Formalising Observer Theory for Environment-Sensitive Bisimulation

Jeremy E. Dawson and Alwen Tiu

Logic and Computation Group
College of Engineering and Computer Science
Australian National University
Canberra ACT 0200, Australia
http://users.rsise.anu.edu.au/~jeremy/
http://users.rsise.anu.edu.au/~tiu/

Abstract. We consider a formalisation of a notion of observer (or intruder) theories, commonly used in symbolic analysis of security protocols. An observer theory describes the knowledge and capabilities of an observer, and can be given a formal account using deductive systems, such as those used in various "environment-sensitive" bisimulation for process calculi, e.g., the spi-calculus. Two notions are critical to the correctness of such formalisations and the effectiveness of symbolic techniques based on them: decidability of message deduction by the observer and consistency of a given observer theory. We consider a formalisation, in Isabelle/HOL, of both notions based on an encoding of observer theories as pairs of symbolic traces. This encoding has recently been used in a theory of open bisimulation for the spi-calculus. We machine-checked some important properties, including decidability of observer deduction and consistency, and some key steps which are crucial to the automation of open bisimulation checking for the spi-calculus, and highlight some novelty in our Isabelle/HOL formalisations of decidability proofs.

1 Introduction

In most symbolic techniques for reasoning about security protocols, certain assumptions are often made concerning the capability of an intruder that tries to compromise the protocols. A well-known model of intruder is the so-called Dolev-Yao model [10], which assumes perfect crytography. We consider here a formal account of Dolev-Yao intruder model, formalised as some sort of deduction system. This deductive formulation is used in formalisations of various "environment-sensitive" bisimulations (see e.g., [6]) for process calculi designed for modeling security protocols, such as the spi-calculus [3]. An environment-sensitive bisimulation is a bisimulation relation which is indexed by a structure representing the intruder's knowledge, which we call an *observer theory.*

An important line of work related to the spi-calculus, or process calculi in general, is that of automating bisimulation checking. The transition semantics of these calculi often involve processes with infinite branching (e.g., transitions for input-prefixed processes in the π-calculus [12]), and therefore a *symbolic method* is needed to deal with potential infinite branches lazily. The resulting bisimulation, called *symbolic bisimulation*, has been developed for the spi-calculus [7]. The work reported in [7] is, however, only aimed at finding an effective approximation of environment-sensitive bisimulation, and there has been no metatheory developed for this symbolic bisimulation so far. A recent work by the second author [14] attempts just that: to establish a symbolic bisimulation that has good metatheory, in particular, a symbolic bisimulation which is also a congruence. The latter is also called *open bisimulation* [13]. One important part of the formulation of open bisimulation for the spi-calculus is a symbolic representation of observer theories, which needs to satisfy certain consistency properties, in addition to closure under a certain notion of "respectful substitutions", as typical in formulations of open bisimulation.

A large part of the work on open bisimulation in [14] deals with establishing properties of observer theories and their symbolic counterparts. This paper is essentially about formally verifying the results of [14] concerning properties of (symbolic) observer theories in Isabelle/HOL. In particular, it is concerned with proving decidability of the deduction system for observer theory, correctness of a finite characterisation of consistency of observer theories (hence decidability of consistency of observer theories), and preservation of consistency under respectful substitutions. Additionally, we also verify some key steps towards a decision procedure for checking consistency of symbolic observer theories, which is needed in automation of open bisimulation. A substantial formalisation work described here concerns decidability proofs. Such proofs are difficult to formalise in Isabelle/HOL, as noted in [17], due to the fact that Isabelle/HOL is based on classical logic. We essentially follow [17] in that decidability in this case can be inferred straightforwardly by inspection on the way we define total functions corresponding to the decidability problems in question. That is, we show, by meta-level inspection, that the definitions of the functions do not introduce any infinite aspect and are therefore are finitely computable.

There is a recent work [11] in formalising the spi-calculus and a notion of environment-sensitive bisimulation (called the *hedged bisimulation* [8]) in Isabelle/HOL. However, this notion of bisimulation is a "concrete" bisimulation (as opposed to symbolic), which means that the structure of observer theories is less involved and much easier to deal with compared to its symbolic counterpart. Our work on observer theories is mostly orthogonal to their work, and it can eventually be integrated into their formalisation to provide a completely formalised open bisimulation for the spi-calculus. Such an integration may not be too difficult, given that much of their work, e.g., formalisation of the operational semantics of the spi-calculus, can be reused without modifications.

We assume that the reader is familar with Isabelle proof assistant, its object logic HOL and logical frameworks in general. In the remainder of this section we briefly describe relevant Isabelle notations used throughout the paper. In Section 2, we give an overview of observer theories and an intuition behind them. We also give a brief description of two problems that will be the focus of subsequent sections, namely, those that concern decidability of consistency checking for (symbolic) observer theories. In Section 3 we consider formalisation of a notion of theory reduction and decidability of consistency checking for observer theories. In Section 4 we discuss a symbolic representation of observer theories using pairs of symbolic traces [5], called *bi-traces*, their consistency requirements and a notion of *respectful substitutions*. We prove a key lemma which relates a symbolic technique for trace refinement [5] to bi-traces, and discuss how this may lead to a decision procedure for testing bi-trace consistency. Section 5 concludes.

Isabelle notation The Isabelle codes for the results of this paper can be found at `http://users.rsise.anu.edu.au/~jeremy/isabelle/2005/spi/`. In the statement of lemma or theorem, a name given in `typewriter` font indicates the name of the relevant theorem in our Isabelle development. We show selected theorems and definitions in the text, and more in the Appendix. A version of the paper, including the Appendix, is in `http://users.rsise.anu.edu.au/~jeremy/pubs/spi/fotesb/`. So now we indicate some key points of the Isabelle notation.

- A name preceded by `?` indicates a variable: other names are entities which have been defined as part of the theory
- Conclusion β depending on assumptions α_i is `[|` α_1`;`α_2`;` $\ldots$ `;`α_n `|] ==>` β
- $\forall$, $\exists$ are written as `ALL`, `EX`
- $\subseteq$, $\supseteq$, $\in$ are written as `<=`, `>=`, `:`

2 Observer theory

An observer theory describes the knowledge accumulated by an observer in its interaction with a process (in the form of messages sent over networks), and its capability in analyzing and synthesizing messages. Since messages can be encrypted, and the encryption key may be unknown to the observer, it is not always the case that the observer can decompose all messages sent over the networks. In the presence of an active intruder, the traditional notion of bisimulation is not fine grained enough to prove interesting equivalence of protocols. A notion of bisimulation in which the knowledge and capability of the intruder is taken into account is often called an *environment-sensitive bisimulation.*

Messages are expressions formed from names, pairing constructor, e.g., $\langle M, N\rangle$, and symmetric encryption, e.g., $\{M\}_K$, where K is the encryption key and M is the message being encrypted. Note that we restrict to pairing and encryption to simplify discussion; there is no difficulty in extending the set of messages to include other constructors, including asymmetric encryption, natural numbers, etc. For technical reasons, we shall distinguish two kinds of names: *flexible names* and *rigid names.* We shall refer to flexible names as simply names. Names will be denoted with lower-case letters, e.g., a, x, y, etc., and rigid names will be denoted with bold letters, e.g., $\mathbf{a}$, $\mathbf{b}$, etc. We let $\mathcal{N}$ denote the set of names and $\mathcal{N}^=$ denote the set of pairs (x, x) of the same name. A name is really just a variable, i.e., a site for substitutions, and rigid names are just constants. This slightly different terminology is to conform with a "tradition" in name-passing process calculi where names are sometimes confused with variables (see e.g., [13]). In the context of open bisimulation for the spi-calculus [14], names stand for undetermined messages which can be synthesized by the observer.

There are two aspects of an observer theory which are relevant to bisimulation methods for protocols verification (for a more detailed discussion, see, e.g., [2]):

- *Message analysis and synthesis*: This is often formalised as a deduction system with judgments of the form $\Sigma \vdash M$, where Σ is a set of messages and M is a message. The intuitive meaning is that the observer can derive M given Σ. The deduction system is given in Figure 1 using sequent calculus. The usual formulation is based on natural deduction, but there is an easy correspondence between the two presentations (see [16] for details). One can derive, for example, that $\Sigma \vdash M$ holds if $\Sigma \vdash \{M\}_K$ and $\Sigma \vdash K$ hold, i.e., if the observer can derive $\{M\}_K$ and the key K, then it can derive M.
- *Indistinguishability of messages*: This notion arises when an observer tries to differentiate two processes based on the messages output by the processes. In the presence of encryption, indistinguishability does not simply mean syntactic equality. The judgment of interest in this case takes the form $\Gamma \vdash M \leftrightarrow N$ where Γ is a finite set of pairs of messages. It means, intuitively, that the observer cannot distinguish between M and N, given the *indistinguishability assumption* Γ. We shall not go into detailed discussion on this notion of indistinguishability; it has been discussed extensively in the literature [2, 8, 6, 14]. Instead we give a proof system for message indistinguishability (or message equivalence) in Figure 2.

Note that there are some minor differences between the inference rules in Figure 1 and Figure 2 and those given in [14]. That is, the "principal" message pairs for the rules (*pl*) and (*el*) in [14], $(\langle M_a, M_b\rangle, \langle N_a, N_b\rangle)$ and $(\{M_p\}_{M_k}, \{N_p\}_{N_k})$, are also in the premises. We proved that the alternative system is equivalent and that, in both systems, weakening on the left of $\vdash$ is admissible: see Appendix A.1.

We note that, by a cut-admissibility-like result, it is possible to further remove (M_k, N_k) from the second premise of (*el*): see Appendix A.2.

Subsequent results in this paper are concerned mainly with the above notion of indistinguishability. We therefore identify an *observer theory* with its underlying indistinguishability assumptions (i.e., Γ in the second item above). Hence, from now on, an observer theory (or theory) is a just finite set of pairs of messages, and will be denoted with Γ. Given a theory Γ, we write $\pi_1(\Gamma)$ to denote the set obtained by projecting on the first components of the pairs in Γ. The set $\pi_2(\Gamma)$ is defined analogously.

