A Fully Abstract Encoding of the π-Calculus with Data Terms

Michael Baldamus, Joachim Parrow, Björn Victor

Department of Information Technology
Uppsala University
Box 337, SE-751 05 Uppsala, Sweden

Technical report 2005-004
February 2005, updated July 2005
ISSN 1404-3203

A Fully Abstract Encoding of the π-Calculus with Data Terms*

Michael Baldamus[1†] Joachim Parrow[2] Björn Victor[2]

[1] The Linnaeus Centre for Bioinformatics /
[2] Department of Information Technology, Uppsala University, Sweden

12 July 2005

Abstract

The π-calculus with data terms (πT) extends the pure π-calculus by data constructors and destructors and allows data to be transmitted between agents. It has long been known how to encode such data types in π, but until now it has been open how to make the encoding *fully abstract*, meaning that two encodings (in π) are semantically equivalent precisely when the original πT agents are semantically equivalent. We present a new type of encoding and prove it to be fully abstract with respect to may-testing equivalence. To our knowledge this is the first result of its kind, for any calculus enriched with data terms. It has particular importance when representing security properties since attackers can be regarded as may-test observers. Full abstraction proves that it does not matter whether such observers are formulated in π or πT, both are equally expressive in this respect. The technical new idea consists of achieving full abstraction by encoding data as table entries rather than active processes, and using a firewalled central integrity manager to ensure data security.

1 Introduction

The increasingly complicated mechanisms to guarantee secure communications have spurred the development of appropriate formal description techniques. In this paper we study a prototypical such formalism, a π-calculus enriched with data terms (πT), and show how it can be encoded in the more fundamental π-calculus [MPW92] while preserving full abstraction. This means that two processes in πT are equivalent precisely when their encodings are equivalent. Although encodings between such calculi has proved a rich field of study this is the first result of its kind. We achieve it by designing the encoding in a different way from what is usually done. The full proof is very technical and we here outline the main ideas, which are quite general and can in principle be applied to many similar calculi.

πT extends the basic π-calculus by including constructors and deconstructors for data terms. In this the calculus resembles a high level parallel programming language, and specifications of security protocols can be made in a familiar operational style. It can be seen as a generalisation of the spi-calculus [AG99] (which extends π with primitives for encryption and

*Work supported by European Union project PROFUNDIS, Contract No. IST-2001-33100.

†To whom correspondence should be addressed (*http://www.lcb.uu.se/∼michaelb*).

decryption) in that arbitrary constructors and destructors are allowed. It can also be seen as a simplification of the applied π-calculus by Abadi and Fournet [AF01], since it does not admit equations over the data terms.

A key idea when encoding data was expressed already in the first papers on the π-calculus: a data structure is represented in π as a process with the ability to interact along designated ports in order to carry out operations. In such a manner all known kinds of data structures, certainly including those of πT, can be encoded.

The problem with this kind of encoding appears when we consider a useful notion of semantic equality between πT processes. We shall here as a prime example look at *testing* equivalence. The main idea is that two processes are behaviourally equivalent precisely when they can satisfy the same *tests*. Tests are arbitrary processes of the calculus, which of course is expressive enough to describe not only security protocols but also the potential attackers on them. So if we know of two processes that they are testing equivalent we know that they will behave similarly in all conceivable environments. In an earlier paper [BPV04] we have proved for the spi calculus that a process satisfies a test if and only if the encoding of the process satisfies the encoding of the test.

This traditional kind of encoding entails only that two processes satisfy the same tests *only if the tests on the π-calculus-side are encodings of πT calculus tests*. But in the π-calculus there are also tests which are not encodings of any πT test, and some of these may be able to discriminate between the encodings of otherwise equivalent processes. Formally *full abstraction*, the result that two πT processes are equivalent precisely when their encodings are equivalent, has proved elusive, not only for πT but for all similar calculi, and for similar operationally defined equivalences.

In this paper we exhibit an encoding from πT to π and prove it to be fully abstract. A key ingredient is that we use a central so called *integrity manager* ($\mathbf{M}$) which stores all data values generated throughout a computation. All access to data must go through a level of indirection at $\mathbf{M}$, which only allows accesses that adheres to the protocols for interacting with data. Thus, the previously dangerous π-calculus processes that are not encodings of πT processes are rendered impotent.

