Basic Research in Computer Science

BRICS RS-02-25 H. Hüttel: Deciding Framed Bisimilarity

Deciding Framed Bisimilarity

Hans Hüttel

BRICS Report Series **RS-02-25**

ISSN 0909-0878 **May 2002**

Copyright © 2002, Hans Hüttel.
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
Ny Munkegade, building 540
DK–8000 Aarhus C
Denmark
Telephone: +45 8942 3360
Telefax: +45 8942 3255
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/02/25/`

Deciding Framed Bisimilarity

Hans Hüttel*

BRICS, Department of Computer Science, Aalborg University
Fredrik Bajers Vej 7E, 9220 Aalborg Øst, Denmark

May 2002

Abstract

The spi-calculus, proposed by Abadi and Gordon, is a process calculus based on the π-calculus and is intended for reasoning about the behaviour of cryptographic protocols. We consider the finite-control fragment of the spi-calculus, showing it to be Turing-powerful (a result which is joint work with Josva Kleist, Uwe Nestmann, and Björn Victor.) Next, we restrict our attention to finite (non-recursive) spi-calculus. Here, we show that framed bisimilarity, an equivalence relation proposed by Abadi and Gordon, showing that it is decidable for this fragment.

1 Introduction

The spi-calculus, originally proposed by Abadi and Gordon [AG97a], is a process calculus based on the π-calculus [MPW92] and is intended for describing and reasoning about the behaviour of cryptographic protocols.

An important insight of the spi-calculus is that correctness properties can be expressed as statements of behavioural equivalence. For instance, a protocol $P(M)$ transmitting the message x satisfies the secrecy property w.r.t. M if we cannot distinguish between two instances of P which transmit different messages. Expressed using behavioural equivalence, this reduces to stating that

*hans@cs.auc.dk

$$\forall M_1, M_2 . P(M_1) \sim P(M_2)$$

Deciding correctness properties of cryptographic protocols now amounts to deciding the behavioural equivalence $\sim$.

Various notions of behavioural equivalence have been put forward. Abadi and Gordon [AG97a] choose *may-testing equivalence* (originally proposed by De Nicola and Hennessy [NH83]). While may-testing is ideal from a philosophical point of view – processes are equivalent iff they behave in the same way under all attacks/observations – this equivalence is defined via universal quantification over observer processes and is therefore less ideal from the perspective of actually determining the equivalence of processes.

Consequently, in [AG98b] Abadi and Gordon define a bisimulation-style equivalence, *framed bisimilarity*, and show it to be as a sound approximation of may-testing equivalence. A main motivation behind their work was to define a notion of behavioural equivalence which has a useful proof technique and is decidable.

The main focus of this paper is to examine to which extent the latter is the case.

As the full spi-calculus is Turing-powerful one can only hope for a positive decidability result within a proper subcalculus. A natural candidate would be the *finite-control* spi-calculus, the spi-calculus counterpart of regular CCS; finite-control processes have a bounded number of parallel components and, because of the presence of recursion, are able to describe multiple protocol runs.

However, even the finite-control spi-calculus is Turing-powerful [HKNV97]. In this paper we first demonstrate this by presenting an encoding of Minsky's two-counter machines into the finite-control calculus, a result which is joint work with Josva Kleist, Uwe Nestmann, and Björn Victor.

Next, we restrict our attention to *finite* spi-calculus processes and show that framed bisimilarity is decidable in this fragment. The finite spi-calculus processes are the recursion-free processes of the spi-calculus, corresponding to single runs of a cryptographic protocol.

In [AL00] Amadio and Lugiez consider a finite spi-calculus similar to ours and show that its associated reachability problem is decidable (albeit NP-hard). As further work they mention finding an algorithm for deciding bisimilarity.

A main problem in obtaining our result stems from matching input transitions, since two processes must be equivalent under *all* value instantiations; we overcome this problem by showing that only finitely many values need be considered.

2 The spi-calculus

The spi-calculus extends the π-calculus [MPW92, Mil99] with primitives for encryption and decryption. As in the π-calculus, communication takes place over channels that can either be public or restricted. Messages may be decrypted; the perfect encryption hypothesis is adopted in the spi-calculus – an attacker cannot guess the key of an encrypted message.

2.1 Syntax

In this section we present the two fragments of the spi-calculus that we shall study in the rest of the paper. Our syntax largely follows that of [AG97a]. We only consider shared key cryptography since the definitions related to framed bisimilarity in [AG98b] only use shared key cryptography. However, an extension of the results in the present paper should be straightforward.

2.1.1 Terms

Common to our two fragments is the set of terms that can be communicated by processes. Unlike the π-calculus, the spi-calculus allows us to communicate composite terms. The set of terms, $\mathcal{T}$, has its syntax defined by the following grammar.

$$L, M, N ::= x \mid n \mid \{M\}_N \mid (M, N)$$

In the above, x ranges over the set of variables, n ranges over the set of *names*, $\{M\}_N$ denotes the term M encrypted using key N and (M, N) denotes the pair whose components are the terms M and N.

2.1.2 The finite-control spi-calculus

The finite-control spi-calculus is a straightforward extension of the finite-control π-calculus introduced by Lin [Lin91].

As the definition below shows, a finite-control process consists of a fixed number of sequential processes running in parallel.

Definition 1 The set of finite-control spi-calculus processes is given by the grammar

$$
\begin{array}{rcl}
R & ::= & M(x).R \mid \overline{M}\langle N\rangle.R \mid (\nu n)R \\
& \mid & D(M) \mid \mathbf{0} \mid [M = N]\, R \mid R_1 + R_2 \\
& \mid & \text{let } (x, y) = M \text{ in } R \mid \mathsf{rec}D(M).R \\
& \mid & \text{case } L \text{ of } \{x\}_N \text{ in } R \\
P & ::= & R \mid (\nu n)P \mid P|P
\end{array}
$$

The spi-calculus distinguishes between variables $x, y, z, \ldots \in \mathcal{V}$ and names $c, m, n, k \ldots \mathcal{N}$. Names refer to a key or a channel, whereas variables are instantiated to messages. When concerning channels, a name c is used for input and its co-name $\overline{c}$ used for output.

