent public keys in the specification and how these are used as process identifiers. Subsection 4 introduces some notational short hand which is useful in the specification, namely agent identifiers and output-guarded nondeterministic choice. Finally the complete specification is presented in Subsection 5, and we conclude with a transition trace leading to the attack in Subsection 6.

7.2.1 The Dolev-Yao Model

Applications of formal methods to security protocol often rely on what is known as the Dolev-Yao model [12]. The Dolev-Yao model is based on the following two assumptions:

- An intruder has complete knowledge of and access to communication channels in the sense that he/she can send arbitrary messages, can intercept any message, and modify it any way he/she sees fit.
- Cryptographic functions are perfect: decryption is only possible with the appropriate decryption key.

In Applied π the first assumption is realised through the labelled semantics and active substitutions: any output will result in a new active substitution which contributes to the frame of the process, and the frame is thus a representation of the environment knowledge which is a available to an attacker. The environment (or attacker) may compose any message by analysing and synthesising the frame of a process and send the message back to the process using labelled input.

The second assumption is realised through the algebraic specification of cryptographic functions by rewrite rules generating an equational theory. In a theory with symmetric key encryption, the equality $dec(enc(x_1, x_2), x_3) =_{\mathcal{E}} a$ holds if and *only if* $x_2 = x_3$.

7.2.2 The Equational Theory

From the informal presentation of the protocol it is clear that an equational theory with public key encryption is required. We also need means of sending multiple terms in one message, and for this purpose an equational theory with lists is convenient. Hence we arrive at the following definition:

Definition 32. *The signature and equational theory used for the subsequent specification is defined below.*

Theory $\mathcal{E}_{NS}$.
Signature: $enc(\cdot,\cdot)$, $dec(\cdot,\cdot)$, $\cdot^{+}$, $\cdot^{-}$,$cons(\cdot,\cdot)$,$head(\cdot)$,$tail(\cdot)$, *NIL.*
Rewrite System:

$$
\begin{aligned}
dec(enc(z_1, {z_2}^{+}), z_2^{-}) &>_{r_1} z_1\\
head(cons(z_1, z_2)) &>_{r_2} z_1\\
tail(cons(z_1, z_2)) &>_{r_3} z_2
\end{aligned}
$$

The part of $\mathcal{E}_{NS}$ concerning public key encryption has been explained earlier in the report. The function symbol $cons(z_1, z_2)$ is the list constructor: it takes a head z_1 and a tail z_2 (which itself is a list) to obtain a new list. The function symbols *head* and *tail* are used to extract the head and tail from a list, respectively, as expressed in the rewrite rules r_2 and r_3. Finally, the constant function symbol *NIL* represents the empty list.

For the present purpose we shall only be needing lists with two or three elements. Therefore we define the following notational short hands.

Definition 33. *Pair construction, triple construction, and projection of first,* *second and third elements, are defined as follows:*

$$
\begin{aligned}
[z_1, z_2] &\triangleq cons(z_1, cons(z_2, NIL)) \\
[z_1, z_2, z_3] &\triangleq cons(z_1, [z_2, z_3]) \\
fst(z) &\triangleq head(z) \\
snd(z) &\triangleq head(tail(z)) \\
trd(z) &\triangleq head(tail(tail z))
\end{aligned}
$$

7.2.3 Process Identifiers and Public Keys

In the formal specification of the protocol in Applied π, principals are represented by processes. The informal protocol description requires that principals have identifiers which can be sent along in messages, e.g. in message 1 which includes the information that the message is from A to B. There is however no immediate way of identifying processes in Applied π (replication would pose problems for such identifiers). Instead we adopt the convention that principals are identified by their public keys.

As explained earlier we assume that public keys have been exchanged and are known to all principals prior to protocol execution. To model this assumption we simply include in the specification an active substitution with the public key for each principal. For instance there will be an active substitution $\{^{a^+}/_{z_a}\}$ representing the public key for A. When other processes (e.g. B and potential intruders) refer to A's public key, this reference must be through the variable z_a in the relevant active substitution because the seed a is private for principal A. If the public key were to be used directly to encrypt a term, the encrypted term could not be sent to A because scope extrusion would fail.

7.2.4 Agent Definitions and Choice

When specifying the protocol in Applied π it will be convenient to employ two notions which are not explicitly part of the calculus, namely agent definitions and nondeterministic choice. Agent definition are on the form

$$
\begin{aligned}
A_1 &\triangleq P_1 \\
A_2 &\triangleq P_2 \\
&\dots
\end{aligned}
$$

where P_1 is the process we wish to identify with A_1 etc. The agent identifiers may then occur in any process, the intuition being that the identifier A_i is replaced by its definition P_i, which may lead to recursion. Although agent identifiers are not explicitly part of the calculus, they can easily be encoded using the bang operator (!) [25, Section 3.4]. Hence we can safely use agent identifiers in the protocol specification.

Nondeterministic choice, $+$, can be used to write processes on the form $P \triangleq \overline{a}\langle x\rangle.P_1 + \overline{a}\langle y\rangle.P_2$, the intuition being that P can choose nondeterministically to proceed as either $\overline{a}\langle x\rangle.P_1$ or $\overline{a}\langle y\rangle.P_2$. Nondeterministic choice cannot generally be encoded in Applied π, but many special cases can [25, Section 3.2], including the output-guarded process P above. For instance, assuming that $z \notin fv(P_1, P_2)$, the process P above is observationally equivalent to the process

$$P' \triangleq (\nu b)(\overline{b}\langle d\rangle \mid b(z).\overline{a}\langle x\rangle.P_1 \mid b(z).\overline{a}\langle y\rangle.P_2)$$

When specifying the protocol we shall use output-guarded nondeterministic choice, knowing that this can be encoded in the basic Applied π calculus.