The remaining sections of this paper are organised as follows:

Section 2 introduces πT. In Section 2 There is also a subsection on how to instantiate our πT-calculus so as to recover the spi calculus.

Section 3 presents the encoding of πT into the polyadic π-calculus.

Section 4 states the full-abstraction result, and provides part of its long and complex proof.

Section 5 discusses related work.

Section 6 concludes the paper with some final remarks, in particular about the lack of compositionality of the encoding due to the global integrity manager.

The appendix provides those parts of the proof of the full-abstraction result that have been relegated from Section 4.

2 Background: π-calculus, πT-calculus

A simplified version of the applied calculus, denoted πT, is presented next. The π-calculus dialect used for implementing πT in Section 3 is a subset of πT modulo a few simple modifications – see Subsection 2.2. Subsection 2.3 is concerned with recovering the spi calculus

from πT. This will serve as an illustration for applying the π-calculus implementation of πT to straightline processes.

We designate finite vectors by means of the $\widetilde{\ }$-symbol. The length of any finite vector $\widetilde{X}$ is denoted by $|\widetilde{X}|$.

2.1 The πT-calculus

As always in π-like calculi, an infinite set of *names* is assumed to be given. This set is here typically ranged over by lower-case letters from the middle of the alphabet, such as n. Sets of names are typically ranged over by upper-case letters from the middle of the alphabet, such as N. As usual in spi-calculus-like extensions of the π-calculus we distinguish names and *variables*. The set of variables is also assumed to be infinite. It is here typically ranged over by lower-case letters from the end of the alphabet, such as x.

The πT-calculus is further characterised by assuming a set of function symbols from which data terms can be formed, distinguishing dedicated *constructors* and *deconstructors* (with minor restrictions, see below). Predictably, constructors are used to build data terms and deconstructors to take them apart. Both the set of constructors and the set of deconstructors are assumed to be finite. Constructors are typically ranged over by f, deconstructors by d. Each constructor f is assumed to have a fixed, finite *arity* $\mathrm{ar}[f]$.

Data terms are then given as follows:

$$T, \ldots ::= n \mid x \mid f(T_1, \ldots, T_{\mathrm{ar}[f]})$$

For the sake of simplicity, no type discipline is assumed for data terms, only the constructor arities must be respected. The set of all names that occur in any given vector $\widetilde{T}$ of data terms is denoted by $\mathrm{nm}[\widetilde{T}]$. Deconstructors must not occur within data terms. They may only be applied via the **let** construct introduced below. A *value* is a data term without any variables. Values are typically ranged over by V and W.

Agent expressions are then given as follows:

$$\begin{aligned} P, \ldots ::= {} & \mathbf{0} \mid T(x).P \mid \overline{T}\langle U\rangle.P \mid P \mid Q \mid (\nu\, n)\, P \mid !\, P \\ \mid {} & \ \mathbf{if}\ T = U\ \mathbf{then}\ P\ \mathbf{else}\ Q \mid \mathbf{let}\ x = d(\widetilde{T})\ \mathbf{in}\ P \end{aligned}$$

The $\mathbf{0}$ constant denotes the inert process. An input prefix $T(x).P$ can be performed in a context where T evaluates to some name n. Then a value, e.g. V, is received over the channel denoted by n and the process continues as P where x is substituted by V. An output prefix $\overline{T}\langle U\rangle.P$ can also be performed in a context where T evaluates to some name n and U to some value V. Then V is sent via the channel denoted by n and the process continues as P. Here n and T are called the *subject*, and x and V the *object* of the input or output. The next three operators are standard in pi-calculus-like calculi: $P \mid Q$ behaves like P and Q acting concurrently; $(\nu\, n)\, P$ behaves like P where n is local; $!\, P$ behaves like infinitely many copies of P put in parallel. The conditional and the **let** have their usual meaning. In πT, **let** is the only place where deconstructors may be applied.

Input prefixing, **let** , and restriction are *binders*: $T(x).P$ and $\mathbf{let}\ x = d(\widetilde{T})\ \mathbf{in}\ P$ bind the variable x in P; $(\nu\, n)\, P$ binds the name n in P. The set of names occurring free (non-bound) in a process expression P is denoted by $\mathrm{fn}[P]$ and similarly for a vector $\widetilde{P} = P_1 \ldots P_n$ of process expressions $\mathrm{fn}[\widetilde{P}]$ means the union of all $\mathrm{fn}[P_i]$.