The spi-calculus has two communication primitives. $\overline{M}\langle N\rangle.P$ is output; N is emitted on the channel M. $M(x).P$ is input; the variable x is received on the channel M, and x is bound in P.

While encryption is handled at the level of message terms, decryption is a process construct. $\text{case } L \text{ of } \{x\}_N \text{ in } P$ is used to decrypt terms; x is bound in P. The other term destructor is $\text{let } (x, y) = M \text{ in } P$ which allows us to split a pair; the variables x and y are bound in P.

The remaining process constructs are also found in the π-calculus: $(\nu n)P$ is the restriction construct. The new name n is bound in P. $P \mid Q$ denotes parallel composition and $\mathbf{0}$ is the empty process. Finally, the match construct $[M = N]\, P$ can proceed as P iff M is equal to N.

In the finite-control calculus we allow two additional constructs, namely nondeterministic choice, $R_1 + R_2$ and recursively defined processes, $\mathsf{rec}D(M).R$. $D(M)$ ranges over recursion constants which may be parameterised by a term.

We identify processes up to renaming of bound names and variables. A process without any free variables is *closed*; we let $\mathcal{P}$ denote the set of closed processes. Furthermore we let $\text{fn}[\![P]\!]$ denote the set of free names in P, and $\text{fv}[\![P]\!]$ the free variables in P. For any set of terms S, we let $\text{n}(S)$ denote the set of names occurring in S, free as well as bound. $P[M/x]$ denotes the substitution of the term M for all free occurrences of x in the process P and is defined as expected.

The original presentation of the spi-calculus in [AG97a] introduces natural numbers into the syntax. This, however, is unimportant as we can encode the naturals using encryption and decryption. Let a, b be fresh names. We then let

$$
\begin{array}{rcl}
[\![0]\!] & = & a \\
[\![n + 1]\!] & = & [\![\{[\![n]\!]\}_b]\!]
\end{array}
$$

The test-for-zero process construct now becomes

$$[\![\text{case } v \text{ of } 0 : P \text{ suc}(x) : Q]\!] = \text{case } v \text{ of } \{x\}_b \text{ in } P + [v = a]Q$$

In our undecidability proof in section 3 we use natural numbers freely by implicit appeal to this encoding.

2.1.3 Finite processes

The syntax of processes in the finite spi-calculus omits nondeterministic choice and recursion from the finite-control fragment.

$$\begin{array}{rcl} P, Q, R & ::= & (\nu n)P \mid \overline{M}\langle N\rangle.P \mid M(x).P \mid P \mid Q \\ & \mid & [M = N]\, P \mid \mathbf{0} \mid \text{let } (x, y) = M \text{ in } P \mid \text{case } L \text{ of } \{x\}_N \text{ in } P \end{array}$$

2.1.4 Agents

An agent can be a process, an abstraction or a concretion. The syntax of agents is defined by the following grammar:

$$\begin{array}{rcl} A, B & ::= & P \mid C \mid F \\ F, G & ::= & (x)P \\ C, D & ::= & (\nu\vec{m})\langle M\rangle P \end{array}$$

$(x)P$ is an *abstraction*, which needs to bind a term to x before proceeding. $(\nu\vec{m})\langle M\rangle P$ is a *concretion*, which is immediately able to output the term M. $\mathcal{A}$ will denote the set of closed agents.

2.2 Semantics

Our labelled commitment semantics of the spi-calculus is that of [AG98b].

2.2.1 Reduction and structural congruence

The reduction relation describes how processes unfold and make preparations for a reaction. In particular, the rules describe how the term deconstructors behave (Table 1) and, for finite-control processes, how nondeterministic choice is resolved and that a recursive process proceeds by unfolding the recursive definition (Table 2) . In the case of a decryption we only proceed if the key is a name. See Table 1.

$$
\begin{array}{rcl}
[M = M] & > & P \\
\mathrm{let}(x, y) = (M, N)\,\mathrm{in}\,P & > & P[M/x][N/y] \\
\mathrm{case}\,\{M\}_n\ \mathrm{of}\ \{x\}_n\,\mathrm{in}\,P & > & P[M/x]
\end{array}
$$

Table 1: The reduction rules for term destructors

$$
\begin{array}{rcl}
\mathsf{rec}D(x).P & > & P[\mathsf{rec}D(M_i).P/D(M_i)] \\
P_1 + P_2 & > & P_1 \\
P_1 + P_2 & > & P_2
\end{array}
$$

Table 2: The reduction rules for recursion and choice

Structural congruence, $\equiv$, is defined in Table 3. It captures the identities that should intuitively hold.

2.2.2 The commitment relation

The *commitment transition system* $(\mathcal{P}, \{\xrightarrow{\alpha} |\ \alpha \in \mathcal{N} \cup \{\tau\}\}, \mathcal{A})$ has its transition relation defined inductively by the rules in Definition 4.

In Definition 4 we use the *interaction* operator $\bullet$ defined by

$$
C \bullet F \triangleq (\nu\vec{n})(Q \mid P[N/x]) \qquad\qquad F \bullet C \triangleq (\nu\vec{n})(P[N/x] \mid Q),
$$

when $\{\vec{n}\} \cap \mathrm{fn}[\![P]\!] = \emptyset$. Here, we extend restriction and composition as follows:

$$
\begin{array}{rcl}
(\nu n)(x)P & \triangleq & (x)(\nu n)P \\
Q \mid (x)P & \triangleq & (x)(Q \mid P) \\
(\nu n)(\nu\vec{m})\langle M\rangle P & \triangleq & \begin{cases} (\nu n, \vec{m})\langle M\rangle P & \text{if } n \in \mathrm{fn}[\![M]\!] \\ (\nu\vec{m})\langle M\rangle(\nu n)P & \text{otherwise} \end{cases} \\
Q \mid (\nu\vec{m})\langle M\rangle P & \triangleq & (\nu\vec{m})\langle M\rangle(Q \mid P)
\end{array}
$$

$$P \mid \mathbf{0} \equiv P \qquad P \mid Q \equiv Q \mid P$$

$$P \mid (Q \mid R) \equiv (P \mid Q) \mid R \qquad (\nu m)(\nu n)P \equiv (\nu n)(\nu m)P$$

$$(\nu n)\mathbf{0} \equiv \mathbf{0} \qquad P \mid (\nu n)Q \equiv (\nu n)(P \mid Q) \quad \text{if } n \notin \text{fn}[\![P]\!]$$

$$\frac{P > Q}{P \equiv Q} \qquad \frac{P \equiv Q \quad Q \equiv R}{P \equiv R}$$

$$\frac{}{P \equiv P} \qquad \frac{P \equiv Q}{Q \equiv P}$$

$$\frac{P \equiv Q}{P \mid R \equiv Q \mid R} \qquad \frac{P \equiv Q}{(\nu n)P \equiv (\nu n)Q}$$