7.2.5 The Specification

We are now ready to present the formal specification of the Needham-Schroeder Public Key Protocol in Applied π– the specification is given in Table 7.1.

The processes A and B represent the two main principals in the protocol. There is no explicit encoding of potential intruders since this aspect is implicit in processes being able to communicate with the environment. Note however the definition of process A: since A initiates the protocol, it must as a first step choose which principal to engage in a protocol run with. This initial choice is essential to include in the specification, since this is the choice leading to A contacting a potential intruder. We assume that A only has two friends which are identified by the public keys z_{b_0} and z_{b_1}. The first is intuitively the "correct" process B , identified by the public key variable z_{b_0}, while the second may be a hostile intruder and is identified by the public key variable z_{b_1}. Process A uses nondeterministic choice to decide which of these to initiate a protocol run with. The process $T(i)$ generalises the first step of the protocol, where the argument i indicates which of A's two friends should be contacted. The derivatives of A are indexed by the identifier (0 or 1) of the responder chosen in the first step.

The process PK consists of active substitutions representing the public key of A, the public key of the well-behaved process B, and the public key of a potential intruder. The process $SPEC$ gathers the agents A and B in a coherent system together with PK, which reflects the assumption that public keys are known in advance. The secrecy of the key seeds a and b_0 is modelled with name restriction to the relevant processes; so is the freshness of nonces n_a and n_{b_0}. Note that the intruders key seed n_{b_1} is *not* private.

Note how the processes $A_2(i)$ and B_3 use the conditional process construct to check if the expected nonce is received. If B_3 receives the expected nonce, then the protocol run is successful and the process can continue its business in B_5. Before doing so, B_4 announces the successful completion of the authentication by sending a "success message" on a distinguished channel s. The success action is necessary when describing the trace with a $\mathcal{LA}$ formula in the next section. The reason is that the test includes the private name n_{b_0} and, as we demonstrated

$$
\begin{aligned}
T(i) &\triangleq \overline{c}\langle[z_a, z_{b_i}, enc([n_a, z_a], z_{b_i})]\rangle \\
A &\triangleq T(0).A_1(0) + T(1).A_1(1) \\
A_1(i) &\triangleq c(y_2).A_2(i) \\
A_2(i) &\triangleq \text{if } fst(dec(trd(y_2), a^-)) =_{\mathcal{E}} n_a \text{ then } A_3(i) \text{ else } FA \\
A_3(i) &\triangleq \overline{c}\langle[z_a, z_{b_i}, enc(snd(dec(trd(y_2), a^-)), z_{b_i})]\rangle.A_4(i) \\
A_4(i) &\triangleq \ldots \\
B &\triangleq c(y_1).B_1 \\
B_1 &\triangleq \overline{c}\langle[z_{b_0}, fst(y_1), enc([fst(dec(trd(y_1), b_0^-)), n_{b_0}], fst(y_1))]\rangle.B_2 \\
B_2 &\triangleq c(y_3).B_3 \\
B_3 &\triangleq \text{if } dec(trd(y_3, b_0^-)) =_{\mathcal{E}} n_{b_0} \text{ then } B_4 \text{ else } FB \\
B_4 &\triangleq \overline{s}\langle succ\rangle.B_5 \\
B_5 &\triangleq \ldots \\
FA &\triangleq \ldots \\
FB &\triangleq \ldots \\
PK &\triangleq \{^{a^+}/_{z_a}\} \mid \{^{b_0{}^+}/_{z_{b_0}}\} \mid \{^{b_1{}^+}/_{z_{b_1}}\} \\
SPEC &\triangleq (\nu a, n_a)A \mid (\nu b_0, n_{b_0})B \mid PK
\end{aligned}
$$

Table 7.1: An Applied π specification of the Needham-Schroeder Public Key Protocol. The specification uses short hand notations for agent definitions and output-guarded choice which can be defined from the basic Applied π calculus.

in Section 6.3, this kind of test cannot be made in the logic. Instead we ensure that the outcome of the test is readily observable by performing an output on channel s, and this output action *can* be captured in the logic.

The remainder of the specification should be fairly self explanatory. The processes $A_4(i)$, B_5, FA and FB are left unspecified; the first two express the behaviour of the respective principals after a successful run of the authentication protocol (and may e.g. continue communication with a symmetric key generated from a nonce), and the last two processes specify the behaviour on unexpected input (which we will not concern ourselves with for the present purpose).

7.2.6 A Trace of The Attack

A formal derivation from the $SPEC$ process, which demonstrates the attack on the protocol, is given in Table 7.2.

The active substitutions arising from the three process outputs have variables x_1,x_2 and x_3. For convenience we also represent the three environment inputs by active substitutions with variables y_1,y_2 and y_3 which are *under restriction*. The labelled semantics prescribe that environment input is substituted directly into the process in question (since we are working with the early semantics). We then use the fact that $A\{^M/_x\} \equiv (\nu x)(\{^M/_x\} \mid A)$ to introduce a new active substitution with private variable.