We do not distinguish between expressions that differ only up to alpha conversion of bound names and variables. Given any data term or process expression H, $H\{\widetilde{x} := \widetilde{T}\}$ denotes the

term after simultaneously substituting each x_i by T_i. Substitution always entails implicit alpha-conversion of bound names in H such that there is no capture of any free names in $\widetilde{T}$. A *process* is an agent expression without free variables.

We often omit trailing $\mathbf{0}$ suffixes, writing $x\langle y\rangle$ for $x\langle y\rangle.\mathbf{0}$. We often also use an "inline" notation for restriction, and e.g. write $\overline{x}\langle\nu\, y\rangle.P$ as shorthand for $(\nu\, y)\,\overline{x}\langle y\rangle.P$. Further, we use the standard notation $\prod_{i\in\{j_1,\ldots,j_k\}} P_i = P_{j_1} \mid \ldots \mid P_{j_k}$.

2.1.1 Deconstructor Equations

Deconstructor equations describe how deconstructors act upon values. To this end, we need to introduce *value patterns*:

$$G ::= x \mid f(G_1, \ldots, G_{\mathrm{ar}[f]})$$

A deconstructor equation is then of the form $d(\widetilde{G}) \stackrel{\Delta}{=} x$ where x must occur in $\widetilde{G}$. There may be several, but only finitely many equations for each deconstructor. Since there are finitely many deconstructors, there are only finitely many deconstructor equations.

2.1.2 Operational Semantics

Abadi's and Fournet's semantics for the applied π-calculus [AF01] use active substitutions and rely heavily on structural congruence rules. Our semantics, using the **let** construct for deconstruction, is more direct and does not utilise structural congruence. The rules for scope opening are similar to the variant in [AF01] giving the finest bisimulation equivalence relation; we are dealing with may testing where this finesse does not matter.

Actions are of one of three forms:

$n(V)$: Input of value V on channel n where V is bound to x as shown below.

$(\nu\, M)\,\overline{n}\langle V\rangle$: Output of value V on channel n where the names in M are extruded as private names. The SOS clauses ensure that we always have $M \subseteq \mathrm{fn}(V)$.

τ: Silent action.

The individual clauses are as shown in Table 1 on the following page. The treatment of replication follows [SW03]. The clauses for **if** _ **then** _ **else** are slightly non-standard since they entail a τ-action, just like in [AF01]. This is advantageous for our purposes of considering may-testing. The missing symmetric clauses for interleaving and closure are left implicit. We denote by $\mathrm{nm}[\alpha]$ the set of names that syntactically occur in any action α; we denote by $\mathrm{bn}[\alpha]$ the set of bound names of α: $\mathrm{bn}[n(x)] = \emptyset$, $\mathrm{bn}[(\nu\, M)\,\overline{n}\langle T\rangle] = M$, $\mathrm{bn}[\tau] = \emptyset$; $(\nu\,\{n_1, \ldots, n_k\})$ stands for $(\nu\, n_1)\ldots(\nu\, n_k)$, $k \geq 0$. Also, we denote mappings from variables to values by σ, and by $\widetilde{T}\sigma$ the vector that results from applying σ to each element of $\widetilde{T}$.

2.2 Polyadic π-Calculus

In the next section we will encode πT into the polyadic π-calculus. The specific dialect that we use is derived from πT via three simple modifications: First, the sets of constructors and deconstructors are assumed to be empty; second, nondeterministic CCS-like choice of the form $P + Q$ and polyadic input and output prefixes of the form $a(\widetilde{x}).P$ or $\overline{a}\langle\widetilde{b}\rangle$, respectively, are admitted, where a and b range over both names and variables; third, *process constants*

Table 1 SOS clauses for the πT-calculus.