Table 3: Rules defining structural congruence

where we assume $x \notin \text{fv}[\![Q]\!]$, $n \notin \{\vec{m}\}$ and $\{\vec{m}\} \cap \text{fn}[\![Q]\!] = \emptyset$. The dual composition $A \mid Q$ is defined symmetrically.

3 The finite-control fragment is Turing-powerful

As the finite-control spi-calculus calculus is the spi-calculus analogue of the finite-control fragment of the π-calculus, introduced by [Lin91], one might expect the situation to be same as in the π-calculus. Here, Dam [Dam97] has shown that late and early bisimilarity [MPW92] as well as open bisimilarity [San96] are all decidable. Dam's result depends on the fact that it is always suffices to consider a finite set of names due to the bounded parallelism of a finite-control process.

However, the finite-control spi-calculus is in fact Turing-powerful, destroying all hope of obtaining positive decidability results for any non-trivial notion of behavioural equivalence. The encoding presented here is joint work with Josva Kleist, Uwe Nestmann, and Björn Victor.

3.1 Encoding two-counter machines in the finite-control fragment

For our proof of this fact, we consider another universal model of computation, namely the two-counter machines of [Min67]. A two-counter machine is a simple imperative program consisting of a sequence of labelled instructions that can modify the values of two nonnegative integer counters, c_0 and c_1. Two instructions are singled out, namely L_{start} and L_{stop}. The program

(Input)	$m(x).P \xrightarrow{m} (x)P$	(Output)	$\overline{m}\langle M\rangle.P \xrightarrow{\overline{m}} (\nu)\langle M\rangle P$
(Com-1)	$\dfrac{P \xrightarrow{m} F \quad Q \xrightarrow{\overline{m}} C}{P \mid Q \xrightarrow{\tau} F \bullet C}$	(Com-2)	$\dfrac{P \xrightarrow{\overline{m}} C \quad Q \xrightarrow{m} F}{P \mid Q \xrightarrow{\tau} C \bullet F}$
(Par-1)	$\dfrac{P \xrightarrow{\alpha} A}{P \mid Q \xrightarrow{\alpha} A \mid Q}$	(Par-2)	$\dfrac{Q \xrightarrow{\alpha} A}{P \mid Q \xrightarrow{\alpha} P \mid A}$
(Res)	$\dfrac{P \xrightarrow{\alpha} A \quad \alpha \notin \{m, \overline{m}\}}{(\nu m)P \xrightarrow{\alpha} (\nu m)A}$	(Red)	$\dfrac{P > Q \xrightarrow{\alpha} A}{P \xrightarrow{\alpha} A}$

Table 4: The commitment semantics of the spi-calculus

starts with the line L_{start} and halts if L_{stop} is reached. The instruction set consists of two different types of instructions (in the indices of the counter variables we always assume addition and subtraction modulo 2):

1. L : $c_k := c_k + 1$; `goto` L_n
2. L : `if` $c_k = 0$ `then goto` L_n^1 `else` $c_k := c_k - 1$; `goto` L_n^2

We can always assume that a type 1 instruction has $L \neq L_n$ (if $L = L_n$ the machine would loop forever) and that a type 2 instruction has $L \neq L_n^1$ (here, too, if $L = L_n^1$ the machine would loop forever) and $L \neq L_n^2$ (we can simply duplicate the instruction in question.)

Theorem 2 Any two-counter machine can be simulated in the finite-control spi-calculus.

PROOF: We define an encoding $[\![]\!]$ from two-counter machine instructions into the finite-control spi-calculus. The idea is simply that the two counters are represented by processes and the each instruction corresponds to a process that communicates with the counters.

We assume the following set of names, which we denote by **n**:

- For every instruction label L_n we introduce the name l_n, used to signal a `goto`, and the constant D_{l_n}.

- For counter c_k we introduce the names

  d_k indicating that the counter is decremented

  c_k indicating that the counter is incremented

  r_k indicating that the value of the counter is being read

  A counter c_k is represented as the process

$$C_k = \mathsf{rec} D_k(x).(\overline{r_k}\langle x\rangle.D_k(x) + d_k.D_k(x-1) + i_k.D_k(x+1))$$

  Instructions are encoded as

$$\begin{aligned}
[\![L : c_k := c_k + 1; \texttt{goto } L_n]\!] &= \mathsf{rec} D_l.l.i_k.\overline{l_n}\,.D_l \\
[\![L &: \texttt{if } c_k = 0 \texttt{ then} \\
&\quad \texttt{goto } L_n^1 \texttt{ else} \\
c_k := c_k - 1;\ \texttt{goto } L_n^2]\!] &= \mathsf{rec} D_l.l.r_k(y).([y=0]\,\overline{l_n^1}\,.D_n + [y \neq 0]\,\overline{d_k}\,.\,\overline{l_n^2}\,.D_n)
\end{aligned}$$

Suppose that a two-counter machine M is composed of a sequence of instructions $S_1, \ldots, S_m$. Then the encoding of the machine is given by

$$[\![M]\!] = (\nu \mathbf{n}) \prod_{i=1}^{m} [\![S_i]\!] \mid C_0 \mid C_1$$

It is now easy see that the two-counter machine can reach a state where $c_0 = v_0$ and $c_1 = v_1$ if and only if $[\![M]\!] \stackrel{\tau}{\longrightarrow}{}^{*} P'$ where the term P' has counter constants whose values are $D_k(v_0)$ and $D_k(v_1)$, respectively. □

Corollary 3 Any nontrivial notion of behavioural equivalence is undecidable in the finite-control spi-calculus.