7.3 Logical Properties

7.3.1 Capturing the Trace of the Attack

Here is a formula in the logic $\mathcal{LA}$ which expresses that a process can exhibit the trace shown in Table 7.2, and hence the formula is satisfied by the $SPEC$ process:

$$A_1 \quad \triangleq \quad \langle \nu\bar{c}\rangle\langle c(y_1)\rangle\langle \nu\bar{c}\rangle\langle c(y_2)\rangle\langle \nu\bar{c}\rangle\langle c(y_3)\rangle\langle \bar{s}\rangle\mathbf{true}$$

Note how we use the output action on the distinguished success channel to implicitly test if the input bound to y_3 contains the expected nonce. The formula A_1 is not in itself interesting though: we would also wish to express the security breach, i.e. that the nonce n_{b_0} is revealed to the environment by the end of the trace. This is where the underlying logic of frames becomes useful. The variable y_3 is bound to the term $[a^+, {b_0}^+, enc(n_{b_0}, {b_0}^+)]$. As a first attempt, one might assert that the encrypted nonce n_{b_0} contained in this message can be decrypted using environment knowledge:

$$\begin{aligned} A_2 \quad \triangleq \quad & \langle \nu\bar{c}\rangle\langle c(y_1)\rangle\langle \nu\bar{c}\rangle\langle c(y_2)\rangle\langle \nu\bar{c}\rangle\langle c(y_3)\rangle\langle \bar{s}\rangle \\ & \qquad \exists x \exists x' \exists x''(trd(y_3) > x \wedge dec(x, x') > x'') \end{aligned}$$

Recall that existential quantification ranges over the synthesis of the frame in question, which intuitively corresponds to the environment knowledge. Note also

$$
\begin{aligned}
&SPEC \equiv (\nu a, n_a)(\overline{c}\langle \xi_1 \rangle.A_1(1)) \mid (\nu b_0, n_{b_0})c(y_1).B_1 \mid PK\\
\equiv &\xrightarrow{\nu x_1 \overline{c}\langle x_1 \rangle} (\nu a, n_a)(\{{}^{\xi_1}/_{x_1}\} \mid A_1(1)) \mid (\nu b_0, n_{b_0})c(y_1).B_1 \mid PK\\
&\equiv (\nu a, n_a, b_0, n_{b_0})(\{{}^{\xi_1}/_{x_1}\} \mid A_1(1) \mid c(y_1).B_1) \mid PK\\
&\xrightarrow{c(y_1)} \equiv (\nu a, n_a, b_0, n_{b_0}, y_1)(\{{}^{\xi_1}/_{x_1}\} \mid \{{}^{\xi_2}/_{y_1}\} \mid A_1(1) \mid B_1) \mid PK\\
&= (\nu a, n_a, b_0, n_{b_0}, y_1)(\{{}^{\xi_1}/_{x_1}\} \mid \{{}^{\xi_2}/_{y_1}\} \mid A_1(1) \mid \overline{c}\langle \xi_3 \rangle.B_2) \mid PK\\
\equiv &\xrightarrow{\nu x_2.\overline{c}\langle x_2 \rangle} (\nu a, n_a, b_0, n_{b_0}, y_1, y_2)(\{{}^{\xi_1}/_{x_1}\} \mid \{{}^{\xi_2}/_{y_1}\} \mid \{{}^{\xi_3}/_{x_2}\} \mid A_1(1) \mid B_2) \mid PK\\
&= (\nu a, n_a, b_0, n_{b_0}, y_1, y_2)(\{{}^{\xi_1}/_{x_1}\} \mid \{{}^{\xi_2}/_{y_1}\} \mid \{{}^{\xi_3}/_{x_2}\} \mid c(y_2).A_2(1) \mid B_2) \mid PK\\
&\xrightarrow{c(y_2)} \equiv (\nu a, n_a, b_0, n_{b_0}, y_1, y_2)(\{{}^{\xi_1}/_{x_1}\} \mid \{{}^{\xi_2}/_{y_1}\} \mid \{{}^{\xi_3}/_{x_2}\} \mid \{{}^{\xi_4}/_{y_2}\}\\
&\quad \mid (\text{if } \xi_5 =_{\mathcal{E}} n_a \text{ then } A_3 \text{ else } FA) \mid B_2) \mid PK\\
&\rightarrow (\nu a, n_a, b_0, n_{b_0}, y_1, y_2)(\{{}^{\xi_1}/_{x_1}\} \mid \{{}^{\xi_2}/_{y_1}\} \mid \{{}^{\xi_3}/_{x_2}\} \mid \{{}^{\xi_4}/_{y_2}\} \mid A_3 \mid B_2 \mid PK\\
&= (\nu a, n_a, b_0, n_{b_0}, y_1, y_2)(\{{}^{\xi_1}/_{x_1}\} \mid \{{}^{\xi_2}/_{y_1}\} \mid \{{}^{\xi_3}/_{x_2}\} \mid \{{}^{\xi_4}/_{y_2}\} \mid \overline{c}\langle \xi_6 \rangle.A_4 \mid B_2 \mid PK\\
\equiv &\xrightarrow{\nu x_3 \overline{c}\langle x_3 \rangle} (\nu a, n_a, b_0, n_{b_0}, y_1, y_2)(\{{}^{\xi_1}/_{x_1}\} \mid \{{}^{\xi_2}/_{y_1}\} \mid \{{}^{\xi_3}/_{x_2}\} \mid \{{}^{\xi_4}/_{y_2}\} \mid \{{}^{\xi_6}/_{x_3}\} \mid A_4 \mid B_2 \mid PK\\
&= (\nu a, n_a, b_0, n_{b_0}, y_1, y_2)(\{{}^{\xi_1}/_{x_1}\} \mid \{{}^{\xi_2}/_{y_1}\} \mid \{{}^{\xi_3}/_{x_2}\} \mid \{{}^{\xi_4}/_{y_2}\} \mid \{{}^{\xi_6}/_{x_3}\} \mid A_4 \mid c(y_3).B_3 \mid PK\\
&\xrightarrow{c(y_3)} \equiv (\nu a, n_a, b_0, n_{b_0}, y_1, y_2, y_3)(\{{}^{\xi_1}/_{x_1}\} \mid \{{}^{\xi_2}/_{y_1}\} \mid \{{}^{\xi_3}/_{x_2}\} \mid \{{}^{\xi_4}/_{y_2}\} \mid \{{}^{\xi_6}/_{x_3}\} \mid \{{}^{\xi_7}/_{y_3}\}\\
&\quad \mid A_4 \mid \text{if } \xi_8 =_{\mathcal{E}} n_{b_0} \text{ then } B_4 \text{ else } FB) \mid PK\\
&\rightarrow (\nu a, n_a, b_0, n_{b_0}, y_1, y_2, y_3)(\{{}^{\xi_1}/_{x_1}\} \mid \{{}^{\xi_2}/_{y_1}\} \mid \{{}^{\xi_3}/_{x_2}\} \mid \{{}^{\xi_4}/_{y_2}\} \mid \{{}^{\xi_6}/_{x_3}\} \mid \{{}^{\xi_7}/_{y_3}\}\\
&\quad \mid A_4 \mid \overline{s}\langle succ \rangle.B_5) \mid PK\\
\equiv &\xrightarrow{\nu x_4 \overline{c}\langle x_4 \rangle} (\nu a, n_a, b_0, n_{b_0}, y_1, y_2, y_3)(\{{}^{\xi_1}/_{x_1}\} \mid \{{}^{\xi_2}/_{y_1}\} \mid \{{}^{\xi_3}/_{x_2}\} \mid \{{}^{\xi_4}/_{y_2}\} \mid \{{}^{\xi_6}/_{x_3}\} \mid \{{}^{\xi_7}/_{y_3}\}\\
&\quad \mid \{{}^{succ}/_{x_4}\} \mid A_4 \mid B_5) \mid PK
\end{aligned}
$$