Observer theory consistency: An important notion in the theory of environment sensitive bisimulation is that of *consistency* of an observer theory. This amounts to the requirement that any observation (i.e., any "destructive" operations related to constructors of the messages, e.g., projection, decryption) that is applicable to the first projection of the theory is also applicable to the second projection. For example, the theory $\{(\{\mathbf{a}\}_\mathbf{b}, \{\mathbf{c}\}_\mathbf{d}), (\mathbf{b}, \mathbf{c})\}$ is not consistent, since on the first projection (i.e., the set $\{\{\mathbf{a}\}_\mathbf{b}, \mathbf{b}\}$), one can decrypt the first message $\{\mathbf{a}\}_\mathbf{b}$ using the second message $\mathbf{b}$, but the same operation cannot be done on

$$\frac{x \in \mathcal{N}}{\Sigma \vdash x}\ (var) \qquad \frac{}{\Sigma, M \vdash M}\ (id) \qquad \frac{\Sigma \vdash M \quad \Sigma \vdash N}{\Sigma \vdash \langle M, N\rangle}\ (pr)$$

$$\frac{\Sigma \vdash M \quad \Sigma \vdash N}{\Sigma \vdash \{M\}_N}\ (er) \qquad \frac{\Sigma, M, N \vdash R}{\Sigma, \langle M, N\rangle \vdash R}\ (pl) \qquad \frac{\Sigma \vdash N \quad \Sigma, M, N \vdash R}{\Sigma, \{M\}_N \vdash R}\ (el)$$

Fig. 1. A proof system for message synthesis

$$\frac{x \in \mathcal{N}}{\Gamma \vdash x \leftrightarrow x}\ (var) \qquad \frac{(M, N) \in \Gamma}{\Gamma \vdash M \leftrightarrow N}\ (id)$$

$$\frac{\Gamma \vdash M_a \leftrightarrow N_a \quad \Gamma \vdash M_b \leftrightarrow N_b}{\Gamma \vdash \langle M_a, M_b\rangle \leftrightarrow \langle N_a, N_b\rangle}\ (pr) \qquad \frac{\Gamma \vdash M_p \leftrightarrow N_p \quad \Gamma \vdash M_k \leftrightarrow N_k}{\Gamma \vdash \{M_p\}_{M_k} \leftrightarrow \{N_p\}_{N_k}}\ (er)$$

$$\frac{\Gamma, (M_a, N_a), (M_b, N_b) \vdash M \leftrightarrow N}{\Gamma, (\langle M_a, M_b\rangle, \langle N_a, N_b\rangle) \vdash M \leftrightarrow N}\ (pl)$$

$$\frac{\Gamma \vdash M_k \leftrightarrow N_k \quad \Gamma, (M_p, N_p), (M_k, N_k) \vdash M \leftrightarrow N}{\Gamma, (\{M_p\}_{M_k}, \{N_p\}_{N_k}) \vdash M \leftrightarrow N}\ (el)$$

Fig. 2. A proof system for deducing message equivalence

the second projection. The formal definition of consistency involves checking all message pairs (M, N) such that $\Gamma \vdash M \leftrightarrow N$ is derivable for certain similarity of observations. The first part of this paper is about verifying that this infinite quantification is not necessary. This involves showing that for every theory Γ, there is a corresponding *reduced theory* that is equivalent, but for which consistency checking requires only checking finitely many message pairs.

Symbolic observer theory: The definition of open bisimulation for name-passing calculi, such as the π-calculus, typically includes closure under a certain notion of *respectful substitutions* [13]. In the π-calculus, this notion of respectfulness is defined w.r.t. to a notion of *distinction* among names, i.e., an irreflexive relation on names which forbids identification of certain names. In the case of the spi-calculus, things get more complicated because the bisimulation relation is indexed by an observer theory, not just a simple distinction on names. We need to define a symbolic representation of observer theories, and an appropriate notion of consistency for the symbolic theories. These are addressed in [14] via a structure called *bi-traces*. A bi-trace is essentially a list of pairs of messages. It can be seen as a pair of symbolic traces, in the sense of [5]. The order of the message pairs in the list indicates the order of their creation (i.e., by the intruder or by the processes themselves). Names in a bi-trace indicate undetermined messages, which are open to instantiations. Therefore the notion of consistency of bi-traces needs to take into account these possible instantiations. Consider the following sequence of message pairs: $(\mathbf{a}, \mathbf{d}), (\{\mathbf{a}\}_\mathbf{b}, \{\mathbf{d}\}_\mathbf{k}), (\{\mathbf{c}\}_{\{x\}_\mathbf{b}}, \{\mathbf{k}\}_\mathbf{l})$. Considered as a theory, it is consistent, since none of the encryption keys are known to the observer. However, if we allow x to be instantiated to a, then the resulting theory $\{(\mathbf{a}, \mathbf{d}), (\{\mathbf{a}\}_\mathbf{b}, \{\mathbf{d}\}_\mathbf{k}), (\{\mathbf{c}\}_{\{\mathbf{a}\}_\mathbf{b}}, \{\mathbf{k}\}_\mathbf{l})\}$ is inconsistent, since on the first projection, $\{\mathbf{a}\}_\mathbf{b}$ can be used as a key to decrypt $\{\mathbf{c}\}_{\{\mathbf{a}\}_\mathbf{b}}$, while in the second projection, no decryption is possible. Therefore to check consistency of a bi-trace, one needs to consider potentially infinitely many instances of the bi-trace. Section 4 shows some key steps to simplify consistency checking for bi-traces.

3 Observer theory reduction and consistency

We now discuss our formalisation of observer theory and its consistency properties in Isabelle/HOL.

The datatype for messages is represented in Isabelle/HOL as follows.

```
datatype msg = Name nat | Rigid nat | Mpair msg msg | Enc msg msg
```

A observer theory, as already noted, is a finite set of pairs of messages. In Isabelle, we just use a set of pairs, so the finiteness condition appears in the Isabelle statements of many theorems. The judgment $\Gamma \vdash M \leftrightarrow N$ is represented by $(\Gamma, (M, N))$, or, equivalently in Isabelle, (Γ, M, N).

In Isabelle we define, inductively, a set of sequents `indist` which is the set of sequents derivable in the proof system for message equivalence (Figure 2). Subsequently we found it helpful to define the corresponding set of rules explicitly, calling them `indpsc`. The rules for message synthesis, given in Figure 1, are just a projection to one component of the rule set `indpsc`; we call this projection `smpsc`. It is straightforward to extend the notion of a projection on rule sets, so we can define the rules for message synthesis as simply `smpsc` $= \pi_1($`indpsc`$)$. The formal expression in Isabelle is more complex: see Appendix A.3. Likewise, we write $pair(X)$ to turn each message M into the pair (M, M) in a theory, sequent, rule or bi-trace X.

The following lemma relates message synthesis and message equivalence. Lemma 1(d) depends on theory consistency, to be introduced later.

Lemma 1. *(a)* `(smpsc_alt)` *Rule* $R \in$ *`smpsc`* *iff* $pair(R) \in$ *`indpsc`*
(b) `(slice_derrec_smpsc_empty)` *if* $\Gamma \vdash M \leftrightarrow N$ *then* $\pi_1(\Gamma) \vdash M$
(c) `(derrec_smpsc_eq)` $\Sigma \vdash M$ *if and only if* $pair(\Sigma) \vdash M \leftrightarrow M$
(d) `(smpsc_ex_indpsc_der)` *if* $\pi_1(\Gamma) \vdash M$ *and* Γ *is consistent, then there exists* N *such that* $\Gamma \vdash M \leftrightarrow N$

3.1 Decidability of $\vdash$ and computability of theory reduction

The first step towards deciding theory consistency is to define a notion of *theory reduction.* Its purpose is to extract a "kernel" of the theory with no redundancy, that is, no pairs in the kernel are derivable from the others. We need to establish the decidability of $\vdash$, and then termination of the theory reduction. In [14], Tiu observes that $\Gamma \vdash M \leftrightarrow N$ is decidable, because the right rules (working upwards) make the right-hand side messages smaller, and the left rules saturate the antecedent theory with more pairs of smaller messages. Hence for a given end sequent, there are only finitely many possible sequents which can appear in any proof of the sequent. Some results relevant to this argument for decidability are presented in Appendix A.4. Here we present an alternative proof for the decidability of $\vdash$ and termination of theory reduction.

Tiu [14, Definition 4] defines a reduction relation of observer theories:

$$\begin{aligned} \Gamma, (\langle M_a, M_b\rangle, \langle N_a, N_b\rangle) &\longrightarrow \Gamma, (M_a, N_a), (M_b, N_b) \\ \Gamma, (\{M_p\}_{M_k}, \{N_p\}_{N_k}) &\longrightarrow \Gamma, (M_p, N_p), (M_k, N_k) \\ &\qquad \text{if } \Gamma, (\{M_p\}_{M_k}, \{N_p\}_{N_k}) \vdash M_k \leftrightarrow N_k \end{aligned}$$

We assume that Γ does not contain $(\langle M_a, M_b\rangle, \langle N_a, N_b\rangle)$ and $(\{M_p\}_{M_k}, \{N_p\}_{N_k})$ respectively (otherwise reduction would not terminate). This reduction relation is terminating and confluent, and so every theory Γ reduces to a unique normal form $\Gamma\Downarrow$. It also preserves the entailment $\vdash$.

Lemma 2. *(a) [15, Lemma 15]* `(red_nc)` *If $\Gamma \longrightarrow \Gamma'$ then $\Gamma \vdash M \leftrightarrow N$ if and only if $\Gamma' \vdash M \leftrightarrow N$*
(b) `(nf_nc)` *Assuming that $\Gamma\Downarrow$ exists, $\Gamma \vdash M \leftrightarrow N$ if and only if $\Gamma\Downarrow \vdash M \leftrightarrow N$*

It is easy to show that $\longrightarrow$ is well-founded, since the sum of the sizes of [the first member of each of] the message pairs reduces each time. Confluence is reasonably easy to see since the side condition for the second rule is of the form $\Gamma' \vdash M_k \leftrightarrow N_k$ where Γ' is exactly the theory being reduced, and, from Lemma 2, this condition (for a particular M_k, N_k) will continue to hold, or not, when other reductions have changed Γ'. Actually, proving confluence in Isabelle was not so easy, and we describe the difficulty and our proof in Appendix A.6. Then it is a standard result, and easy in Isabelle, that confluence and termination give normal forms.

Theorem 3 **(**`nf_oth_red`**).** *Any theory Γ has a $\longrightarrow$-normal form $\Gamma\Downarrow$.*

A different reduction relation As a result of Lemma 2, to decide whether $\Gamma \vdash M \leftrightarrow N$ one might calculate $\Gamma\Downarrow$ and determine whether $\Gamma\Downarrow \vdash M \leftrightarrow N$, which is easier (see Lemma 5). However to calculate $\Gamma\Downarrow$ requires determining whether $\Gamma \vdash M_k \leftrightarrow N_k$, so the decidability of this procedure is not obvious.

We defined an alternative version, $\longrightarrow'$, of the reduction relation, by changing the condition in the second rule, so our new relation is:

$$\begin{aligned} \Gamma, (\langle M_a, M_b\rangle, \langle N_a, N_b\rangle) &\longrightarrow' \Gamma, (M_a, N_a), (M_b, N_b) \\ \Gamma, (\{M_p\}_{M_k}, \{N_p\}_{N_k}) &\longrightarrow' \Gamma, (M_p, N_p), (M_k, N_k) \quad \text{if } \Gamma \vdash M_k \leftrightarrow N_k \end{aligned}$$

This definition does not give the same relation, but we are able to show that the two relations have the same normal forms. Using this reduction relation, the procedure to decide whether $\Gamma \vdash M \leftrightarrow N$ is: calculate $\Gamma\Downarrow$ and determine whether $\Gamma\Downarrow \vdash M \leftrightarrow N$. Calculating $\Gamma\Downarrow$ requires deciding questions of the form $\Gamma' \vdash M_k \leftrightarrow N_k$, where Γ' is smaller than Γ (because a pair $(\{M_p\}_{M_k}, \{N_p\}_{N_k})$ is omitted). Thus this procedure terminates.