(IN), (OUT) $$n(x).P \xrightarrow{n(V)} P\{x := V\} \quad \overline{n}\langle V\rangle.P \xrightarrow{(\nu)\,\overline{n}\langle V\rangle} P$$

($|_1$), (CLOSE$_1$) $$\frac{P \xrightarrow{\alpha} P' \quad \mathrm{bn}[\alpha] \cap \mathrm{fn}[Q] = \emptyset}{P \mid Q \xrightarrow{\alpha} P' \mid Q} \qquad \frac{P \xrightarrow{n(V)} P' \quad Q \xrightarrow{(\nu\, M)\,\overline{n}\langle V\rangle} Q' \quad \mathrm{fn}[P] \cap M = \emptyset}{P \mid Q \xrightarrow{\tau} (\nu\, M)(P' \mid Q')}$$

(ν), (OPEN) $$\frac{P \xrightarrow{\alpha} P' \quad n \notin \mathrm{nm}[\alpha]}{(\nu\, n)\, P \xrightarrow{\alpha} (\nu\, n)\, P'} \qquad \frac{P \xrightarrow{(\nu\, M)\,\overline{n}\langle V\rangle} P' \quad n \neq m' \quad m' \in \mathrm{nm}[V] \setminus M}{(\nu\, m')\, P \xrightarrow{(\nu\, M + m')\,\overline{n}\langle V\rangle} P'}$$

(!-|), (!-CLOSE) $$\frac{P \xrightarrow{\alpha} P' \quad \mathrm{bn}[\alpha] \cap \mathrm{fn}[P] = \emptyset}{!\,P \xrightarrow{\alpha} P' \mid !\,P} \qquad \frac{P \xrightarrow{n(V)} P_1' \quad P \xrightarrow{(\nu\, M)\,\overline{n}\langle V\rangle} P_2' \quad \mathrm{fn}[P] \cap M = \emptyset}{!\,P \xrightarrow{\tau} ((\nu\, M)(P_1' \mid P_2')) \mid !\,P}$$

(**if**-$\top$), (**if**-$\bot$) $$\textbf{if } T = T \textbf{ then } P \textbf{ else } Q \xrightarrow{\tau} P \qquad \frac{T \neq U}{\textbf{if } T = U \textbf{ then } P \textbf{ else } Q \xrightarrow{\tau} Q}$$

(**let**) $$\frac{d(\widetilde{G}) \stackrel{\Delta}{=} x \quad \widetilde{G}\sigma = \widetilde{V}}{\textbf{let } y = d(\widetilde{V}) \textbf{ in } P \xrightarrow{\tau} P\{y := x\sigma\}}$$

are admitted. They are typically ranged over by upper-case letters from the beginning of the alphabet, such as A. *Process constant definitions* are of the form $A(\widetilde{x}) \stackrel{\Delta}{=} P$ where the free variables of P must be from $\widetilde{x}$. There must be a unique definition for each process constant. The number of actual parameters in each instantiation of any process constant must be the same as the number of formal parameters in the constant's definition. The necessary modifications of the above SOS clauses and the accompanying notion of action are standard. Here, we just give the semantics of the nondeterministic choice operator:

$$\frac{P \xrightarrow{\alpha} P'}{P + Q \xrightarrow{\alpha} P'} \qquad \frac{Q \xrightarrow{\alpha} Q'}{P + Q \xrightarrow{\alpha} Q'}$$

As α ranges over all actions including τ, $+$ is the ordinary CCS-like nonderministic choice operator.

2.3 Communication Security in πT

To finish this section, we show how the spi calculus can be obtained as an instance of πT. First of all, the constructors are fixed. Each constructor is stated together with its arity.

$0 : 0$ / *succ* $: 1$	*zero / successor*
pair $: 2$	*pair formation*
s›*enc* $: 2$ / *a*›*enc* $: 2$	*symmetric / asymmetric encryption*
priv $: 1$ / *pub* $: 1$	*private / public key*
hashc $: 1$	*hashcode*

Next the deconstructors and deconstructor equations are fixed. It suffixes to display only the equations:

$$\begin{array}{ll}
pred(succ(x)) \stackrel{\Delta}{=} x & \textit{predecessor} \\
fst(pair(x,y)) \stackrel{\Delta}{=} x & \textit{projection onto first /} \\
scd(pair(x,y)) \stackrel{\Delta}{=} y & \textit{second component} \\
dec(s{\rhd}enc(p,k),k) \stackrel{\Delta}{=} p & \textit{symmetric decryption} \\
dec(a{\rhd}enc(p,pub(x)),priv(x)) \stackrel{\Delta}{=} p & \textit{asymmetric decryption} \\
dec(a{\rhd}enc(p,priv(x)),pub(x)) \stackrel{\Delta}{=} p &
\end{array}$$