$$
\begin{array}{ll}
\xi_1 = [z_1, z_{b_1}, enc([n_a, z_a], z_{b_1})] & [a^+, {b_1}^+, enc([n_a, a^+], {b_1}^+)]\\
\xi_2 = [z_a, z_{b_0}, enc([fst(dec(trd(x_1), b_1^-)), z_a], z_{b_1})] & [a^+, {b_0}^+, enc([n_a, a^+], {b_0}^+)]\\
\xi_3 = [z_{b_0}, fst(y_1), enc([fst(dec(trd(y_1), b_0^-)), n_{b_0}], fst(y_1))] & [{b_0}^+, a^+, enc([n_a, n_{b_0}], a^+)]\\
\xi_4 = [z_{b_1}, z_a, trd(x_2)] & [{b_1}^+, a^+, enc([n_a, n_{b_0}], a^+)]\\
\xi_5 = fst(dec(trd(y_2), a^-)) & n_a\\
\xi_6 = [z_a, z_{b_1}, enc(snd(dec(trd(y_2), a^-)), z_{b_1})] & [a^+, {b_1}^+, enc(n_{b_0}, {b_1}^+)]\\
\xi_7 = [z_a, z_{b_0}, enc(dec(trd(x_3), b_1^-), z_{b_0})] & [a^+, {b_0}^+, enc(n_{b_0}, {b_0}^+)]\\
\xi_8 = dec(trd(y_3, b_0^-)) & n_{b_0}
\end{array}
$$

Table 7.2: A trace of the attack on the Needham-Schroeder Public Key Protocol. The trace is shown in the top part of the table and relies on auxiliary definitions of terms as given in the bottom part, which consists of two columns: the left column gives the actual terms, while the right column gives the corresponding normal forms of the terms after applying active substitutions.

that we have to use an existentially quantified variable x'' instead of directly using n_{b_0} because this is a private name. Does the process $SPEC$ satisfy A_2? The answer is no, because b_0^- is not in the frame synthesis of the relevant derivative of $SPEC$. The reason is that the name b is private in process B and is not revealed to the environment at any point in the trace. This corresponds to the intuition of the attack: the nonce n_{b_0} is not obtained by decrypting the message embedded in y_3, but rather by using A as an oracle and tricking A into decrypting the message embedded in y_3.