Note that Lemma 2 also holds for $\longrightarrow'$ since $\longrightarrow' \subseteq \longrightarrow$.

To show the two relations have the same normal forms, we first show (in Theorem 4(b)) that if Γ is $\longrightarrow$-reducible, then it is $\longrightarrow'$-reducible, even though the same reduction may not be available.

Theorem 4. *(a)* `(red_alt_lem)` *If $\Gamma \vdash M_k \leftrightarrow N_k$ then either $\Gamma \setminus \{(\{M_p\}_{M_k}, \{N_p\}_{N_k})\} \vdash M_k \leftrightarrow N_k$ or there exists Γ' such that $\Gamma \longrightarrow' \Gamma'$*
(b) `(oth_red_alt_lem)` *If $\Gamma \longrightarrow \Delta$ then there exists Δ' such that $\Gamma \longrightarrow' \Delta'$*
(c) `(rsmin_or_alt)` *If Γ is $\longrightarrow'$-minimal (i.e., cannot be reduced further) then Γ is $\longrightarrow$-minimal*
(d) `(nf_acc_alt)` *$\Gamma \longrightarrow' \Gamma\Downarrow$ (where $\Gamma\Downarrow$ is the $\longrightarrow$-normal form of Γ)*
(e) `(nf_alt, nf_same)` *$\Gamma\Downarrow$ is also the $\longrightarrow'$-normal form of Γ*

Proof. We show a proof of (a) here. We prove a stronger result namely: If $\Gamma \vdash M \leftrightarrow N$ and $size\ M \leq size\ Q_k$ then either $\Gamma' = \Gamma \setminus \{(\{Q_p\}_{Q_k}, \{R_p\}_{R_k})\} \vdash M \leftrightarrow N$ or there exists Δ such that $\Gamma \longrightarrow' \Delta$.

We prove it by induction on the derivation of $\Gamma \vdash M \leftrightarrow N$. If the derivation is by the (var) rule, ie, $(M, N) = (x, x)$, then clearly $\Gamma' \vdash M \leftrightarrow N$ by the (var) rule. If the derivation is by the (id) rule, ie, $(M, N) \in \Gamma$, then the size condition shows that $(M, N) \in \Gamma'$, and so $\Gamma' \vdash M \leftrightarrow N$ by the (id) rule.

If the derivation is by either of the right rules (pr) or (er), then we have $\Gamma \vdash M' \leftrightarrow N'$ and $\Gamma \vdash M'' \leftrightarrow N''$, according to the rule used, with M' and M'' smaller than M. Then, unless $\Gamma \longrightarrow' \Delta$ for some Δ, we have by induction $\Gamma' \vdash M' \leftrightarrow N'$ and $\Gamma' \vdash M'' \leftrightarrow N''$, whence, by the same right rule, $\Gamma' \vdash M \leftrightarrow N$.

If the derivation is by the left rule (pl), then $\Gamma \longrightarrow' \Delta$ for some Δ.

If the derivation is by the left rule (el), then we apply the inductive hypothesis to the *first* premise of the rule. Let the "principal" message pair for the rule be $(\{M_p\}_{M_k}, \{N_p\}_{N_k})$, so the first premise is $\Gamma \vdash M_k \leftrightarrow N_k$. Note that we apply the inductive hypothesis to a possibly *different* pair of encrypts in Γ, namely $(\{M_p\}_{M_k}, \{N_p\}_{N_k})$ instead of $(\{Q_p\}_{Q_k}, \{R_p\}_{R_k})$.

By induction, either $\Gamma \longrightarrow' \Delta$ for some Δ or (since $size\ M_k \leq size\ M_k$), we have $\Gamma \setminus \{(\{M_p\}_{M_k}, \{N_p\}_{N_k})\} \vdash M_k \leftrightarrow N_k$. Then we have $\Gamma \longrightarrow' \Delta$, as required, where $\Delta = \Gamma \setminus \{(\{M_p\}_{M_k}, \{N_p\}_{N_k})\}, (M_p, N_p), (M_k, N_k)$. □

Since the process of reducing a theory essentially replaces pairs of compound messages with more pairs of simpler messages, this suggests that to show that $\Gamma \vdash M \leftrightarrow N$ for a reduced Γ, one need only use the rules which build up pairs of compound messages on the right. That is, one would use the right rules (pr) and (er), but not the left rules (pl) and (el). Let us define $\Gamma \vdash_r M \leftrightarrow N$ to mean that $\Gamma \vdash M \leftrightarrow N$ can be derived using the rules $(var), (id), (pr)$ and (er) of Figure 2. We call the set of these rules `indpsc_virt`.

We define a function `is_der_virt` which shows how to test $\Gamma \vdash_r M \leftrightarrow N$, and, in Lemma 5(b), prove that it does this. It terminates because at each recursive call, the size of M gets smaller. When we define a function in this way, Isabelle requires termination to be *proved* (usually it can do this automatically). Then *inspection* of the function definition shows that, assuming the theory `oth` is finite, the function is finitely computable. We discuss this idea further later. We also define a simpler function `is_der_virt_red`, as an alternative to `is_der_virt`, which gives the same result when Γ is reduced, see Appendix A.13.

```
recdef "is_der_virt" "measure (%(oth, M, N). size M)"
  "is_der_virt (oth, Name i, Name j) = ((Name i, Name j) : oth | i = j)"
  "is_der_virt (oth, Mpair Ma Mb, Mpair Na Nb) =
    ((Mpair Ma Mb, Mpair Na Nb) : oth |
      is_der_virt (oth, Ma, Na) & is_der_virt (oth, Mb, Nb))"
  "is_der_virt (oth, Enc Mp Mk, Enc Np Nk) =
    ((Enc Mp Mk, Enc Np Nk) : oth |
      is_der_virt (oth, Mp, Np) & is_der_virt (oth, Mk, Nk))"
  "is_der_virt (oth, M, N) = ((M, N) : oth)"
```

Lemma 5. *(a)* (nf_no_left) *If Γ is reduced and $\Gamma \vdash M \leftrightarrow N$ then $\Gamma \vdash_r M \leftrightarrow N$*
(b) (virt_dec) *$\Gamma \vdash_r M \leftrightarrow N$ if and only if* `is_der_virt` $(\Gamma, (M, N))$

We can now define a function `reduce` which computes a $\longrightarrow'$-normal form.

```
recdef (permissive) "reduce" "measure (setsum (size o fst))"
  "reduce S = (if infinite S then S else
    let P = (%x. x : Mpairs <*> Mpairs & x : S) ;
      Q = (%x.  (if x : Encs <*> Encs & x : S then
          is_der_virt (reduce (S - {x}), keys x) else False))
    in if Ex P then reduce (red_pair (Eps P) (S - {Eps P}))
      else if Ex Q then reduce (red_enc (Eps Q) (S - {Eps Q}))
      else S)"
```

To explain this: $P\ (M, N)$ means $(M, N) \in S$ and M, N are both pairs; $Q\ (M, N)$ means $(M, N) \in S$ and M, N are both encrypts, say $\{M_p\}_{M_k}, \{N_p\}_{N_k}$, where $S \setminus \{(M, N)\} \vdash_r (M_k, N_k)$; `red_pair` and `red_enc` do a single step reduction based on the message pairs or encrypts given as their argument, `Ex P` means $\exists x.\ P\ x$, and `Eps P` means some x satisfying P, if such exists. Thus the function selects arbitrarily a pair of message pairs or encrypts suitable for a single reduction step, performs that step, and then reduces the result.

The expression `measure (setsum (size o fst))` is the termination measure, the sum of the sizes of the first member of each message pair in a theory. The function `reduce` is recursive, and necessarily terminates since at each iteration this measure function, applied to the argument, is smaller. However this function definition is sufficiently complicated that Isabelle cannot automatically prove that it terminates — thus the notation `(permissive)` in the definition.

Isabelle produces a complex definition dependent on conditions that if we change a theory by applying `red_pair` or `red_enc`, or by deleting a pair, then we get a theory which is smaller according to the measure function. Since in the HOL logic of Isabelle all functions are total, we have a function `reduce` in any event; we need to prove the conditions to prove that `reduce` conforms to the definition given above. We then get Theorem 6(a) and (b), which show how to test $\Gamma \vdash M \leftrightarrow N$ as a manifestly finitely computable function. We also prove a useful characterisation of $\Gamma\Downarrow$.

Theorem 6. *(a)* (reduce_nf, reduce_nf_alt) `reduce` $\Gamma = \Gamma\Downarrow$
(b) (virt_reduce, idvr_reduce) *$\Gamma \vdash M \leftrightarrow N$ if and only if* `is_der_virt` $(\Gamma\Downarrow, (M, N))$*, equivalently, if and only if* `is_der_virt_red` $(\Gamma\Downarrow, (M, N))$

(c) `(reduce_alt)` *For* $(M,N) \notin \mathcal{N}^=$, $(M,N) \in \Gamma\Downarrow \setminus \mathcal{N}^=$ *iff*

(i) $\Gamma \vdash M \leftrightarrow N$,

(ii) M *and* N *are not both pairs, and*

(iii) *if* $M = \{M_p\}_{M_k}, N = \{N_p\}_{N_k}$, *then* $\Gamma \not\vdash M_k \leftrightarrow N_k$

As Urban et al point out in [17] formalising decidability — or computability — is difficult. It would require formalising the computation process, as distinct from simply defining the quantity to be computed. However, as is done in [17, §3.4], it is possible to define a quantity in a certain way which makes it reasonable to assert that it is computable. This is what we have aimed to do in defining the function `reduce`. It specifies the computation to be performed (with a caveat mentioned later). Isabelle requires us to show that this computation is terminating, and we have shown that it produces the $\longrightarrow'$-normal form. To ensure termination, we needed to base the definition of `reduce` on $\longrightarrow'$, not on $\longrightarrow$, but by Theorem 4(e), $\longrightarrow'$ and $\longrightarrow$ have the same normal forms.

Certain terminating functions are not obviously computable, for example $f\ x = (\exists y.\ P\ y)$ (even where P is computable). So our definition of `reduce` requires inspection to ensure that it contains nothing which makes it not computable. It does contain existential quantifiers, but they are in essence quantification over a finite set. The only problem is the subterms `Eps P` and `Eps Q`, that is $\epsilon x.\ P\ x$ and $\epsilon x.\ Q\ x$. These mean "some x satisfying P" (similarly Q). In Isabelle's logic, this means *some* x, but we have no knowledge of which one (and so we cannot perform precisely this computation). But our proofs went through without any knowledge of which x is denoted by $\epsilon x.\ P\ x$. Therefore it would be safe to implement a computation which makes any choice of $\epsilon x.\ P\ x$, and we can safely assert that our proofs would still hold for that computation.[1] That is, in general we assert that if a function involving $\epsilon x.\ P\ x$ can be proven to have some property, then a function which replaces $\epsilon x.\ P\ x$ by some other choice of x (satisfying P if possible) would also have that property. Based on this assertion we say that our definition of `reduce` shows that the $\longrightarrow$-normal form is computable, and so that $\Gamma \vdash M \leftrightarrow N$ is decidable.