4 Framed bisimilarity

Framed bisimilarity was introduced by Abadi and Gordon in [AG98b].

4.1 Frames and theories

Processes are related with respect to a *frame-theory pair* which represents the knowledge of the environment.

Definition 4 A *frame fr* is a finite set of names. A *theory th* is a finite set of pairs of terms (M, N). We let e range over the set of frame-theory pairs.

$$\text{(Ind Var)} \quad \frac{}{e \vdash x \leftrightarrow x}$$

$$\text{(Ind Frame)} \quad \frac{n \in fr}{e \vdash n \leftrightarrow n} \qquad \text{(Ind Theory)} \quad \frac{(M,N) \in th}{e \vdash M \leftrightarrow N}$$

$$\text{(Ind Pair)} \quad \frac{e \vdash M \leftrightarrow M' \quad e \vdash N \leftrightarrow N'}{e \vdash (M,N) \leftrightarrow (M',N')} \qquad \text{(Ind Enc)} \quad \frac{e \vdash M \leftrightarrow M' \quad e \vdash N \leftrightarrow N'}{e \vdash \{M\}_N \leftrightarrow \{M'\}_{N'}}$$

Table 5: Rules defining the indistinguishability relation

Intuitively, when comparing processes P and Q, the elements of the frame are the names from P and Q that the attacker knows. If $(M,N) \in th$ the attacker cannot distinguish the term M coming from P and the term N coming from Q.

In what follows, when given an environment e we refer to its frame part as fr_e and its environment part as th_e.

Definition 5 Let $e = (fr, th)$ be an environment. Terms M and N are indistinguishable under e, written $e \vdash M \leftrightarrow N$, if it can be derived by the rules in Table 5.

An environment must be consistent. This is captured by

Definition 6 Environment e is ok, written $e \vdash \text{ok}$, if:

1. $\forall (M,N) \in th$ it must hold that M is closed, $\exists M_1, M_2 : M = \{M_1\}_{M_2}$ and $\nexists N_2 : e \vdash M_2 \leftrightarrow N_2$. The converse must also hold for N.

2. whenever $(M,N) \in th$ and $(M',N') \in th$, $M = M'$ iff $N = N'$.

Definition 7 Let e and e' be environments. e' extends e, written $e \leq e'$, iff $\forall M, N : e \vdash M \leftrightarrow N \Rightarrow e' \vdash M \leftrightarrow N$.

A *framed process pair* is a quadruple (fr, th, P, Q), where $P, Q \in \mathcal{P}$. If $\mathcal{R}$ is a set of framed process pairs, we write $e \vdash P\mathcal{R}Q$ when $(fr, th, P, Q) \in \mathcal{R}$. A *framed relation* is a set $\mathcal{R}$ of framed process pairs, such that $e \vdash \text{ok}$ whenever $e \vdash P\mathcal{R}Q$.

4.2 Framed simulations and bisimulations

Framed simulation is a *late* simulation [MPW92]; the choice of a matching transition for an input transition does not depend on the value that will eventually be received.

Definition 8 A *framed simulation* is a framed relation $\mathcal{S}$ such that, whenever $e \vdash P\mathcal{S}Q$, the following three conditions hold

1. If $P \xrightarrow{\tau} P'$ then there exists a process Q' such that $Q \xrightarrow{\tau} Q'$ and $e \vdash P'\mathcal{S}Q'$.

2. If $P \xrightarrow{c} (x)P'$ and $c \in fr$ then there exists an abstraction $(x)Q'$ with $Q \xrightarrow{c} (x)Q'$ and, for all sets $\{\vec{n}\}$ disjoint from $\mathrm{fn}[\![P]\!] \cup \mathrm{fn}[\![Q]\!] \cup f \cup \mathrm{fn}(th)$ and all closed terms M and N, if $(fr \cup \{\vec{n}\}, th) \vdash M \leftrightarrow N$ then $(f \cup \{\vec{n}\}, th) \vdash P'[M/x]\mathcal{S}Q'[N/x]$.

3. If $P \xrightarrow{\overline{c}} (\nu\vec{m})\langle M\rangle P'$, $c \in fr$ and $\{\vec{m}\} \cap (\mathrm{fn}[\![P]\!] \cup \mathrm{fn}(\pi_1(th)) \cup fr) = \emptyset$ then there exists a concretion $(\nu\vec{n})\langle N\rangle Q'$ with $Q \xrightarrow{\overline{c}} (\nu\vec{n})\langle N\rangle Q'$ and $\{\vec{n}\} \cap (\mathrm{fn}[\![Q]\!] \cup \mathrm{fn}(\pi_2(th)) \cup f) = \emptyset$. Furthermore $\exists e' : e \leq e'$, $e' \vdash M \leftrightarrow N$, and $e' \vdash P'\mathcal{S}Q'$.

Definition 9 A *framed bismulation* is a framed simulation $\mathcal{S}$ such that $\mathcal{S}^{-1} = \{e' \vdash Q\mathcal{S}P \mid e \vdash P\mathcal{S}Q \wedge e' = (fr, \{(M,N) \mid (N,M) \in th\})\}$ is also a framed simulation.

Definition 10 Framed bisimilarity is the greatest framed bisimulation, written $\sim_f$.

5 A decidability result

Definitions 8 and 9 do not provide us with a straightforward means of checking bisimilarity. The goal of the rest of our paper is to address this issue. More precisely, we shall show that in the case of finite processes

- we only need to consider finitely many terms when matching input transitions.
- we only need to consider finitely many possible frame extensions when matching input transitions
- we only need to consider finitely many frame-theory extensions when matching output transitions

Taken together, these observations will allow us to obtain a simple decision procedure for framed bisimilarity.

5.1 Matching input transitions

Assume that we are trying to determine whether $(fr, th) \vdash P \sim_f Q$. We have an input commitment $P \xrightarrow{c} (x)P'$, have a candidate for a matching commitment, $Q \xrightarrow{c} (x)Q'$, and now need to determine whether $P' \sim_f Q'$.