Hence an alternative formulation is called for:

$$A_3 \triangleq \langle\nu\bar{c}\rangle\langle c(y_1)\rangle\langle\nu\bar{c}\rangle\langle c(y_2)\rangle\langle\nu\bar{c}\rangle\langle c(y_3)\rangle\langle\bar{s}\rangle\exists x(\mathit{trd}(y_3) = \mathit{enc}(x, z_{b_0}))$$

This formula asserts that there is some synthesis term (x) which is syntactically equal to the term encrypted in y_3, namely n_{b_0}. The process $SPEC$ *does* satisfy A_3 because n_{b_0} *is* in the frame synthesis of the relevant derivative of $SPEC$. More precisely, the frame of the second last derivate of $SPEC$ is given as follows:

$$\varphi = (\nu a, n_a, b_0, n_{b_0}, y_1, y_2, y_3)(\{{}^{\xi_1}/_{x_1}\} \mid \{{}^{\xi_2}/_{y_1}\} \mid \{{}^{\xi_3}/_{x_2}\} \mid \{{}^{\xi_4}/_{y_2}\} \mid \{{}^{\xi_6}/_{x_3}\} \mid \{{}^{\xi_7}/_{y_3}\})$$

The interesting term is ξ_6 (bound to x_3) which, when the private active substitutions are applied, reduces to the triple $[a^+, {b_1}^+, \mathit{enc}(n_{b_0}, {b_1}^+)]$. Since the name b_1 is free in φ, the last term of this triple can be decrypted so we have that

$$\mathit{ecores}(x_3, \varphi) = \{a^+, {b_1}^+, \mathit{enc}(n_{b_0}, {b_1}^+), n_{b_0}\}$$

Since $\mathit{ecores}(\varphi) \subseteq \mathcal{S}(\varphi)$ we have that $n_{b_0} \in \mathcal{S}(\varphi)$, confirming the satisfaction

$$\varphi \vDash_{\mathcal{LF}} \exists x(\mathit{trd}(y_3) = \mathit{enc}(x, z_{b_0}))$$

and hence $SPEC \vDash A_3$.

Yet another approach would be employ the technique in the proof of Proposition 3 in [2], which states that the problem of deciding static equivalence reduces to deciding membership of the synthesis set in theories which include public-key axioms. The idea is to add an active substitution with a fresh private name encrypted under the public key generated from n_{b_0} to the $SPEC$ process:

$$SPEC' \triangleq SPEC \mid (\nu h)\{{}^{\mathit{enc}(h, {n_{b_0}}^+)}/_{y}\}$$

Then the nonce n_{b_0} can be deduced by the environment in the end of a successful protocol run exactly if $SPEC'$ satisfies the following formulae:

$$A_4 \triangleq \langle\nu\bar{c}\rangle\langle c(y_1)\rangle\langle\nu\bar{c}\rangle\langle c(y_2)\rangle\langle\nu\bar{c}\rangle\langle c(y_3)\rangle\langle\bar{s}\langle \mathit{succ}\rangle\rangle\exists x\exists x'(\mathit{dec}(y, x) > x')$$

The formula A_3 describes the desired assertion more directly than A_4, but A_4 has the advantage of also working in a setting with symmetric key encryption. If the message containing the nonce n_{b_0} were encrypted with a symmetric key not known by the environment, the term containing the nonce n_{b_0} could not be reconstructed in the logic and hence syntactical equality could not be tested as in A_3.

7.3.2 General Safety Properties – Invariance

We have now demonstrated that the logic is sufficiently strong to capture the attack on the Needham-Schroeder Public Key Protocol. However the formulae A_3 and A_4 exhibited above were designed with the specific attack in mind. In a more general situation, when one is faced with a protocol specification to be verified, one would wish to express more general safety properties, e.g. that there is no possible trace leading to the disclosure of the nonce n_{b_0}.

More specifically, we would like to assert that whenever an (arbitrary) execution trace of the protocol leads to the input of y_3 in B_3 followed an output of the success message, the nonce embedded in the input cannot be deduced by the environment. This amounts to asserting that the following formula (which is just the last part of A_3) holds *invariantly* – i.e. in any possible derived state of $SPEC$:

$$A_5 \triangleq \langle c(y_3)\rangle\langle\overline{s}\langle succ\rangle\rangle\neg\exists x(trd(y_3) = enc(x, z_{b_0}))$$

If we assume that $A_4(i)$, B_5, FA and FB are finite processes, in the sense that they do not afford infinite transition sequences, then the $SPEC$ process is also finite. In this case invariance can be defined using only constructs from the basic logic $\mathcal{LA}$. Let $\mathcal{U}$ be any finite set of names and variables. We then define *invariance of A in k steps* inductively as follows:

$$\begin{aligned} Inv^0_{\mathcal{U}}(A) &\triangleq A \\ Inv^{k+1}_{\mathcal{U}}(A) &\triangleq A \wedge [\tau]Inv^k_{\mathcal{U}}(A) \wedge \\ &\quad \bigwedge_{u\in\mathcal{U}} [\bar{u}]Inv^k_{\mathcal{U}}(A) \wedge [\nu\bar{u}]Inv^k_{\mathcal{U}}(A) \wedge [u(x)]Inv^k_{\mathcal{U}}(A) \end{aligned}$$

Recall that the box modalities express *necessity*, so e.g. $[\tau]A$ expresses that no matter how an internal reduction is performed, the derived process must satisfy A. Hence the formula $Inv^k(A)$ expresses that no matter how a process performs k consecutive actions, the resulting derived process must satisfy A.