The constructs of the spi calculus can then be represented either one-to-one or by means of very simple combination of πT constructs. For example,

$$\textbf{case } T \textbf{ of } 0 : P \; succ(x) : Q$$

can be translated to

$$\textbf{if } T = 0 \textbf{ then } P \textbf{ else let } x = pred(T) \textbf{ in } Q.$$

3 The encoding of the πT-calculus

In this section we present the encoding of πT-processes into the polyadic π-calculus. As in previous work [BPV04], the main issue is encoding values.

The idea is to let encoded processes operate on encoded values only via *value identifiers* (IDs). To this end, an *integrity manager* **M** is set up, which maintains tables of the existing IDs and the values these correspond to. Whenever an encoded process wants to operate on one or more encoded values, it must send a specific request carrying the value IDs to **M**. **M** will check whether it already knows the IDs, i.e., whether the IDs correspond to any actual values: if that is the case, then **M** will perform the operation; if not, then the request will deadlock. If the operation requires generating a new value, **M** will do so, and it will also generate a new ID that it will associate with the value. Completion will be signalled on a dedicated channel that has been supplied when the request was issued. The channel carries an ID which refers to the result of the operation, if there was any.

M thus shields encoded processes from any malformed complex values that might be sent to them by π-observers which are not encodings of πT processes. Moreover, as πT- and π-processes have the same pure synchronisation capabilities, π-observers are unable to do anything to encoded πT-processes that encoded πT-observers cannot already do. The only requirement is that no encoded πT-process may ever get connected to a "fake" integrity manager. This is assured by (a) restricting the channels that are used for interacting with **M** and (b) setting up a uni-directional firewall **F** via which **M** can receive requests from an external observer via duplicate, non-restricted channels.

The encoding of πT processes is constructed using one encoding function $(\!|\cdot|\!)$ for processes as such (section 3.1), and one encoding function $(\!|\cdot|\!)_r$ of values occurring in the processes (section 3.2), parameterised by a location r used to represent the value. **M** is presented in section 3.3.

The full encoding of a πT process P is thus $[\![P]\!] = (\nu\, \mathrm{K}_{\mathbf{M}})((\!|P|\!) \mid \mathbf{M} \mid \mathbf{F})$, and our main theorem

$$P \eqsim_{\mathrm{may}} Q \text{ iff } [\![P]\!] \eqsim_{\mathrm{may}} [\![Q]\!].$$

The encoded processes communicate with $\mathbf{M}$ using a simple protocol over the free names of $\mathbf{M}$, which we denote by $\mathrm{K}_{\mathbf{M}}$. Below we describe their use and the parameters passed over them.

input : (i, r) i is the ID of the channel the process wants to input a value over; r is a fresh name over which the process will receive the value (itself an ID).

output : (i, j, r) i is the ID of the channel the process wants to output a value over; j is the ID of the value to be sent; r is a fresh name which will be used for synchronisation when the output has been performed by $\mathbf{M}$.

$apply_E$: $(\widetilde{i}, r)$ (for each deconstructor equation $E : d(G_1, \ldots, G_k) = x$, where $|\widetilde{i}| = k$). $\widetilde{i}$ are the IDs of the actual components of the value pattern; r is a fresh name where the ID of the value corresponding to x (if any) can be received by the process.

r : (i) (for each value occurring in the process). Each value used in the process is represented by a fresh name r, where the process can receive the ID of the value, e.g. to pass in the above operations.

The encoded values register with $\mathbf{M}$ by communicating with it over two additional channels:

new : (n, r) n is a πT name occurring in the source process; r is a name where the ID of the name can be received.

new_f : $(\widetilde{x}, r)$ (for each constructor f, where $|\widetilde{x}| = \mathrm{ar}[f]$). $\widetilde{x}$ are the actual components of the constructor f; r is a name where the ID of the constructed value $f(\widetilde{x})$ can be received.