Assume that the maximal number of successive term destructors in P and Q is m, and that the maximal number of term constructors of any term in th is d. Then we need only consider the finitely many terms of depth $\leq m + d$ constructed from (fr, th) and a bounded number of new names in order to determine if $(fr, th) \vdash P' \sim_f Q'$. This must hold as the process can only inspect any input term up to m levels of encryption/pairing and because the environment may ask us to regards terms whose depth is up to d as indistinguishable.

5.1.1 The depth of terms and processes

The notion of the maximal constructor depth of a term is as expected. It counts the *level of encryption* and the *level of pairing*. The level of decryption takes precedence over the level of pairing and only the level of decryption within the contents of a ciphertext matters, as terms appearing in key position must be names. Otherwise, they will cause the process not to evolve any further.

Definition 11 The *maximal constructor depth* $\mathrm{d}(M)$ of a term M is defined inductively by the clauses

$$\begin{aligned} \mathrm{d}(n) &= 0 \\ \mathrm{d}(x) &= 0 \\ \mathrm{d}(\{M\}_N) &= \mathrm{d}(M) + 1 \\ \mathrm{d}((M, N)) &= \max(\mathrm{d}(M), \mathrm{d}(N)) \end{aligned}$$

The above definition easily extends to frame-theory pairs.

Definition 12 Let (fr, th) be a frame-theory pair where $fr = \{(M_1, N_1), \ldots, (M_k, N_k)\}$. The maximal constructor depth of (fr, th) is defined b

$$\mathrm{d}((fr, th)) = \max\{\max(\mathrm{d}(M_i), \mathrm{d}(N_i)) \mid 1 \leq i \leq k\}$$

The maximal destructor depth of a process P is the maximal number of encryptions and pairing operators that can ever be removed along the process P. Decryption and pair splitting operations each contribute by 1, whereas a parallel composition $P \mid Q$ may contribute with decryptions from both P and Q.

Definition 13 Let P be a finite process. The *maximal destructor depth* of P is denoted by $\mathrm{mdd}(P)$ and defined inductively by the clauses

$$\begin{aligned}
\mathrm{mdd}(\mathbf{0}) &= 0 \\
\mathrm{mdd}((\nu n)P) &= \mathrm{mdd}(P) \\
\mathrm{mdd}(\overline{M}\langle N\rangle.P) &= \mathrm{mdd}(P) \\
\mathrm{mdd}(M(x).P) &= \mathrm{mdd}(P) \\
\mathrm{mdd}(P \mid Q) &= \mathrm{mdd}(P) + \mathrm{mdd}(Q) \\
\mathrm{mdd}([M = N]\, P) &= \mathrm{mdd}(P) \\
\mathrm{mdd}(\text{let } (x, y) = M \text{ in } P) &= \mathrm{mdd}(P) + 1 \\
\mathrm{mdd}(\text{case } L \text{ of } \{x\}_N \text{ in } P) &= \mathrm{mdd}(P) + 1
\end{aligned}$$

5.1.2 d-framed bisimilarity

d-framed bisimilarity is a variant of framed bisimilarity that only requires input transitions to be matched for transmitted message terms up to a certain depth.

Definition 14 Let k be a nonnegative integer and let e be a frame-theory pair such that $e \vdash \mathrm{ok}$. We write $e \vdash M \leftrightarrow^k N$ if $e \vdash M \leftrightarrow N$ and $\max(\mathrm{d}(M), \mathrm{d}(N)) = k$. Whenever $e \vdash M \leftrightarrow^k N$ we say that M and N are *k-indistinguishable in e.*

Since we only consider terms up to a certain depth, we need only consider finitely many extensions of the frame. This is expressed in the following lemma.

Lemma 15 Let (fr, th) be a frame-theory pair and assume that $\max(\mathrm{d}(M), \mathrm{d}(N)) = k$. If there is a $(fr \cup \{\vec{n}\}, th)$ such that $(fr \cup \{\vec{n}\}, th) \vdash M \leftrightarrow^k N$, then we may choose a $\{\vec{n}\}$ where $|\vec{n}| \leq 2k$ satisfying $(fr \cup \{\vec{n}\}, th) \vdash M \leftrightarrow^k N$.

Proof: If M and N are not indistinguishable under (fr, th), this must be amended by applying the constructor rules, the rule (Ind Theory) and the rule (Ind Frame) to new names. Every application of a constructor rule can introduce at most two new names, so at most $2k$ new names can be introduced. □

Lemma 15 leads to the following definition of d-framed simulation.

Definition 16 For any nonnegative integer d, a *d-framed simulation* is a framed relation $\mathcal{S}$ such that, whenever $(fr, th) \vdash P\mathcal{S}Q$, the following three conditions hold

1. If $P \xrightarrow{\tau} P'$ then there exists a process Q' such that $Q \xrightarrow{\tau} Q'$ and $e \vdash P'\mathcal{S}Q'$.

2. If $P \xrightarrow{c} (x)P'$ and $c \in fr$ then there exists an abstraction $(x)Q'$ with $Q \xrightarrow{c} (x)Q'$ and, for all sets $\{\vec{n}\}$ disjoint from $\mathrm{fn}[\![P]\!] \cup \mathrm{fn}[\![Q]\!] \cup fr \cup \mathrm{fn}(th)$ such that $|\vec{n}| \leq 2d$ and all closed terms M and N, if $(fr \cup \{\vec{n}\}, th) \vdash M \leftrightarrow^i N$ and $0 \leq i \leq d$ then $(fr \cup \{\vec{n}\}, th) \vdash P'[M/x]\mathcal{S}Q'[N/x]$.

3. If $P \xrightarrow{\overline{c}} A \equiv (\nu\vec{m})\langle M\rangle P'$, $c \in fr$ and $\{\vec{m}\} \cap (\mathrm{fn}[\![Q]\!] \cup \mathrm{fn}(\pi_1(th)) \cup fr) = \emptyset$ then there is a concretion $B \equiv (\nu\vec{n})\langle N\rangle Q'$ such that $Q \xrightarrow{\overline{c}} B$, the set $\{\vec{n}\}$ is disjoint from $\mathrm{fn}[\![Q]\!] \cup \mathrm{fn}(\pi_2(th)) \cup fr$ and $e' \vdash P'\mathcal{S}Q'$ for some $e' \geq (fr, th)$ where $e' \vdash M \leftrightarrow N$.