Let $\mathcal{U}$ be the set of variables and names occurring in $SPEC$ and let k_{max} be the upper bound on the length of possible transition sequences that $SPEC$ can afford. We are now equipped to express a safety property which asserts that $SPEC$ does not reveal the nonce n_{b_0} to the environment at any point in its execution:

$$A_6 \triangleq Inv^{k_{max}}_{\mathcal{U}}(A_5)$$

Clearly $SPEC \not\models A_6$ as exemplified by the formula A_3 given in the previous subsection, i.e. the safety property is not satisfied by our specification.

The definition of the invariance operator relies on processes with finite-length transition sequences (not to be mistaken for processes with finite *states*). However, systems with infinite behaviour arise in many settings where the above definition of invariance therefore would not suffice. A standard solution is to introduce process variables and recursion into the logic and thereby obtain a μ-calculus [17]. This allows invariance to be defined as a maximal fixed point of an appropriate semantic function over a complete lattice [6, Section 5.2]. Tarski's Fixed Point Theorem [6, Appendix A] gives us that a unique such fixed point exists if the function is monotonic, and in order to satisfy this one would have to omit negation from the logic. We leave further developments in this direction for future work.

7.4 Summary

This chapter has demonstrated how the logic for Applied π can be applied to express properties of security protocols. We have introduced the Needham-Schroeder Public Key Protocol and shown a well known "man in the middle" attack in which an intruder may learn a secret nonce. We have modelled the protocol in Applied π and shown a transition sequence leading to the attack. Finally we have discussed how the logic for Applied π may be used to express properties of the protocol, and presented formulae which capture the attack.

Chapter 8

Conclusion

We have introduced Abadi and Fournet's Applied π calculus and presented a logic for the calculus which characterises labelled bisimilarity and hence also observational equivalence. The motivation is similar to that of Applied π itself, namely generality: the logic can be adapted to a particular application simply by defining a suitable function signature and equational theory.

Since labelled bisimilarity contains a condition on static equivalence on frames, the first step towards a logic for Applied π is a logic for frames which characterises static equivalence. A logic for frames in turn relies on a definition of static equivalence which does not contain a universal quantification over arbitrary terms. The following section summarises the major contributions of this report, and the subsequent section gives pointers to future work.

8.1 Contributions

8.1.1 Strong Static Equivalence

The first major contribution of this report is an strong version of static equivalence, $\approx_{ss}$, in which frames may be distinguished by testing on reduction of terms in addition to equality. We have argued that strong static equivalence is meaningful in applications. But strong static equivalence is particularly useful because a refined definition, $\approx'_{ss}$, can be given which does not depend on universal quantification over arbitrary terms. The refined definition is based on the notion of ecores (extended cores, which intuitively are the minimal relevant terms which can be deduced from a frame): corresponding terms in the two frames must be equal up to ecores, and any syntactic equalities and reductions on contexts over ecores which hold in one frame must also hold in the other. We then show that $\approx_{ss}$ and $\approx'_{ss}$ coincide in independent convergent subterm theories, which are the theories we have considered in this report. The refined definition is used as a basis for the logic of frames and to show that this logic characterises strong static equivalence. A further refinement, $\approx''_{ss}$, which is also shown to be equivalent to $\approx_{ss}$, is given and used as a basis for characteristic formulae in the logic of frames. Finally we show that strong static equivalence coincides with the standard static equivalence in cases where frames do not contain inaccessible terms. This is for example satisfied by theories of symmetric key encryption and by theories of public key encryption if public keys are assumed to always be known.

8.1.2 A Logic for Frames

The second major contribution of this report is a first order logic for frames, $\mathcal{LF}$, which characterises strong static equivalence and which is amenable to construction of characteristic formulae. $\mathcal{LF}$ includes the atomic propositions $M_1 =_{\mathcal{E}} M_2$ (equality), $M_1 > M_2$ (reduction) and $M_1 = M_2$ (syntactic equality) which allow direct reasoning about the terms in a frame. Common to all three is that M_1 and M_2 may not contain private names of a frame in order for the frame to satisfy them. $\mathcal{LF}$ also contains first order quantification on the form $\exists x(A)$ ranging over the synthesis of a frame (i.e. any terms which can be constructed from ecores), which allows indirect reasoning about the terms that can be deduced from a frame and greatly increases the expressiveness of the logic. This enables formulae which e.g. express that
there exists a term y (known by the environment) such that x_1 can be decrypted with y to obtain x_2

8.1.3 A Logic for Applied π

The third and final major contribution of this report is a modal logic for Applied π, $\mathcal{LA}$, which characterises labelled bisimilarity on processes. $\mathcal{LA}$ is obtained by adding suitable Hennessy-Milner style modalities to $\mathcal{LF}$. More specifically, we add the modalities $\langle\tau\rangle A_1$ (possibility of *internal reduction*), $\langle\bar{a}\rangle A_1$ (possibility of *some output on a*), $\langle\nu\bar{a}\rangle A_1$ (possibility of *bound output on a*) and $\langle a(x)\rangle A_1$ (possibility of *input of some term bound to x on a*). We have further demonstrated how a restricted version of match can be defined based on the underlying frame logic, and likewise how a matching input modality $\langle a({}^{M}/_{x})\rangle A_1$ (possibility of *input of term M on a*) can be defined from the standard input modality and match. The result is a logic which can reason both about the dynamic behaviour and static characteristics of processes, and which can be used to make assertions on the form
process P can perform an input of an encrypted term M_1 followed by some output, after which the decryption key for the term M_1 is known by the environment
Finally, we have demonstrated how $\mathcal{LF}$ can be applied to capture a well known attack on the Needham-Schroeder Public Key Protocol.