We found that although the definition of `reduce` gives the function in a computable form, many proofs are much easier using the characterisation as the normal form. For example Lemma 2(b) is much easier using Lemma 2(a) than using the definition of `reduce`. We found this with some other results, such as: if Γ is finite, then so is $\Gamma\Downarrow$, and if Γ consists of identical pairs then so does $\Gamma\Downarrow$.

Since also Γ and $\Gamma\Downarrow$ entail the same message pairs, it is reasonable to ask which theories, other than those with the same normal form as Γ, entail the same message pairs as Γ. Now it is clear, due to the (*var*) rule, that deleting (x,x) from a theory does not change the set of entailed message pairs or the reductions available. However we find that the condition is that theories entail the same pairs iff their normal forms are equal, modulo $\mathcal{N}^=$.

We could further change $\longrightarrow'$ by deleting the (M_k, N_k) from the second rule. Lemma 2(a) holds for this new relation. For further discussion see Appendix A.7.

[1] In general a repeated choice must be made consistently; the HOL logic *does* imply $\epsilon x.\ P\ x = \epsilon x.\ P\ x$. This point clearly won't arise for the `reduce` function.

Theorem 7. *(a)* `(rsmin_names)` *Γ is reduced if and only if $\Gamma \setminus \mathcal{N}^=$ is reduced*
(b) [15, Lemma 8] `(name_equivd)` *$\Gamma \vdash M \leftrightarrow N$ if and only if $\Gamma \setminus \mathcal{N}^= \vdash M \leftrightarrow N$*
(c) `(nf_equiv_der)` *Theories Γ_1 and Γ_2 entail the same message pairs if and only if $\Gamma_1 \Downarrow \setminus \mathcal{N}^= = \Gamma_2 \Downarrow \setminus \mathcal{N}^=$*

3.2 Theory consistency

Definition 8. *[15, Definition 11] A theory Γ is* consistent *if for every M and N, if $\Gamma \vdash M \leftrightarrow N$ then the following hold:*

(a) M and N are of the same type of expressions, i.e., M is a pair (an encrypted message, a (rigid) name) if and only if N is.
(b) If $M = \{M_p\}_{M_k}$ and $N = \{N_p\}_{N_k}$ then $\pi_1(\Gamma) \vdash M_k$ implies $\Gamma \vdash M_k \leftrightarrow N_k$ and $\pi_2(\Gamma) \vdash N_k$ implies $\Gamma \vdash M_k \leftrightarrow N_k$.
(c) For any R, $\Gamma \vdash M \leftrightarrow R$ implies $R = N$ and $\Gamma \vdash R \leftrightarrow N$ implies $R = M$.

This definition of consistency involves infinite quantification. We want to eliminate this quantification by finding a finite characterisation on *reduced theories*. But first, let us define another equivalent notion of consistency, which is simpler for verification, as it does not use the deduction system for message synthesis.

Definition 9. *A theory Γ satisfies the predicate* `thy_cons` *if for every M and N, if $\Gamma \vdash M \leftrightarrow N$ then the following hold:*

(a) M and N are of the same type of expressions, i.e., as in Definition 8(a)
(b) for every M, N', M_p, N_p if $\Gamma \vdash M' \leftrightarrow N'$ or $\Gamma \vdash \{M_p\}_{M'} \leftrightarrow \{N_p\}_{N'}$, then $M' = M$ iff $N' = N$

Lemma 10. *(a)* `(thy_cons_equiv)` *Γ is consistent iff it satisfies Definition 9*
(b) `(thy_cons_equivd)` *Γ is consistent if and only if $\Gamma \setminus \mathcal{N}^=$ is consistent*
(c) [15, Lemma 19] `(nf_cons)` *Γ is consistent if and only if $\Gamma\Downarrow$ is consistent*
(d) `(cons_der_same)` *If Γ_1 and Γ_2 entail the same message pairs then Γ_1 is consistent if and only if Γ_2 is consistent*

Tiu [15, Proposition 20] gives a characterisation of consistency (reproduced below in Proposition 11) which is finitely checkable. In Definition 12 we define a predicate `thy_cons_red` which is somewhat similar. In Theorem 13 we show that, for a reduced theory, that our `thy_cons_red` is equivalent to consistency and to the conditions in Proposition 11. Decidability of consistency then follows from decidability of $\vdash$, and termination of normal form computation.

Proposition 11. *[15, Proposition 20] A theory Γ is consistent if and only if $\Gamma\Downarrow$ satisfies the following conditions: if $(M, N) \in \Gamma\Downarrow$ then*

(a) M and N are of the same type of expressions, in particular, if $M = x$, for some name x, then $N = x$ and vice versa,
(b) if $M = \{M_p\}_{M_k}$ and $N = \{N_p\}_{N_k}$ then $\pi_1(\Gamma\Downarrow) \not\vdash M_k$ and $\pi_2(\Gamma\Downarrow) \not\vdash N_k$.
(c) for any $(U, V) \in \Gamma\Downarrow$, $U = M$ if and only if $V = N$.

Definition 12. *A theory Γ satisfies the predicate* `thy_cons_red` *if*

(a) *for all $(M, N) \in \Gamma$, M and N satisfy Proposition 11(a)*
(b) *for all (M, N) and $(M', N') \in \Gamma$, $M' = M$ iff $N' = N$*
(c) *for all $(\{M_p\}_{M_k}, \{N_p\}_{N_k}) \in \Gamma$, for all M, N such that $\Gamma \vdash M \leftrightarrow N$, $M \neq M_k$ and $N \neq N_k$*

Theorem 13. *(a)* `(tc_red_iff)` *Γ is consistent iff $\Gamma\Downarrow$ satisfies* `thy_cons_red`
(b) `(thy_cons_red_equiv)` *$\Gamma\Downarrow$ satisifes Proposition 11(a) to (c) iff it satisfies* `thy_cons_red`*, ie, Definition 12(a) to (c)*

4 Respectful Substitutions and Bi-trace Consistency

We now consider a symbolic representation of observer theories from [14], given below. We denote with $fn(M)$ the set of names in M. This notation is extended straightforwardly to pairs of messages, lists of (pairs of) messages, etc.

Definition 14. *A* bi-trace *is a* list *of message pairs marked with i (indicating input) or o (output), i.e., elements in a bi-trace have the form $(M, N)^i$ or $(M, N)^o$. Bi-traces are ranged over by h. We denote with $\pi_1(h)$ the list obtained from h by taking the first component of the pairs in h. The list $\pi_2(h)$ is defined analogously. Bi-traces are subject to the following restriction: if $h = h_1.(M, N)^o.h_2$ then $fn(M, N) \subseteq fn(h_1)$. We write $\{h\}$ to denote the set of message pairs obtained from h by forgetting the marking and the order.*

Names in a bi-trace represent *symbolic values* which are input by a process at some point. This explains the requirement that the free names of an output pair in a bi-trace must appear before the output pair. We express this restriction on name occurrences by defining a predicate `validbt` on lists of marked message pairs, and we do not mention it in the statement of each result, although it does appear in their statements in Isabelle. In our Isabelle representation the list is reversed, so that the latest message pair is the first in the list. The theory $\{h\}$ obtained from a bi-trace h is represented by `oth_of` h. Likewise for a list s of marked messages (which can be seen as a symbolic trace [5]), we can define the set $\{s\}$ of messages by forgetting the annotations and ordering.

A substitution pair $\vec{\theta} = (\theta_1, \theta_2)$ replaces free names $x \in \mathcal{N}$ by messages, using substitutions $\theta_1(\theta_2)$ for the first (second) component of each pair. For a bi-trace h, $\vec{\theta}$ *respects* h, or is h-respectful [15, Definition 34], if for every free name x in an input pair $(M, N)^i$, $\{h'\}\vec{\theta} \vdash x\theta_1 \leftrightarrow x\theta_2$, where h' is the part of h preceding $(M, N)^i$. This is expressed in Isabelle by $h \in$ `bt_resp` $\vec{\theta}$.

Definition 15. *[15, Definition 35] The set of* consistent bi-traces *are defined inductively (on the length of bi-traces) as follows:*

(a) *The empty bi-trace is consistent.*
(b) *If h is a consistent bi-trace then $h.(M, N)^i$ is also a consistent bi-trace, provided that $h \vdash M \leftrightarrow N$.*

(c) *If h is a consistent bi-trace, then $h' = h.(M, N)^o$ is a consistent bi-trace, provided that for every h-respectful substitution pair $\vec{\theta}$, if $h\vec{\theta}$ is a consistent bi-trace then $\{h'\vec{\theta}\}$ is a consistent theory.*

Given Lemma 16(c) below, it may appear that leaving out the underlined words of Definition 15 would make no difference. This minor fact can indeed be proved formally: details are given in Appendix A.16.

The following are significant lemmas from [15] which we proved in Isabelle. As an illustration of the value of automated theorem proving, we found that the original proof of (b) in a draft of [15] contained an error (which was easily fixed).

Lemma 16. *(a) [15, Lemma 24]* (`subst_indist`) *Let $\Gamma \vdash M \leftrightarrow N$ and let $\vec{\theta} = (\theta_1, \theta_2)$ be a substitution pair such that for every free name x in Γ, M or N, $\Gamma\vec{\theta} \vdash \theta_1(x) \leftrightarrow \theta_2(x)$. Then $\Gamma\vec{\theta} \vdash M\theta_1 \leftrightarrow N\theta_2$.*

(b) [15, Lemma 40] (`bt_resp_comp`) *Let h be a consistent bi-trace, let $\vec{\theta} = (\theta_1, \theta_2)$ be an h-respectful substitution pair, and let $\vec{\gamma} = (\gamma_1, \gamma_2)$ be an $h\vec{\theta}$-respectful substitution pair. Then $\vec{\theta} \circ \vec{\gamma}$ is also h-respectful.*

(c) [15, Lemma 41] (`cons_subs_bt`) *If h is a consistent bi-trace and $\vec{\theta} = (\theta_1, \theta_2)$ respects h, then $h\vec{\theta}$ is also a consistent bi-trace.*

Respectfulness of a substitution relative to a theory Testing consistency of bi-traces involves testing whether a theory Γ is consistent after applying any respectful substitution pair $\vec{\theta}$ to it. We will present some results that (under certain conditions) if we reduce $\{h\}$ first, and then apply an h-respectful substitution, then the result is a reduced theory, to which the simpler test for consistency, `thy_cons_red`, applies.