Definition 17 A *d-framed bisimulation* is a d-framed simulation $\mathcal{S}$ such that $\mathcal{S}^{-1} = \{e' \vdash Q\mathcal{S}P \mid e \vdash P\mathcal{S}Q \wedge e' = (fr_e, \{(M, N) \mid (N, M) \in th_e\})\}$ is also a d-framed simulation.

Definition 18 d-framed bisimilarity is the greatest d-framed bisimulation, written $\sim_f^d$.

Our goal is to show that for finite processes P and Q we have that P and Q are framed bisimilar iff they are d-bisimilar where d is the *critical depth*.

The critical depth of (e, P, Q) is the maximal depth of terms that must be considered as inputs when determining whether P and Q are framed bisimilar under e.

Definition 19 Let (e, P, Q) be a framed process pair. The critical depth of (e, P, Q) is defined by

$$\mathrm{cd}(e, P, Q) = \mathrm{d}(e) + \max(\mathrm{mdd}(P), \mathrm{mdd}(Q))$$

We let

$$\mathrm{cd}(e, P) = \mathrm{cd}(e, P, P)$$

When considering the result of an input commitment, we only need to consider instantiations with terms whose depths do not exceed the critical depth. Intuitively, this suffices as all subterms occurring below the critical depth are inaccessible by the destructors of a process.

If two terms are indistinguishable, their subterms appearing at depth d can be replaced by fresh names for any d such that the resulting terms will still be indistinguishable. This is the idea behing d-pruning.

Example 20 Let $M = \{\{a\}_b\}_c$ and $N = \{\{d\}_e\}_f$ and assume that we have $(M, N) \in th$ for some theory th. Let $fr = \{h\}$. Then we have $(fr, th) \vdash \{M\}_h \leftrightarrow \{N\}_h$. We also have $(fr \cup \{g\}, th) \vdash \{\{g\}_g\}_h \leftrightarrow \{\{g\}_g\}_h$ where g is a fresh name not found in fr. $((fr \cup \{g\}, th), \{g\}_h \leftrightarrow \{g\}_h)$ is the 1-pruning of (e, M, N).

The pruning of a pair of terms (M, N) at depth d generates a pair of pruned terms (M', N'). M' and N' are constructed by replacing subterms appearing at levels greater than d by encryptions of arbitary fresh names by the same fresh names. The fresh names are then added to the frame.

Definition 21 Let M and N be closed terms and let $e \vdash \mathrm{ok}$. Further assume that $e \vdash M \leftrightarrow N$, that all subterms appearing in key position in M and N are names and that d is a nonnegative integer. The *d-pruning* of (e, M, N), denoted by $\mathrm{pr}_d((e, M, N))$, is defined inductively by the clauses

$$\begin{array}{lcll}
\mathrm{pr}_0(((fr, th), n, n)) & = & ((fr, th), n, n) & \\
\mathrm{pr}_0(((fr, th), M, N)) & = & ((fr, th), M, N) & \text{if } (M, N) \in th \\
\mathrm{pr}_0(((fr, th), M, N)) & = & ((fr \cup \{a\}, th), \{a\}_a, \{a\}_a) & \text{if } (M, N) \notin th \text{ and } a \text{ is fresh} \\
\mathrm{pr}_{d+1}((fr, th), \{M_1\}_k, \{N_1\}_k) & = & (e', \{M'\}_k, \{N'\}_k) & \text{where } (e', M', N') = \mathrm{pr}_d(((fr, th), M_1, N_1))
\end{array}$$

If M is an open term, we define $\mathrm{pr}_d((e, M)) = (e, M)$.

The pruning operator extends to single terms by defining $\mathrm{pr}_d((e)(M)) = \mathrm{pr}_d((e)(M, M))$.

Note that, because of the usage of unspecified fresh names, the pruning operator as defined here does not generate a unique pair of terms. This can be dealt with by means of introducing suitable bookkeeping.

Note also how the definition exploits the fact that only names are allowed in key position.

Lemma 22 If $e \vdash M \leftrightarrow N$, $d = \max(\mathrm{d}(M), \mathrm{d}(N))$ and $\mathrm{pr}_d((e, M, N)) = (e', M', N')$ then $e' \vdash M' \leftrightarrow^d N'$.

PROOF: A straightforward induction in d, appealing to Definition 21. □

We can extend the pruning operation to pairs of term vectors. This is done inductively; we prune the components of the vectors successively, extending the frame as we proceed.

Definition 23 Let $|\vec{M}| = |\vec{N}| = k$. Then $\mathrm{pr}_d((\vec{M}, \vec{N}))$ is defined inductively by

$$\mathrm{pr}_d((e, (M_1, \ldots, M_k), (N_1, \ldots, N_k))) = (e', (M_1', \ldots, M_k'), (N_1', \ldots, N_k'))$$

where

$$(e'', M_1', N_1') = \mathrm{pr}_d((e, M_1, N_1))$$

and

$$(e', (M_2', \ldots, M_k'), (N_2', \ldots, N_k')) = \mathrm{pr}_d((e'', (M_2, \ldots, M_k), (N_2, \ldots, N_k)))$$

Lemma 24 Let P be a process such that $P = A[\vec{M}/\vec{x}]$ and let $d = \mathrm{cd}(e, P)$. $P > A$ iff $P_1 > A_1$ where $P_1 = A[\vec{N}/\vec{x}]$ where $\mathrm{pr}_d((e, \vec{M})) = (e', \vec{N})$ and $A_1 = A[\vec{N}/\vec{x}]$.