8.2 Future Work

Many interesting directions for future work still remain, and in this section we discuss a few of them.

8.2.1 A Broader Class of Theories.

In this report we have focused on independent convergent subterm theories. We have argued that the assumption of independence (namely that destructor function symbols do not occur inside rewrite rules) is not very strong. However the assumption of subterm theories is rather strong, and convergent theories which are *not* subterm theories arise in many practical applications. For example, in [19] an analysis of security protocols based on weaknesses in block chaining modes is carried out. A block chaining mode such as ECB (Electronic Code Book) specifies how a clear text can be broken into a list of blocks which are encrypted separately. A decryption function symbol operating on lists could therefore be defined as follows (where the $cons(\cdot, \cdot)$ function symbol is the list constructor from the theory $\mathcal{E}_{NS}$ defined in Subsection 7.2.2):

$$dec(cons(enc(z_1, z_2), z_3), z_2) > cons(z_1, dec(z_3, z_2))$$

This is however not a subterm rule. Hence it would be interesting to extend the results in this report to convergent theories which are not necessarily subterm theories. We expect this to be fairly easy since we only use the assumption of subterm theories to argue that the reduct of a context over ecores is itself a context over ecores. However, a more careful definition of cores and ecores may be necessary.

8.2.2 Expressiveness

We have demonstrated how the logic $\mathcal{LA}$ can be used to reason about an authentication protocol. One of the key motivations for a logic for Applied π is generality, so one would hope that the logic captures other logics given a suitable equational theory. We have suggested that $\mathcal{LA}$ resembles the Spi logic in a theory of symmetric key encryption. There are also similarities to the BAN logic [11] which has been used to reason about authentication protocols. The BAN logic includes formulae such as "P sees X", which asserts that process P can see message X based on the messages it has received in its current state; this resembles existential quantification over synthesis terms in $\mathcal{LA}$. Therefore, a general future direction could be to investigate the expressiveness of $\mathcal{LA}$ in relation to existing logics.

8.2.3 Algorithms and Complexity

Decidability of refined strong static equivalence and of satisfiability in the logics clearly rely on finding algorithms for computing the analysis, cores and ecores of a frame. Although we are fairly confident that such algorithms exist, we have not paid much attention to them in this report. Hence a possible future direction would be to devise and implement such algorithms. It would also be interesting to analyse the time complexity of deciding static equivalence in convergent subterm theories using our refined definitions of strong static equivalence. In particular it would be interesting to investigate if one could match or improve on the polynomial time bound given in [2].

8.2.4 Tool Support

Analysis of models in Applied π is supported by the tool ProVerif [8] by Bruno Blanchet. The motivation for introducing a logic for Applied π is to provide an alternative approach to specifying security properties about a model. Hence it would be of interest to extend ProVerif with support for our logic $\mathcal{LA}$.

8.2.5 Recursion

In Chapter 7 we described how the logic $\mathcal{LA}$ can be used to specify properties about a security protocol. We also motivated the need for invariance formulae, which can be realised by introducing recursion and hence obtaining a μ-calculus. This extension has however been left for future work and, although standard, may involve making restrictions to the logic such as disallowing negation.

Bibliography

[1] Martín Abadi, Mathieu Baudet, and Bogdan Warinschi. Guessing attacks and the computational soundness of static equivalence. In *FoSSaCS*, volume 3921 of *Lecture Notes in Computer Science*, pages 398–412. Springer, 2006.

[2] Martín Abadi and Véronique Cortier. Deciding knowledge in security protocols under equational theories. In *Proc. 31st Int. Coll. Automata, Languages, and Programming (ICALP'2004), volume 3142 of Lecture Notes in Computer Science*, pages 46–58. Springer, 2004.

[3] Martín Abadi and Véronique Cortier. Deciding knowledge in security protocols under (many more) equational theories. In *CSFW '05: Proceedings of the 18th IEEE Computer Security Foundations Workshop (CSFW'05)*, pages 62–76, Washington, DC, USA, 2005. IEEE Computer Society.

[4] Martín Abadi and Cedric Fournet. Mobile values, new names, and secure communication. In *POPL '01: Proceedings of the 28th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 104–115, New York, NY, USA, 2001. ACM Press.

[5] Martín Abadi and Andrew D. Gordon. A calculus for cryptographic protocols: The Spi calculus. In *Fourth ACM Conference on Computer and Communications Security*, pages 36–47. ACM Press, 1997.

[6] Luca Aceto, Anna Ingólfsdóttir, Kim G. Larsen, and Jiri Srba. *Reactive Systems: Modelling, Specification and Verification. DRAFT.* Unpublished, 2006.

[7] Franz Baader and Tobias Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.

[8] Bruno Blanchet. An efficient cryptographic protocol verifier based on prolog rules. In *CSFW '01: Proceedings of the 14th IEEE Workshop on Computer Security Foundations*, page 82. IEEE Computer Society, 2001.

[9] Michele Boreale, Rocco De Nicola, and Rosario Pugliese. Proof techniques for cryptographic processes. *SIAM J. Comput.*, 31(3):947–986, 2001.

[10] Johannes Borgström. Static equivalence *is* harder than knowledge. To appear in Proceedings of EXPRESS 2005.