The complication here is that reduction applies to a theory whereas the definition of bi-trace consistency crucially involves the ordering of the pairs of messages. We overcome this by devising the notion, `thy_strl_resp`, of a substitution being respectful with respect to an (unordered) theory and an ordered list of sets of variable names. Importantly, this property holds for $\{h\}$ where $\vec{\theta}$ is h-respectful, and it is preserved by reducing a theory. We use this to prove some later results involving $\{h\}\Downarrow$ and h-respectful substitutions, such as Theorem 17. Details are in Appendix A.19.

Simplifying testing consistency after substitution Recall that a theory Γ is consistent if and only if $\Gamma\Downarrow$ is consistent (Lemma 10(c)), and if and only if $\Gamma \backslash \mathcal{N}^=$ is consistent (Lemma 10(b)). Thus, to determine whether Γ is consistent, one may calculate $\Gamma\Downarrow$ or $\Gamma\Downarrow \backslash \mathcal{N}^=$ (which is reduced, by Lemma 7(a)), and use the function `thy_cons_red` (by virtue of Theorem 13(a)). Therefore, the naive approach to testing bi-trace consistency is to apply θ to Γ and then reduce the result, and delete pairs $(x, x) \in \mathcal{N}^=$. We can derive results which permit a simpler approach.

Theorem 17. *Let h be a bi-trace, and let $\Gamma = \{h\}$. Let $\vec{\theta}$ be an h-respectful substitution pair, and denote its action on Γ by $\vec{\theta}$ also.*

(a) (`nf_subst_nf_Ne`) $\Gamma\vec{\theta}\Downarrow \backslash \mathcal{N}^{=} = (\Gamma\Downarrow \backslash \mathcal{N}^{=})\vec{\theta}\Downarrow \backslash \mathcal{N}^{=}$
(b) (`subst_nf_Ne_tc`) $\Gamma\vec{\theta}$ *is consistent if and only if* $(\Gamma\Downarrow \backslash \mathcal{N}^{=})\vec{\theta}$ *is consistent*

This, given a bi-trace h and a respectful substitution pair $\vec{\theta}$, if one wants to test whether $\Gamma\vec{\theta} = \{h\vec{\theta}\}$ is consistent, it makes no difference to the consistency of the resulting theory if one reduces the theory and deletes pairs (x, x) before substituting. This means that we need only consider substitution in a theory which is reduced and has pairs (x, x) removed.

If we disallow encryption where keys are themselves pairs or encrypts, then further simplification is possible. Thus we will require that keys are atomic (free names or rigid names, `Name` n or `Rigid` n), both initially and after substitution.

Theorem 18 (`subs_not_red_ka`). *Let Γ be reduced, consistent and have atomic keys. Then $(\Gamma \setminus \mathcal{N}^{=})\vec{\theta}$ is reduced.*

Thus, if keys are atomic, the effect of Theorem 18 is to simplify the consistency test thus: to test the consistency of the substituted theory $\Gamma\vec{\theta}$, one reduces Γ to $\Gamma\Downarrow$ and deletes pairs (x, x) to get $\Gamma' = \Gamma\Downarrow \backslash \mathcal{N}^{=}$. One then considers substitution pairs $\vec{\theta}$ of Γ', knowing that any $\Gamma'\vec{\theta}$ is reduced and so the simpler criterion for theory consistency, `thy_cons_red`, applies to it. Thus we get:

Theorem 19. *Let h be a bi-trace, and let $\Gamma = \{h\}$, where Γ is consistent with atomic keys. Let $\vec{\theta}$ be an h-respectful substitution pair, and write $\Gamma\vec{\theta} = \{h\vec{\theta}\}$.*

(a) (`nfs_comm`) $\Gamma\vec{\theta}\Downarrow \backslash \mathcal{N}^{=} = (\Gamma\Downarrow \backslash \mathcal{N}^{=})\vec{\theta} \setminus \mathcal{N}^{=}$
(b) (`nfs_comm_tc`) $\Gamma\vec{\theta}$ *is consistent iff* `thy_cons_red` *holds of* $(\Gamma\Downarrow \backslash \mathcal{N}^{=})\vec{\theta} \setminus \mathcal{N}^{=}$

Unique Completion of a Respectful Substitution A bi-trace can be projected into the fist or second component of each pair, giving lists of marked messages. We can equally project the definition of a respectful substitution pair, so that for a list s of marked messages, substitution θ_i respects s, $s \in$ `sm_resp` θ_i, iff for every free name x in an input message M^i, $\{s'\}\theta_i \vdash x\theta_i$, where $\vdash$ is here the message synthesis relation, and $\{s'\}$ is the set of marked messages prior to M^i. Given h, whose projections are s_1, s_2, if $\vec{\theta}$ respects h then clearly θ_i respects s_i (proved as `bt_sm_resp`, see Appendix A.20). Conversely, given θ_i which respects s_i ($i = 1$ *or* 2), can we complete θ_i to an h-respectful pair $\vec{\theta}$?

Theorem 20 (`subst_exists, subst_unique`). *Given a consistent bi-trace h whose projections to a single message trace are s_1 and s_2, and a substitution θ_1 which respects s_1, there exists θ_2 such that $\vec{\theta} = (\theta_1, \theta_2)$ respects h, and θ_2 is "unique" in the sense that any two such θ_2 act the same on names in $\pi_2(h)$*

We defined a function which, given θ_1 in this situation, returns θ_2. First we defined a function `match_rc1` which, given a theory Γ and a message M, "attempts" to determine a message N such that $\Gamma \vdash M \leftrightarrow N$. By Theorem 20 such a message is unique if Γ is consistent.

The definition of `match_rc1` (Appendix A.24) follows that of `is_der_virt_red` (Appendix A.13), so Theorem 21(a) holds whether or not Γ is actually reduced.

It will be seen that it involves testing for membership of a finite set, and corresponding uses of the ϵ operator, (as in the case of `reduce`, as discussed earlier). Therefore we assert that `match_rc1` is finitely computable.

The return type of `match_rc1` is *message* `option`, which is `Some` *res* if the result *res* is successfully found, or `None` to indicate failure.

Theorem 21. *(a)* `(match_rc1_iff_idvr)` *If* Γ *satisfies* `thy_cons_red`*, then* `is_der_virt_red` (Γ, M, N) *iff* `match_rc1` Γ $M =$ `Some` N
(b) `(match_rc1_indist)` *If* Γ *is consistent, then* $\Gamma \vdash M \leftrightarrow N$ *iff* `match_rc1` $\Gamma\Downarrow$ $M =$ `Some` N

Then we defined a function `second_sub` which uses `match_rc1` to find the appropriate value of $x\theta_2$ for each new x which appears in the bi-trace, and we proved that `second_sub` does in fact compute the θ_2 of Theorem 20. See Appendix A.26 for the definition of `second_sub` and this result. The function `second_sub` tests membership of a finite set, and uses `reduce` and `match_rc1`, so we assert that `second_sub` is also finitely computable.

5 Conclusions and Further Work

We have modelled observer theories and bi-traces in the Isabelle theorem prover, and have confirmed, by proofs in Isabelle, the results of a considerable part of [14]. This work constitutes a significant step formalising open bisimulation for the spi-calculus in Isabelle/HOL, and ultimately towards a logical framework for proving process equivalence.

We discussed the issue of showing finite computability in Isabelle/HOL, using a mixed formal/informal argument, and building upon the discussion in Urban et al [17]. We defined a function `reduce` in Isabelle, and showed that it computes $\Gamma\Downarrow$. Isabelle required us to show that the function terminates. We asserted, with relevant discussion, that inspection shows that the definition does not introduce any infinite aspect into the computation and so asserted that therefore the function is finitely computable. Similarly, we provided a finitely computable function `is_der_virt` and proved that it tests $\Gamma \vdash M \leftrightarrow N$ for a reduced theory Γ.

We then considered bi-traces and bi-trace consistency. The problem here is that, to test bi-trace consistency, it is necessary to test whether $\Gamma\theta$ is consistent for all θ satisfying certain conditions. We proved a number of lemmas which simplify this task, and appear to lead to a finitely computable algorithm for this. In particular, our result on the unique completion of respectful substitutions that relates symbolic trace and bi-trace opens up the possibility to use symbolic trace refinement algorithm [5] to compute a notion of *bi-trace refinement*, which will be useful for bi-trace consistency checking.

Another approach to representating observer theories is to use equational theories, instead of deduction rules, e.g., as in the applied-pi calculus [1]. In this setting, the notion of consistency of a theory is replaced by the notion of *static* *equivalence* between knowledge of observers [1]. Baudet has shown that static equivalence between two symbolic theories is decidable [4], for a class of theories called subterm-convergent theories (which subsumes the Dolev-Yao model of intruder). It will be interesting to work out the precise correspondence between static equivalence and our notion of bi-trace consistency, as such correspondence may transfer proof techniques from one approach to the other.

Acknowledgment We thank the anonymous referees for their comments on an earlier draft. This work is supported by the Australian Research Council through the Discovery Projects funding scheme (project number DP0880549).

References

1. Martín Abadi and Cédric Fournet. Mobile values, new names, and secure communication. In *POPL*, pages 104–115, 2001.
2. Martín Abadi and Andrew D. Gordon. A bisimulation method for cryptographic protocols. *Nord. J. Comput.*, 5(4):267–303, 1998.
3. Martin Abadi and Andrew D. Gordon. A calculus for cryptographic protocols: The spi calculus. *Information and Computation*, 148(1):1–70, 99.
4. Mathieu Baudet. *Sécurité des protocoles cryptographiques: aspects logiques et calculatoires*. PhD thesis, École Normale Supérieure de Cachan, France, 2007.
5. Michele Boreale. Symbolic trace analysis of cryptographic protocols. In *Proceedings of ICALP 2001*, volume 2076 of *LNCS*, pages 667 – 681. Springer-Verlag, 2001.
6. Michele Boreale, Rocco De Nicola, and Rosario Pugliese. Proof techniques for cryptographic processes. *SIAM J. Comput.*, 31(3):947–986, 2001.
7. Johannes Borgström, Sébastien Briais, and Uwe Nestmann. Symbolic bisimulation in the spi calculus. In *CONCUR*, vol. 3170 of *LNCS*, pp 161–176. Springer, 2004.
8. Johannes Borgström and Uwe Nestmann. On bisimulations for the spi calculus. *Mathematical Structures in Computer Science*, 15(3):487–552, 2005.
9. J. E. Dawson and R. Goré. Formalising cut-admissibility for provability logic. Submitted, 2009.
10. D. Dolev and A. Yao. On the security of public-key protocols. *IEEE Transactions on Information Theory*, 2(29), 1983.
11. Temesghen Kahsai and Marino Miculan. Implementing spi calculus using nominal techniques. In *CiE*, volume 5028 of *LNCS*, pages 294–305. Springer, 2008.
12. Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, Part II. *Information and Computation*, pages 41–77, 1992.
13. Davide Sangiorgi. A theory of bisimulation for the pi-calculus. *Acta Inf.*, 33(1):69–97, 1996.
14. Alwen Tiu. A trace based bisimulation for the spi calculus: An extended abstract. In *Proc. APLAS*, vol. 4807 of *LNCS*, pages 367–382. Springer, 2007.
15. Alwen Tiu. A trace based bisimulation for the spi calculus. Preprint, available from `http://arxiv.org/pdf/0901.2166v1`, 2009.
16. Alwen Tiu and Rajeev Goré. A proof theoretic analysis of intruder theories. In *Proceedings of RTA 2009*, 2009. To appear.
17. Christian Urban, James Cheney, and Stefan Berghofer. Mechanizing the metatheory of LF. In *LICS*, pages 45–56. IEEE Computer Society, 2008.