PROOF: Both implications are seen to hold by an inspection of the clauses in the definition of the reduction relation. The interesting case is the decryption clause:

$$\mathrm{case}\,\{M\}_k \ \mathrm{of}\ \{y\}_k \,\mathrm{in}\, P' > P'[M/y]$$

If $P = \mathrm{case}\,\{M\}_k \ \mathrm{of}\ \{y\}_k \,\mathrm{in}\, P'$, then the definition of the pruning operator tells us that $P_1 = \mathrm{case}\,\{N\}_k \ \mathrm{of}\ \{y\}_k \,\mathrm{in}\, P_1'$ where $P' = A_1'[\vec{M}/\vec{x}]$ and $P_1' = A'[\vec{N}/\vec{x}]$ for some A_1'. We now see that

$$\mathrm{case}\,\{N\}_k \ \mathrm{of}\ \{y\}_k \,\mathrm{in}\, P_1' > P_1'[N/y]$$

□

Lemma 25 Let $P = A[\vec{M}/\vec{x}]$ and let $d = \mathrm{cd}(e, P)$. $P \xrightarrow{\alpha} A'$ iff $P_1 \xrightarrow{\alpha} A_1'$ where $P_1 = A[\vec{N}/\vec{x}]$ where $\mathrm{pr}_d((e, \vec{M})) = (e', \vec{N})$ and $A_1' = B[\vec{N}/\vec{x}]$ and $A' = B[\vec{M}/\vec{x}]$ for some B.

PROOF: In the case of both implications, the proof proceeds by transition induction. The induction hypothesis in the case concerning the rule (Red) uses Lemma 24. The only other interesting cases are the prefix axioms. □

Theorem 26 Let P and Q be finite spi processes and let $d = \mathrm{cd}(e, P, Q)$ where $e \vdash \mathrm{ok}$. We have that $e \vdash P \sim_f Q$ iff $e \vdash P \sim_f^d Q$.

PROOF: By definition, any framed bisimulation is also a d-framed bisimulation. It therefore suffices to establish that $e \vdash P \sim_f Q$ whenever $e \vdash P \sim_f^d Q$. We show that

$$\mathcal{R} = \left\{ (e, P, Q) \mid \exists e', A, B, \vec{M}, \vec{N}. \begin{array}{l} P = A[\vec{M}/\vec{x}], Q = B[\vec{N}/\vec{y}] \\ e' \vdash A[\vec{M}'/\vec{x}] \sim_f^n B[\vec{N}'/\vec{y}] \\ (e', (\vec{M}', \vec{N}')) = \mathrm{pr}_d((e, \vec{M}, \vec{N})) \\ d = \mathrm{cd}(e, P, Q) \end{array} \right\}$$

is a framed bisimulation. This follows from Lemma 25. □

5.2 Matching output transitions

Next, we have to deal with matching output transitions. Fortunately, there are only finitely many candidates for an environment extension in the case of the output clause.

Unfortunately, as was shown in [BN02], the characterization of framed bisimilarity presented in [EHHN99] is sound but not complete. We are therefore unable to fall back on the algorithm for computing environment extensions presented in [EHHN99]. Instead we use

Lemma 27 Let $e \vdash \mathrm{ok}$ and let $M, N \in \mathcal{T}$. It is decidable whether there is an $e \leq e'$ such that $e \vdash M \leftrightarrow N$.

PROOF: To construct an e' such that $e' \vdash M \leftrightarrow N$, we only need to add pairs of the form (M_1, N_1) where $\max(\mathrm{d}(M_1), \mathrm{d}(N_1)) \leq \max(\mathrm{d}(M), \mathrm{d}(N))$ and such that $\mathrm{n}[\![M_1]\!] \cup \mathrm{n}[\![N_1]\!] \subseteq \mathrm{n}[\![M]\!] \cup \mathrm{n}[\![N]\!]$. Only finitely many such candidate pairs exist. □

6 Deciding framed bisimilarity

We can now state the main results of our paper.

Theorem 28 Let $e \vdash \mathrm{ok}$ and let P and Q be finite spi-calculus processes. For any $d \geq 0$ it is decidable whether $e \vdash P \sim_f^d Q$.

PROOF: Table 6 presents a nondeterministic recursive algorithm $\mathcal{B}((e, (P, Q))$ for determining if $e \vdash P \sim_f^d Q$.

As the algorithm encodes the 'bisimulation game' of Definition 16, $e \vdash P \sim_f^d Q$ iff there exists a successful evaluation of $\mathcal{B}((e, (P, Q)))$. The algorithm always terminates, as Lemma 15 and Lemma 27 guarantee that the checks performed in the conditional statements of the algorithm are effective

$\mathcal{B}(((fr, th), (\mathbf{0}, \mathbf{0}))) = \mathtt{tt}$

$\mathcal{B}(((fr, th), (P_1, P_2))) =$

let $(fr, th) = e$ **in**

for each $P_i \xrightarrow{a} (x)P_i'$ where $a \in fr$

select a $P_{i+1} \xrightarrow{a} (y)P_{i+1}'$

if no such P_{i+1}' exists

then *fail*

else

for each $\vec{n}$ where $|\vec{n}| \leq d, \vec{n} \cap \mathrm{fn}[\![P_i]\!] \cup \mathrm{fn}[\![P_{i+1}]\!] \cup \mathrm{fn}(th) = \emptyset$

for each $(fr \cup \{\vec{n}\}, th) \vdash M \leftrightarrow^d N$

$\mathcal{B}(((fr \cup \{\vec{n}\}, th), (P_i'[M/x], P_{i+1}'[N/y])))$

for each $P_i \xrightarrow{\overline{a}} (\nu \vec{c})\langle M \rangle P_i'$ where $a \in fr$

select a $P_{i+1} \xrightarrow{\overline{a}} (\nu \vec{d})\langle N \rangle P_{i+1}'$

if no such P_{i+1}' exists

then *fail*

else

select $e \leq (fr', th')$ such that $(fr', th') \vdash M \leftrightarrow N$

$\mathcal{B}(((fr', th'), P_i', P_{i+1}'))$

for each $P_i \xrightarrow{\tau} P_i'$

select a $P_{i+1} \xrightarrow{\tau} P_{i+1}'$

if no such P_{i+1}' exists

then *fail*

else

$\mathcal{B}(((fr, th), P_i', P_{i+1}'))$

Table 6: A nondeterministic algorithm for checking bisimilarity

and as all transition sequences examined along recursive calls are finite due to the absence of recursion. □

Corollary 29 Let $e \vdash \mathrm{ok}$ and let P and Q be finite spi-calculus processes. It is decidable whether $e \vdash P \sim_f Q$.