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

[12] D. Dolev and A. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.

[13] Ulrik Frendrup, Hans Hüttel, and Jesper Nyholm Jensen. Modal logics for cryptographic processes. *Electr. Notes Theor. Comput. Sci.*, 68(2), 2002.

[14] Ulrik Frendrup and Jesper Nyholm Jensen. Bisimilarity in the Spi-calculus. Master's thesis, Aalborg University, Department of Computer Science, 2001.

[15] Matthew Hennessy and Robin Milner. Algebraic laws for nondeterminism and concurrency. *J. ACM*, 32(1):137–161, 1985.

[16] Tony Hoare and Robin Milner. Grand challenges for computing research. *Comput. J.*, 48(1):49–52, 2005.

[17] Dexter Kozen. Results on the propositional mu-calculus. *Theor. Comput. Sci.*, 27:333–354, 1983.

[18] Steve Kremer and Mark Ryan. Analysis of an electronic voting protocol in the Applied Pi calculus. In Shmuel Sagiv, editor, *ESOP*, volume 3444 of *Lecture Notes in Computer Science*, pages 186–200. Springer, 2005.

[19] Steve Kremer and Mark D. Ryan. Analysing the vulnerability of protocols to produce known-pair and chosen-text attacks. In Riccardo Focardi and Gianluigi Zavattaro, editors, *Proceedings of the 2nd International Workshop on Security Issues in Coordination Models, Languages and Systems (SecCo'04)*, volume 128 of *Electronic Notes in Theoretical Computer Science*, pages 84–107. Elsevier Science Publishers, May 2005.

[20] Gavin Lowe. Breaking and fixing the Needham-Schroeder Public-Key Protocol using fdr. In *TACAs '96: Proceedings of the Second International Workshop on Tools and Algorithms for Construction and Analysis of Systems*, pages 147–166, London, UK, 1996. Springer-Verlag.

[21] Robin Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer-Verlag, 1980.

[22] Robin Milner. *Communicating and Mobile Systems – The Pi Calculus*. Cambridge University Press, 1999.

[23] Robin Milner, Joachim Parrow, and David Walker. Modal logics for mobile processes. *Theoretical Computer Science*, 114(1):149–171, 1993.

[24] Roger M. Needham and Michael D. Schroeder. Using encryption for authentication in large networks of computers. *Commun. ACM*, 21(12):993–999, 1978.

[25] Joachim Parrow. *Handbook of Process Algebra*, chapter An introduction to the Pi-calculus. Elsevier, 2001.

[26] D. Sangiorgi and D. Walker. Some results on barbed equivalences in Pi-calculus. In *Proc. CONCUR '01*, volume 2154, 2001.

[27] OpenSSL: The open source toolkit for SSL/TLS. http://user.it.uu.se/ joachim/intro.ps.

Recent BRICS Report Series Publications

RS-06-19 **Michael David Pedersen.** ***Logics for The Applied π Calculus*****. December 2006. viii+111 pp.**

RS-06-18 **Małgorzata Biernacka and Olivier Danvy.** ***A Syntactic Correspondence between Context-Sensitive Calculi and Abstract Machines*****. dec 2006. iii+39 pp. Extended version of an article to appear in TCS. Revised version of BRICS RS-05-22.**

RS-06-17 **Olivier Danvy and Kevin Millikin.** ***A Rational Deconstruction of Landin's J Operator*****. December 2006. ii+37 pp. Revised version of BRICS RS-06-4. A preliminary version appears in the proceedings of IFL 2005, LNCS 4015:55–73.**

RS-06-16 **Anders Møller.** ***Static Analysis for Event-Based XML Processing*****. October 2006. 16 pp.**

RS-06-15 **Dariusz Biernacki, Olivier Danvy, and Kevin Millikin.** ***A Dynamic Continuation-Passing Style for Dynamic Delimited Continuations*****. October 2006. ii+28 pp. Revised version of BRICS RS-05-16.**

RS-06-14 **Giorgio Delzanno, Javier Esparza, and Jiří Srba.** ***Monotonic Set-Extended Prefix Rewriting and Verification of Recursive Ping-Pong Protocols*****. July 2006. 31 pp. To appear in ATVA '06.**

RS-06-13 **Jiří Srba.** ***Visibly Pushdown Automata: From Language Equivalence to Simulation and Bisimulation*****. July 2006. 21 pp. To appear in CSL '06.**

RS-06-12 **Kristian Støvring.** ***Higher-Order Beta Matching with Solutions in Long Beta-Eta Normal Form*****. June 2006. 13 pp. To appear in *Nordic Journal of Computing*, 2006.**

RS-06-11 **Kim G. Larsen, Ulrik Nyman, and Andrzej Wasowski.** ***An Interface Theory for Input/Output Automata*****. June 2006. 40 pp. Appears in Misra, Nipkow and Sekerinski, editors, *Formal Methods: 14th International Symposium*, FM '06 Proceedings, LNCS 4085, 2006, pages 82–97.**

RS-06-10 **Christian Kirkegaard and Anders Møller.** ***Static Analysis for Java Servlets and JSP*****. June 2006. 23 pp. Full version of paper presented at SAS '06.**

RS-06-9 **Claus Brabrand, Robert Giegerich, and Anders Møller.** ***Analyzing Ambiguity of Context-Free Grammars*****. April 2006. 19 pp.**