3.1 Encoding Processes

The encoding of processes is homomorphic for $\mathbf{0}, |, (\nu)$ and ! operators:

$$(\!|\mathbf{0}|\!) = \mathbf{0} \quad (\!|P \mid Q|\!) = (\!|P|\!) \mid (\!|Q|\!) \quad (\!|(\nu\, n)\, P|\!) = (\nu\, n)(\!|P|\!) \quad (\!|!\, P|\!) = !(\!|P|\!)$$

Input and output prefixes (below) encode the subject, request its ID, and contacts $\mathbf{M}$ for doing the actual input or output. In the case of input, the response channel s is used for performing the input, binding the object x, while in the case of output, $\mathbf{M}$ performs the output and responds with a pure synchronisation. r, s, t, i and j are fresh.

$$\begin{array}{rcl} (\!|T(x).P|\!) & = & (\nu\, r)((\!|T|\!)_r \mid r(i).\overline{input}\langle i, \nu\, s\rangle.s(x).(\!|P|\!)) \\ (\!|\overline{T}\langle U\rangle.P|\!) & = & (\nu\, r, s)((\!|T|\!)_r \mid (\!|U|\!)_s \mid r(i).s(j).\overline{output}\langle i, j, \nu\, t\rangle.t().(\!|P|\!)) \end{array}$$

The conditional retrieves the IDs of the encodings of the compared values and tests them for identity. ($\mathbf{M}$ ensures that identical values have the same ID.) r, s, i and j are fresh.

$$(\!|\mathbf{if}\ T = U\ \mathbf{then}\ P\ \mathbf{else}\ Q|\!) = (\nu\, r, s)((\!|T|\!)_r \mid (\!|U|\!)_s \mid r(i).s(j).\mathbf{if}\ i = j\ \mathbf{then}\ (\!|P|\!)\ \mathbf{else}\ (\!|Q|\!))$$

The **let** processes use a deconstructor. Each deconstructor d of arity k has a set of associated deconstructor equations $\mathbb{E}_{d,k}$. Each such equation E has a handler in $\mathbf{M}$, contacted over $apply_E$. In the encoding of **let** below (left), all such handlers are applied in parallel (line 2), and the first one to complete (line 3) locks the others out (using the one-time lock u on line 1). $\widetilde{r}, \widetilde{i}$ and u are fresh.

$$
\begin{aligned}
(\!|\textbf{let } x = d(T_1,\ldots,T_k) \textbf{ in } P|\!) = (\nu\, r_1,\ldots,r_k)(\;& \\
& (\!|T_1|\!)_{r_1} \mid \ldots \mid (\!|T_k|\!)_{r_k} \\
\mid\; & r_1(i_1).\ \ldots\ r_k(i_k). \\
& (\nu\, u)(\\
& \quad \overline{u}\langle\rangle && (1)\\
& \mid\; \textstyle\prod_{E\in\mathbb{E}_{d,k}}(\\
& \quad \overline{apply_E}\langle i_1,\ldots,i_k,\nu\, r\rangle. && (2)\\
& \quad r(x).u().(\!|P|\!)) && (3)
\end{aligned}
$$

$$
\begin{aligned}
(\!|f(T_1,\ldots,T_{\mathrm{ar}[f]})|\!)_r &= (\nu\, r_1,\ldots,r_{\mathrm{ar}[f]})(\\
& \quad (\!|T_1|\!)_{r_1} \mid \ldots \mid (\!|T_{\mathrm{ar}[f]}|\!)_{r_{\mathrm{ar}[f]}} && (4)\\
& \mid\; r_1(i_1).\ \ldots\ r_{\mathrm{ar}[f]}(i_{\mathrm{ar}[f]}). && (5)\\
& \quad \overline{new_f}\langle i_1,\ldots,i_{\mathrm{ar}[f]},r\rangle) && (6)\\
(\!|n|\!)_r &= \overline{new}\langle n, r\rangle \\
(\!|x|\!)_r &= \overline{r}\langle x\rangle
\end{aligned}
$$

3.2 Encoding Value Terms

Value terms come in three kinds: constructor terms, names, and variables. Their encoding is above (right). Terms using a constructor f encode their component values (line 4), and supply their IDs (line 5) to the new_f handler (line 6). Names are encoded by calling the new handler of **M**, while occurrences of variables will always be substituted at runtime by an ID, which will then be returned.