A Isabelle definitions and theorems

A.1 Definitions of message equivalence deducibility

Lemma 22. *(a) [15, Lemma 7]* (`indist_wk, indist_alt_wk`) *Weakening (that is, augmenting Γ) is admissible (in both systems)*
(b) (`indist_alt`) *the same sequents $\Gamma \vdash M \leftrightarrow N$ are deducible in both systems*

In Isabelle we defined, inductively, sets of sequents `indist` and `indist_alt`, which are the sets of sequents derivable in the two systems. Subsequently we found it helpful to define the corresponding sets of rules explicitly, calling them `indpsc` and `indpsc_alt`. The distinction between these approaches is illustrated by the two expressions of the rule (*pr*) in the text below. Note that the set `indpsc` is the infinite set of all instantiations of the six rules shown in Figure 2.

```
"[| (?oth, ?Ma, ?Na) : indist; (?oth, ?Mb, ?Nb) : indist |] ==>
  (?oth, (Mpair ?Ma ?Mb, Mpair ?Na ?Nb)) : indist"
"([(?oth, ?Ma, ?Na), (?oth, ?Mb, ?Nb)],
  (?oth, (Mpair ?Ma ?Mb, Mpair ?Na ?Nb))) : indpsc"
```

This provides an example of shallow or deep embedding of inference rules, discussed in [9], and illustrates a point made in the discussion there, that formalising the rules as explicit objects enables us to express notions such as `indpsc` $\subseteq$ `indpsc_alt`. As discussed in [9], `derrec` *rls* {} means the set of things (here, sequents $\Gamma \vdash M \leftrightarrow N$) derivable using the set of inference rules *rls*.

```
indist_wk : "[| (?oth, ?MN) : indist; ?oth <= ?othl |] ==>
  (?othl, ?MN) : indist"
indist_alt : "indist_alt == indist"
indpsc_le_alt : "indpsc <= indpsc_alt"
indist_derrec : "indist == derrec indpsc {}"
```

A.2 Invertibility and cut-admissibility

Lemma 6 of [15] gives invertibility of the right rules (*pr*) and (*er*), and Lemma 5 gives invertibility-like results for the left rules (*pl*) and (*el*). We proved these as `pr_inv`, `er_inv`, `pl_inv`, `el_inv`. In [15, Lemma 10] a "cut-admissibility" result

$$\text{If } \Gamma \vdash M \leftrightarrow N \text{ and } \Delta, (M, N) \vdash R \leftrightarrow T \text{ then } \Gamma \cup \Delta \vdash R \leftrightarrow T$$

is proved from Lemma 5, by induction on the derivation of $\Gamma \vdash M \leftrightarrow N$. In Isabelle we showed that it can also be proved from Lemma 6, by induction on the derivation of $\Delta, (M, N) \vdash R \leftrightarrow T$. In fact we found it easier to prove cut-admissibility directly for the modified system of Figure 2. We actually proved a stronger result involving several cut-formulae:

Theorem 23 (`indist_alt_ca`)**.** *If $\Gamma \vdash M_i \leftrightarrow N_i$ for all $i \in I$, and $\Delta' \vdash R \leftrightarrow T$, where $\Delta' \subseteq \Delta \cup \bigcup_{i \in I}(M_i, N_i)$, then $\Gamma \cup \Delta \vdash R \leftrightarrow T$*

```
indist_alt_ca : "[| (?Del', ?RT) : indist_alt;
  ?Del' <= ?Del Un ?MNs; ALL MNi:?MNs. (?Gam, MNi) : indist_alt |]
                 ==> (?Del Un ?Gam, ?RT) : indist_alt"
```

Note how we also used a stronger inductive hypothesis, whose assumption is an inequality ($\Delta' \subseteq \ldots$); this was much easier than if the "$\subseteq$" were "$=$" instead.

We noted that the cut-admissibility result easily implies Lemmas 5 and 6 (so proving both before proving cut-admissibility was redundant). We also observed that these cut-admissibility proofs, unusually, did not require the inductive assumption that cut is admissible for smaller objects, which is because INSERT REASON.

In fact if we remove (M_k, N_k) from the second premise of (el) in Figure 2, we get another set of rules which is apparently weaker, for which we called the derivable sequents `indist_nk_alt`. There is a corresponding cut-admissibility result for `indist_nk_alt` (called `indist_nk_alt_ca`), which can be used to show that the systems `indist_k_alt` and `indist_alt` are equivalent.

A.3 Isabelle theorems for `smpsc` and Lemma 1

In mathematical notation we can just write write π_1 or π_2 to denote the projection to the first or second component, for pairs, theories, sequents and rules, eg, `smpsc` $= \pi_1($`indpsc`$)$. But the definition in Isabelle, `smpsc_def`, uses `seqmap fst`, which applies `fst` (ie, π_1) to each member of the antecedent and to the consequent of a sequent, and `pscmap` f, which applies f to each premise and to the conclusion of a rule.

```
pair_def : "pair == %x. (x, x)"
smpsc_def : "smpsc == pscmap (seqmap fst) ` indpsc"
smpsc_alt : "smseq : smpsc == pscmap (seqmap pair) smseq : indpsc"

slice_derrec_smpsc_empty :
  "?c : derrec indpsc {} ==> seqmap fst ?c : derrec smpsc {}"
derrec_smpsc_eq :
  "(?c : derrec smpsc {}) = (seqmap pair ?c : derrec indpsc {})"
smpsc_ex_indpsc_der : "[| (fst ` ?oth, ?M) : derrec smpsc {};
    thy_cons ?oth |] ==> EX N. (?oth, ?M, N) : derrec indpsc {}"
```

A.4 Decidability of $\vdash$

The simplest approach to proving decidability of $\vdash$ uses the fact that there only finitely many sequents which will appear in a proof, or in an attempted backwards proof, of a given sequent. (The proof given in §3.1 is more complex but leads to a more computationally efficient approach for determining theory consistency).

We define the set of pairs which might appear in a sequent in a backwards proof of a given sequent $\Gamma \vdash M \leftrightarrow N$: for a pair (M, N) of messages, pairs_of_msgs (M, N) is the set of pairs of messages obtained by "decomposing" (M, N), including (M, N) and partially decomposed pairs, and pairs_of_seq $(\Gamma, (M, N))$ returns all pairs in Γ or (M, N). Then rules_in_pairs S is the set of rules such that all the pairs of any of the sequents in the rule are in S.

```
inductive "pairs_of_msgs MN"
  intros
    id : "MN : pairs_of_msgs MN"
    pa : "(Mpair Ma Mb, Mpair Na Nb) : pairs_of_msgs MN ==>
      (Ma, Na) : pairs_of_msgs MN"
    pb : "(Mpair Ma Mb, Mpair Na Nb) : pairs_of_msgs MN ==>
      (Mb, Nb) : pairs_of_msgs MN"
    ep : "(Enc Mp Mk, Enc Np Nk) : pairs_of_msgs MN ==>
      (Mp, Np) : pairs_of_msgs MN"
    ek : "(Enc Mp Mk, Enc Np Nk) : pairs_of_msgs MN ==>
      (Mk, Nk) : pairs_of_msgs MN"

primrec
  "pairs_of_seq (G, MN) = (UN p: insert MN G. pairs_of_msgs p)"

inductive "rules_in_pairs Ps"
  intros
    I : "ALL s : insert c (set ps). pairs_of_seq s <= Ps ==>
      (ps, c) : rules_in_pairs Ps"
```

We find that a sequent $\Gamma \vdash M \leftrightarrow N$ has finitely many pairs.

```
finite_pairs_of_seq : "finite ?G ==> finite (pairs_of_seq (?G, ?MN))"
```

Given a finite set S of pairs, there are only finitely many sequents whose pair sets are contained in S, and the same holds for rules provided that we bound the number of premise sequents. Consequently there are only finitely many instantiations of the rules in Figure 2 whose pairs are in a given set S. (Recall that inspsc is the infinite set of all instantiations of the six rules shown in Figure 2).

```
finite_seqs_pairs_in : "finite ?S ==> finite {seq. pairs_of_seq seq <= ?S}"
finite_rules : "finite ?S ==> finite {(ps, c). length ps <= 2 &
   insert c (set ps) <= {seq. pairs_of_seq seq <= ?S}}"
finite_indpsc : "finite ?S ==> finite (indpsc Int rules_in_pairs ?S)"
```

Now we find that the pairs of premises of rules in inspsc are contained in the pairs of the conclusion, and consequently, whenever any sequent $\Gamma \vdash M \leftrightarrow N$ is derivable using rules in inspsc, then it is derivable using only those rules whose pairs are in the set of pairs of $\Gamma \vdash M \leftrightarrow N$.

```
indpsc_pairs : "[| (?ps, ?c) : indpsc; ?p : set ?ps |] ==>
```

```
   pairs_of_seq ?p <= pairs_of_seq ?c"
derrec_indpsc_pairs : "?seq : derrec indpsc {} ==>
   ?seq : derrec (indpsc Int rules_in_pairs (pairs_of_seq ?seq)) {}"
```

Combining all these we find that given a sequent $\Gamma \vdash M \leftrightarrow N$ that we wish to derive, we know that the only rules used in the derivation will be those whose pairs are in `pairs_of_seq` $(\Gamma, (M, N))$, and there are only finitely many of these. Further, since no derivation need contain the same sequent twice, no rule (that is, instantiated rule) need be used twice, so the process of searching for a derivation of a given end-sequent $\Gamma \vdash M \leftrightarrow N$ will necessarily be finite.

A.5 Isabelle theorems for Lemma 2

```
red_nc : "(?oia, ?oib) : oth_red ==>
  ((?oia, ?x) : indist) = ((?oib, ?x) : indist)"
nf_nc : "finite ?oia ==>
  ((nf oth_red ?oia, ?MN) : indist) = ((?oia, ?MN) : indist)"
```

A.6 Proof of confluence of $\longrightarrow$

Confluence of $\longrightarrow$ is reasonably easy to see since the side condition for the second rule is of the form $\Gamma' \vdash M_k \leftrightarrow N_k$ where Γ' is exactly the theory being reduced, and, from Lemma 2, this condition (for a particular M_k, N_k) will continue to hold, or not, when other reductions have changed Γ'.