7 Conclusions and further work

In this paper we have shown that framed bisimilarity is decidable for finite processes. The ideas used in this paper are closely related to those employed in giving *symbolic semantics* to process calculi. The precise relationship is a topic for further work.

Recent, currently unpublished results [FN01, BN02] establish that the *en-*

vironment sensitive bisimilarity of Boreale et al. [BDP99] corresponds to a variant of framed bisimilarity which omits the frame-component. We therefore conjecture that our results and techniques carry over to environment sensitive bisimilarity.

A topic for further work is how to develop an efficient version of the bisimulation checking algorithm. However, framed bisimulation subsumes the late bisimulation equivalence of the π-calculus and the decision problem for this latter equivalence is known to be PSPACE-complete for a number of recursion-free process calculi with value-passing [BT00].

As we have omitted recursion, we can only study attacks that involve a given number of runs of a protocol. Another topic for further work is therefore to study the class of attacks that can be detected within the finite spi-calculus.

References

[AG97a] M. Abadi and A. D. Gordon. A calculus for cryptographic protocols: The spi-calculus. In *Fourth ACM Conference on Computer and Communications Security*. ACM Press, 1997.

[AG98b] M. Abadi and A. D. Gordon. A bisimulation method for cryptographic protocols. *Nordic Journal of Computing*, 5(4), pp. 267-303, Winter 1998.

[AL00] R.M. Amadio, D. Lugiez. On the reachability problem in cryptographic protocols *Proceedings of CONCUR 00*, LNCS 1877, Springer-Verlag.

[BDP99] Boreale, Michele & De Nicola, Rocco & Pugliese, Rosario. Proof Techniques for Cryptographic Processes (Extended version). *Proceedings of LICS 99*, pp. 157–166, 1999.

[BT00] M. Boreale and L. Trevisan A Complexity Analysis of Bisimilarity for Value-passing Processes *Theoretical Computer Science*, Vol. 238, Number 1-2, pp. 313-345, May 2000.

[BN02] J. Borgström and U. Nestmann On Bisimulations for the Spi Calculus Submitted for publication.

[Dam97] Mads Dam On the Decidability of Process Equivalences for the π-Calculus *Theoretical Computer Science*, vol. 183, 1997, pp. 215–228.

[EHHN99] A.S. Elkjær, H. Hüttel, M. Höhle and K. O. Nielsen. Towards automatic bisimilarity checking in the spi calculus. Proceedings of DMTCS'99 and CATS'99. *Australian Computer Science Communications*, 21(3), Springer, 1999.

[FN01] U. Frendrup and J. Nyholm Jensen. Bisimilarity in the Spi-Calculus Masters' Thesis, Department of Computer Science, Aalborg University, June 2001.

[HKNV97] Hans Hüttel, Josva Kleist, Uwe Nestmann and Björn Victor. A symbolic semantics for the spi-calculus. Unpublished manuscript.

[Lin91] Huimin Lin Complete Proof Systems for Observation Congruences in Finite-Control Pi-calculus. In Kim G. Larsen and Mogens Nielsen (editors), *Automata, Languages and Programming, 25th Colloquium*. Volume 1443 of *Lecture Notes in Computer Science*, pages 443-454, Aalborg, Denmark, July 1998, Springer-Verlag.

[MPW92] Robin Milner, Joachim Parrow and David Walker. A Calculus of Mobile Processes, Parts I and II. *Information and Computation*, vol. 100(1), 1992, pp. 1–77.

[Mil99] Robin Milner. Communicating and mobile systems: the π-calculus. Cambridge University Press, 1999.

[Min67] Marvin Minsky. Computation: Finite and Infinite Machines. Prentice-Hall 1967.

[NH83] Rocco De Nicola and Matthew C. B. Hennessy. Testing equivalence for processes. In Josep Díaz (editor) *Automata, Languages and Programming, 10th Colloquium*. Volume 154 of *Lecture Notes in Computer Science*, pages 548–560, Barcelona, Spain, 18–22 July 1983. Springer-Verlag.

[San96] Davide Sangiorgi. A theory of bisimulation for the π-calculus. *Acta Informatica*, vol. 33, 1996, pp. 69–97.

Recent BRICS Report Series Publications

RS-02-25 Hans Hüttel. *Deciding Framed Bisimilarity*. May 2002. 20 pp.

RS-02-24 Aske Simon Christensen, Anders Møller, and Michael I. Schwartzbach. *Static Analysis of Dynamic XML*. May 2002.

RS-02-23 Antonio Di Nola and Laurenţiu Leuştean. *Compact Representations of BL-Algebras*. May 2002. 25 pp.

RS-02-22 Mogens Nielsen, Catuscia Palamidessi, and Frank D. Valencia. *On the Expressive Power of Concurrent Constraint Programming Languages*. May 2002. 34 pp.

RS-02-21 Zoltán Ésik and Werner Kuich. *Formal Tree Series*. April 2002. 66 pp.

RS-02-20 Zoltán Ésik and Kim G. Larsen. *Regular Languages Definable by Lindström Quantifiers (Preliminary Version)*. April 2002. 56 pp.

RS-02-19 Stephen L. Bloom and Zoltán Ésik. *An Extension Theorem with an Application to Formal Tree Series*. April 2002. 51 pp.

RS-02-18 Gerth Stølting Brodal and Rolf Fagerberg. *Cache Oblivious Distribution Sweeping*. April 2002. To appear in *29th International Colloquium on Automata, Languages, and Programming*, ICALP '02 Proceedings, LNCS, 2002.

RS-02-17 Bolette Ammitzbøll Madsen, Jesper Makholm Nielsen, and Bjarke Skjernaa. *On the Number of Maximal Bipartite Subgraphs of a Graph*. April 2002. 7 pp.

RS-02-16 Jiří Srba. *Strong Bisimilarity of Simple Process Algebras: Complexity Lower Bounds*. April 2002. To appear in *29th International Colloquium on Automata, Languages, and Programming*, ICALP '02 Proceedings, LNCS, 2002.

RS-02-15 Jesper Makholm Nielsen. *On the Number of Maximal Independent Sets in a Graph*. April 2002. 10 pp.

RS-02-14 Ulrich Berger and Paulo B. Oliva. *Modified Bar Recursion*. April 2002. 23 pp.