But it was not easy to prove confluence the natural way in Isabelle, since there are so many cases. Two reductions, each of which may use either rule, gives four cases. Then either of these reductions may introduce the same pair which is removed by the other reduction, so it then needs to be removed again. This gives four sub-cases for each of the main cases, and each case required at some point an instantiation specific to that case.

We therefore devised an alternative proof. This proof used the fact that reduction is well-founded although that is not necessary to prove confluence.

Firstly we simplified the main result by changing the side condition for the second rule, by defining $\longrightarrow_K$ for a set K, which remains fixed throughout the reduction process. This K will eventually be equated to the set $\{(M_k, N_k) \mid \Gamma \vdash M_k \leftrightarrow N_k$ (which, as noted above, remains unchanged as Γ gets reduced). Thus, in defining $\longrightarrow_K$, the side condition for the second rule is simply $(M_k, N_k) \in K$.

The key lemmas here follow. Lemma 24(a) encapsulates the multitude of cases referred to above. Lemma 24(b) is proved by well-founded induction, based on the fact that reduction is well-founded.

Lemma 24. *For a set K,*

(a) (`oth_red_k_lem`) *if $\Gamma \longrightarrow_K \Gamma'$, $p \in \Gamma \setminus \Gamma'$ and $p \in \Delta$, then there exists Δ' such that $\Delta \longrightarrow_K \Delta'$ and $\Delta \setminus \{p\} \subseteq \Delta' \subseteq \Delta \cup \Gamma'$*

(b) (`oth_red_k_min`) *Suppose $\Gamma \longrightarrow_K \Gamma'$ and $\Delta \longrightarrow_K \Delta'$. Let $\Theta' = \Gamma' \cup \Delta' \setminus (\Gamma \setminus \Gamma') \setminus (\Delta \setminus \Delta')$. If $\Theta' \subseteq \Theta \subseteq \Gamma \cup \Delta \cup \Gamma' \cup \Delta'$ then $\Theta \longrightarrow_K^* \Theta'$.*

```
oth_red_k_lem :
 "[| (?G', ?G) : oth_red_k ?K; ?b : ?D; ?b : ?G - ?G' |] ==>
   EX D'.  (D', ?D) : oth_red_k ?K & ?D - {?b} <= D' & D' <= ?D Un ?G'"

oth_red_k_min : "[| ?T : wfp (oth_red_k ?K);
  (?G', ?G) : oth_red_k ?K; (?D', ?D) : oth_red_k ?K;
     ?T' = ?G' Un ?D' - (?G - ?G') - (?D - ?D');
     ?T' <= ?T; ?T <= ?G Un ?D Un ?G' Un ?D' |]
  ==> (?T', ?T) : (oth_red_k ?K)^*"
```

From here we successively proved the following results

Lemma 25. *(a)* (oth_red_k_confl) *If, for a set K, $\Gamma \longrightarrow_K \Delta'$ and $\Gamma \longrightarrow_K \Delta''$ then there exists Θ such that $\Delta' \longrightarrow^*_K \Theta$ and $\Delta'' \longrightarrow^*_K \Theta$*

(b) (oth_red_confl) *If $\Gamma \longrightarrow \Delta'$ and $\Gamma \longrightarrow \Delta''$ then there exists Θ such that $\Delta' \longrightarrow^* \Theta$ and $\Delta'' \longrightarrow^* \Theta$*

(c) (oth_red_confl_rtc) *If $\Gamma \longrightarrow^* \Delta'$ and $\Gamma \longrightarrow^* \Delta''$ then there exists Θ such that $\Delta' \longrightarrow^* \Theta$ and $\Delta'' \longrightarrow^* \Theta$*

```
oth_red_k_confl :
   "[| finite ?G; (?D, ?G) : oth_red_k ?K; (?D', ?G) : oth_red_k ?K |]
      ==> EX T. (T, ?D) : (oth_red_k ?K)^* & (T, ?D') : (oth_red_k ?K)^*"
oth_red_confl :
   "[| finite ?G; (?D, ?G) : oth_red; (?D', ?G) : oth_red |]
      ==> EX T. (T, ?D) : oth_red^* & (T, ?D') : oth_red^*"
oth_red_confl_rtc :
   "[| finite ?G; (?D, ?G) : oth_red^*; (?D', ?G) : oth_red^* |]
      ==> EX T. (T, ?D) : oth_red^* & (T, ?D') : oth_red^*"
```

We define nf_props $r\ x\ y$ to mean that y has the properties of a r-normal form of x, and show easily (in nf_unique) from this definition that such a y is unique. We define nf $r\ x = \epsilon y.$ nf_props $r\ x\ y$, which means some y satisfying nf_props $r\ x$, if such y exists; otherwise some y of the right type. Then, if such a y exists, then the definition nf $r\ x$ refers to it.

The definition nf_def is nf $r\ x = \epsilon y.$ nf_props $r\ x\ y$, which means some y satisfying nf_props $r\ x$, if such y exists; otherwise some y of the right type.

```
tri_prop_def : "tri_prop ?r ?x ?y == ALL z. (z, ?x) : ?r^* --> (?y, z) : ?r^*"
rsmin_def : "rsmin ?r ?S == {x : ?S. ALL y. (y, x) : ?r --> y ~: ?S}"
nf_props_def : "nf_props ?r ?x ?y ==
  (?y, ?x) : ?r^* & ?y : rsmin ?r UNIV & tri_prop ?r ?x ?y"
nf_unique : "nf_props ?r ?x ?nx ==> ?nx = nf ?r ?x"
nf_def : "nf ?r ?x == SOME y. nf_props ?r ?x y"
```

It is a standard result, and easy in Isabelle, that confluence and termination give normal forms satisfying these properties.

Theorem 26 **(nf_oth_red).** *Any theory Γ has a $\longrightarrow$-normal form $\Gamma\Downarrow$*

```
nf_oth_red : "finite ?x ==> nf_props oth_red ?x (nf oth_red ?x)"
```

A.7 Isabelle theorems for Theorem 4

```
red_alt_lem : "(?othe, ?M, ?N) : indist ==>
  (?othe - {(Enc ?Qp ?M, Enc ?Rp ?Rk)}, ?M, ?N) : indist |
             (EX oth'. (oth', ?othe) : oth_red_alt)"

oth_red_alt_lem : "(?D, ?G) : oth_red ==> EX D'. (D', ?G) : oth_red_alt"
rsmin_or_alt : "rsmin oth_red_alt UNIV = rsmin oth_red UNIV"
nf_acc_alt : "finite ?G ==> (nf oth_red ?G, ?G) : oth_red_alt^*"
nf_alt : "[| finite ?G; nf_props oth_red ?G ?D |] ==>
  nf_props oth_red_alt ?G ?D"
nf_same : "finite ?G ==> nf oth_red_alt ?G = nf oth_red ?G"
```

As noted in the text we may further change $\longrightarrow'$ by deleting the (M_k, N_k) from the second rule: call the resulting relation $\longrightarrow''$, or, in Isabelle, `oth_red_nk`.

Use of cut-admissibility shows that Lemma 2(a) applies to this relation as it does to $\longrightarrow$, ie

```
red_nk_nc : "(?G', ?G) : oth_red_nk ==>
    ((?G', ?x) : indist) = ((?G, ?x) : indist)"
```

Clearly the same theories are irreducible for $\longrightarrow'$ as for $\longrightarrow''$. We did not establish confluence for this relation but noted that if $\Gamma \longrightarrow''^{*} \Delta$ and Δ is $\longrightarrow''$-irreducible, then, by Lemma 7(c), $\Delta \setminus \mathcal{N}^{=} = \Gamma\Downarrow \setminus\mathcal{N}^{=}$.

```
red_nk_same_ex_Ne : "[| (?D, ?G) : oth_red_nk^*;
      ?D : rsmin oth_red_nk UNIV; finite ?G |] ==>
    ?D - Ne = reduce ?G - Ne"
```

A.8 Isabelle theorems for Lemma 5

```
nf_no_left : "[| ?seq : derrec indpsc {}; fst ?seq : rsmin oth_red UNIV |]
          ==> ?seq : derrec indpsc_virt {}"

virt_dec : "(?seq : derrec indpsc_virt {}) = is_der_virt ?seq"
```

A.9 Definitions used in `reduce`

```
consts
  red_pair :: "msg * msg => obs_thy => obs_thy"
  red_enc :: "msg * msg => obs_thy => obs_thy"
  reduce :: "obs_thy => obs_thy"

recdef "red_pair" "{}"
  "red_pair (Mpair Ma Mb, Mpair Na Nb) =
    (%oth. insert (Ma, Na) (insert (Mb, Nb) oth))"
  "red_pair p = (%oth. oth)"
```

```
recdef "red_enc" "{}"
  "red_enc (Enc Mp Mk, Enc Np Nk) =
    (%oth. insert (Mp, Np) (insert (Mk, Nk) oth))"
  "red_enc p = (%oth. oth)"
```

A.10 Isabelle theorems for Theorem 6

```
reduce_nf : "finite ?x ==> nf oth_red ?x = reduce ?x"
reduce_nf_alt : "finite ?x ==> nf oth_red_alt ?x = reduce ?x"

virt_reduce : "finite ?oth ==>
  is_der_virt (reduce ?oth, ?MN) = ((?oth, ?MN) : indist)"

idvr_reduce : "finite ?S ==>
  ((?S, ?M, ?N) : indist) = is_der_virt_red (reduce ?S, ?M, ?N)

reduce_nc : "((reduce ?oib, ?MN) : indist) = ((?oib, ?MN) : indist)"
```

A.11 Definitions and theorems for single message sets

```
reduce_ss_def : "reduce_ss S == fst ‘ reduce (pair ‘ S)"
idv_ss_def : "idv_ss seq == is_der_virt (seqmap pair seq)"
reduce_paired : "G <= range pair ==> reduce G <= range pair"
idv_reduce_ss : "finite S ==>
  ((S, M) : derrec smpsc {}) = idv_ss (reduce_ss S, M)"
```

A.12 Isabelle theorem for Theorem 7

```
nf_equiv_der : "[| finite ?otha; finite ?othb |] ==>
  (ALL MN. ((?otha, MN) : indist) = ((?othb, MN) : indist)) =
     (nf oth_red ?otha - Ne = nf oth_red ?othb - Ne)" :
```

A.13 Definition of `is_der_virt_red`

```
recdef "is_der_virt_red" "measure (%(oth, M, N). size M)"
  "is_der_virt_red (oth, Name i, Name j) = ((Name i, Name j) : oth | i = j)"
  "is_der_virt_red (oth, Mpair Ma Mb, Mpair Na Nb) =
    (is_der_virt_red (oth, Ma, Na) & is_der_virt_red (oth, Mb, Nb))"
  "is_der_virt_red (oth, Enc Mp Mk, Enc Np Nk) =
    (if is_der_virt_red (oth, Mk, Nk) then is_der_virt_red (oth, Mp, Np)
      else (Enc Mp Mk, Enc Np Nk) : oth)"
  "is_der_virt_red (oth, M, N) = ((M, N) : oth)"
```

The following is the theorem which says that if Γ is reduced, then `is_der_virt_red` $(\Gamma, (M, N))$ iff `is_der_virt` $(\Gamma, (M, N))$.

```
is_der_virt_red : "?oth : rsmin oth_red UNIV ==>
  is_der_virt (?oth, ?M, ?N) = is_der_virt_red (?oth, ?M, ?N)"
```

A.14 Isabelle theorems for Lemma 10

```
thy_cons_equiv :
   "thy_cons ?oth =
      (ALL M N.
          (?oth, M, N) : indist -->
          ((M : Mpairs) = (N : Mpairs) &
           (M : Encs) = (N : Encs) & (M : Rigids) = (N : Rigids)) &
          (ALL Mp Mk Np Nk.
              M = Enc Mp Mk & N = Enc Np Nk -->
              (fst ` ?oth, Mk) : derrec smpsc {} |
              (snd ` ?oth, Nk) : derrec smpsc {} -->
              (?oth, Mk, Nk) : indist) &
          (ALL R.
              ((?oth, M, R) : indist --> R = N) &
              ((?oth, R, N) : indist --> R = M)))"

thy_cons_equivd : "thy_cons ?oth = thy_cons (?oth - Ne)"

nf_cons : "finite ?oth ==> thy_cons (nf oth_red ?oth) = thy_cons ?oth"

cons_der_same : "[| finite ?oth; finite ?oth';
    ALL MN. ((?oth, MN) : indist) = ((?oth', MN) : indist) |]
        ==> thy_cons ?oth = thy_cons ?oth'"
```

A.15 Isabelle theorems for Theorem 13

```
tc_red_iff : "finite ?S ==> thy_cons ?S = thy_cons_red (reduce ?S)"
```

A.16 Definitions of a consistent bi-trace

Definition 15 can be modelled in Isabelle using its `recdef` feature, where the definition is recursive, depending on the consistency of smaller bi-traces. In Isabelle we define `bt_cons_rec` in this way. Here, the `size` of a bi-trace is simply its length as a list.

```
recdef "bt_cons_rec" "measure size"
  "bt_cons_rec [] = True"
  "bt_cons_rec (In MN # bt) = (bt_cons_rec bt & (oth_of bt, MN) : indist)"
  "bt_cons_rec (Out MN # bt) = (bt_cons_rec bt &
     (ALL subs. bt : bt_resp subs --> bt_cons_rec (subst_bt subs bt) -->
         thy_cons (oth_of (subst_bt subs (Out MN # bt)))))"
```

We also defined `bt_cons`, an inductively defined set, where $h \in$ `bt_cons` iff h satisfies the variant of Definition 15 with the underlined words omitted.

An inductively defined set in Isabelle is the unique largest set which satisfies the given clauses such as (a) to (c) of Definition 15. This is the least fixpoint of a certain function. Here, it would include all bi-traces which can be shown to be consistent by repeated use of the given clauses.

```
inductive "bt_cons"
  intros
    NilI : "[] : bt_cons"
    InI : "bt : bt_cons ==> (oth_of bt, MN) : indist ==> (In MN # bt) : bt_cons"
    OutI : "bt : bt_cons ==> ALL subs. bt : bt_resp subs -->
        thy_cons (oth_of (subst_bt subs (Out MN # bt))) ==>
      (Out MN # bt) : bt_cons"
```

But with the underlined words included, the we could not define bi-trace consistency as an inductively defined set. This is because to show that one bi-trace is consistent may lead to another bi-trace not being consistent; the resulting inductively defined set, if allowed, would be the least fixpoint (however defined) of some non-monotonic function, and so would not have the required properties.

Proposition 27 (`bt_cons_equiv`)**.** *Omitting the underlined words from Definition 15 defines the same notion of bi-trace consistency*

```
bt_cons_equiv : "validbt ?bt ==> bt_cons_rec ?bt = (?bt : bt_cons)"
```

A.17 Isabelle theorems for Lemma 16

Our Isabelle proofs of Lemma 16(b) and (c), that is, of `bt_resp_comp` and `cons_subs_bt`, actually used Definition 15 with the underlined words omitted.

```
subst_indist : "[| (?oth, ?MN) : indist;
  ALL n:fn_msgs ?MN Un fn_oth ?oth.
    (subst_oth ?subs ?oth, subst_nat ?subs n) : indist |] ==>
  (subst_oth ?subs ?oth, subst_pr ?subs ?MN) : indist"

bt_resp_comp : "[| ?bt : bt_resp ?subs; validbt ?bt;
  subst_bt ?subs ?bt : bt_resp ?subsa; ?bt : bt_cons |] ==>
  ?bt : bt_resp (comp_psub ?subsa ?subs)"

cons_subs_bt : "[| ?bt : bt_cons; validbt ?bt; ?bt : bt_resp ?subs |]
  ==> subst_bt ?subs ?bt : bt_cons"
```

Arguably, as a consequence of Lemma 16(c) the words underlined in Definition 15 should be redundant. To be precise, $h \in$ `bt_cons` iff h satisfies the clauses of Definition 15. Then, to show that $h \in$ `bt_cons` $\Leftrightarrow$ `bt_cons_rec` h requires also that the clauses of Definition 15 *uniquely* define a function. This consideration justifies the effort of making a separate definition, `bt_cons_rec`, in Isabelle, and then proving the equivalence, Proposition 27, that is, `bt_cons_equiv`.

A.18 Definition of fn_bt, the free names of a bi-trace

```
primrec
  "rem_io (In mp) = mp"
  "rem_io (Out mp) = mp"

primrec
  "fn_msgs (M, N) = fn_msg M Un fn_msg N"

primrec
  Nil : "fn_bt [] = []"
  Cons : "fn_bt (mmp # mmps) = fn_msgs (rem_io mmp) # fn_bt mmps"
```

A.19 Definition and theorems for thy_strl_resp and Theorem 29

Definition 28. *For a theory Γ and a list ns of sets of free names, and a substitution pair θ,* thy_strl_resp *θ ns Γ holds if, for each $n \in ns$ and $x \in n$, $\Gamma' \vdash x\theta_1 \leftrightarrow x\theta_2$, where Γ' is obtained by deleting from Γ all message pairs containing free names which are not in some n' that appears after n in ns*

```
"thy_strl_resp ?f [] ?oth = True"
"thy_strl_resp ?f (?n # ?ns) ?oth = (thy_strl_resp ?f ?ns ?oth &
  (ALL x:?n. (subst_oth ?f (?oth - oth_ctg_fns (- Union (set ?ns))),
    subst_nat ?f x) : indist))"
```

Theorem 29. *(a)* (thy_strl_resp_reduce) *If* thy_strl_resp *$\vec{\theta}$ ns Γ then* thy_strl_resp *$\vec{\theta}$ ns $\Gamma\Downarrow$*
(b) (bt_thy_strl_resp) *if $\vec{\theta}$ is h-respectful then* thy_strl_resp *$\vec{\theta}$ (*fn_bt *h) $\{h\}$*

```
thy_strl_resp_reduce :
  "thy_strl_resp ?f ?ns ?oth ==> thy_strl_resp ?f ?ns (reduce ?oth)"

bt_thy_strl_resp : "[| validbt ?bt; ?bt : bt_resp ?f |] ==>
  thy_strl_resp ?f (fn_bt ?bt) (oth_of ?bt)"
```

A.20 Isabelle theorem bt_sm_resp

```
bt_sm_resp : "[| ?f = (?fl, ?fr); ?bt : bt_resp ?f |] ==>
  map (map_io fst) ?bt : sm_resp ?fl"
```

A.21 Isabelle theorems for Theorem 17

```
nf_subst_nf_Ne : "[| validbt ?bt; ?bt : bt_resp ?f |] ==>
  nf oth_red (subst_oth ?f (oth_of ?bt)) - Ne =
  nf oth_red (subst_oth ?f (nf oth_red (oth_of ?bt) - Ne)) - Ne"
```

A.22 Isabelle theorems for Theorem 18

```
subs_not_red_ka : "[| ka_oth ?oth; finite ?oth; thy_cons ?oth;
    ?oth : rsmin oth_red UNIV |] ==>
  subst_oth ?f (?oth - Ne) : rsmin oth_red UNIV"
```

A.23 Isabelle theorems for Theorem 20

```
subst_unique : "[| ?bt : bt_resp ?f; ?bt : bt_resp ?g;
    ?bt : bt_cons; validbt ?bt |] ==>
    (map (map_io (fst o subst_pr ?f)) ?bt =
      map (map_io (fst o subst_pr ?g)) ?bt) =
    (map (map_io (snd o subst_pr ?f)) ?bt =
       map (map_io (snd o subst_pr ?g)) ?bt)"

subst_exists :
   "[| map (map_io fst) ?bt : sm_resp ?fl; ?bt : bt_cons; validbt ?bt |]
         ==> EX fr. ?bt : bt_resp (?fl, fr)"
```

A.24 Definition of `match_rc1`

```
primrec
  Names : "match_rc1 oth (Name i) = Some (Name i)"
  Rigid : "match_rc1 oth (Rigid i) =
    (if (EX msg. (Rigid i, msg) : oth) then
      Some (SOME msg. (Rigid i, msg) : oth) else None)"
  Mpair : "match_rc1 oth (Mpair Ma Mb) =
    (if match_rc1 oth Ma = None | match_rc1 oth Mb = None then None
    else Some (Mpair (the (match_rc1 oth Ma)) (the (match_rc1 oth Mb))))"
  Enc : "match_rc1 oth (Enc Mp Mk) =
    (if match_rc1 oth Mk = None then
      if (EX msg. (Enc Mp Mk, msg) : oth) then
        Some (SOME msg. (Enc Mp Mk, msg) : oth) else None
    else if match_rc1 oth Mp = None then None
    else Some (Enc (the (match_rc1 oth Mp)) (the (match_rc1 oth Mk))))"
```

A.25 Isabelle theorems for Theorem 21

```
match_rc1_iff_idvr : "thy_cons_red ?oth ==>
  is_der_virt_red (?oth, ?M, ?N) = (match_rc1 ?oth ?M = Some ?N)"

match_rc1_indist : "[| finite ?S; thy_cons ?S |] ==>
  ((?S, ?M, ?N) : indist) = (match_rc1 (reduce ?S) ?M = Some ?N)"
```

A.26 Definition of second_sub

```
primrec
  Nil : "second_sub [] f n = Name n"
  Cons : "second_sub (mmp # bt) f n =
    (if n : fn_oth (oth_of bt) then second_sub bt f n
    else if n ~: fn_msgs (rem_io mmp) then Name n
    else the (match_rc1 (reduce (subst_oth (f, second_sub bt f) (oth_of bt)))
      (f n)))"
```

The following theorem shows that `second_sub` does in fact compute the θ_2 of Theorem 20.

```
second_sub : "[| map (map_io fst) ?bt : sm_resp ?fl;
      ?bt : bt_cons; validbt ?bt |] ==>
    ?bt : bt_resp (?fl, second_sub ?bt ?fl)"),
```