3.3 Integrity Manager M

M is shown in Figure 1. The names *input*, *output*, *new*, new_f for all $f \in \mathbb{F}$, and $apply_E$ for all $E \in \mathbb{E}$ are free in **M**. They may be used by π-calculus observers to interact with any encoded process, but thanks to the firewall, they can not interfere in the communication between the encoded processes and **M**.

M uses an initially empty value table (**EmptyValTbl**) to maintain the correspondence between values and their value IDs. Each name in the table has an *alias*, used for the actual input and output, such that **M** can monitor all values passed, making sure they are value IDs. For each constructor in the table, the IDs of its subcomponents are maintained. The **Mutex** is used as a mutual exclusion lock for the table. (The reader may refer to [BPV05] for parts that have to be left out here due to a lack of space.)

Additional components of **M** insert new values into the tables, if necessary:

NameRegistrar The first time a name is used in the encoded process, the name registrar adds its corresponding ID and alias to the name table. Later uses of the name only result in a lookup; the ID and alias is maintained.

ConsTermRegistrar$_f$ (for each $f \in \mathbb{F}$). The constructor term registrars perform the corresponding function for values built by constructors f; the ID of the value and its components is maintained, and together the name and constructor term registrars ensure that identical values have the same ID.

Figure 1 Defining equation of integrity manager and empty value table.

$$
\begin{aligned}
\mathbf{M} = (\nu\, & lock, unlock, put, getId, getAlias, [\forall f \in \mathbb{F}.\, put_f, getId_f, getArgIds_f], lookUp)(\\
& \mathbf{EVT} \\
& \mid \mathbf{NameRegistrar}(lock, unlock, put, getId, lookUp, new) \\
& \mid \textstyle\prod_{f\in\mathbb{F}} \mathbf{ConsTermRegistrar}_f(lock, unlock, put_f, getId_f, new_f) \\
& \mid \mathbf{InputInterpreter}(lock, unlock, getAlias, input) \\
& \mid \mathbf{OutputInterpreter}(lock, unlock, getAlias, lookUp, output) \\
& \mid \textstyle\prod_{E\in\mathbb{E}} \mathbf{EquationInterpreter}_E(lock, unlock, [\forall f \in \mathbb{F}.getArgIds_f], apply_E) \\
& \mid \mathbf{Mutex}(lock, unlock))
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{EmptyValTbl}(put, getId, getAlias, & [\forall f \in \mathbb{F}.put_f, getId_f, getArgIds_f], lookUp) \triangleq \\
& put(i, n, a). \\
& (\nu\, getId', getAlias', [\forall f \in \mathbb{F}.\, getId'_f, getArgIds'_f], lookup')(\\
& \quad \mathbf{NameRecord}(\\
& \quad\quad i, n, a, getId, getId', getAlias, getAlias', \\
& \quad\quad [\forall f \in \mathbb{F}.getId_f, getId'_f, getArgIds_f, getArgIds'_f], lookUp, lookup') \\
& \quad \mid \mathbf{EVT}') \\
& + put_f(i, \widetilde{i'} : \mathrm{ar}[f]). \\
& \quad (\nu\, getId', getAlias', [\forall f \in \mathbb{F}.\, getId'_f, getArgIds'_f], lookup')(\\
& \quad \mathbf{ConsTermRecord}_f(\\
& \quad\quad i, \widetilde{i'}, getId, getId', getAlias, getAlias', \\
& \quad\quad [\forall f \in \mathbb{F}.getId_f, getId'_f, getArgIds_f, getArgIds'_f], lookUp, lookup') \\
& \quad \mid \mathbf{EVT}') \\
& + getId(m, r^+, r^-).\overline{r^-}\langle\rangle.\mathbf{EVT} + getAlias(j, r^+, r^-).\overline{r^-}\langle\rangle.\mathbf{EVT} \\
& + [\Sigma_{f\in\mathbb{F}} getId_f(\widetilde{j} : \mathrm{ar}[f], r^+, r^-).\overline{r^-}\langle\rangle.\mathbf{EVT}] + getArgIds(j, r^+, r^-).\overline{r^-}\langle\rangle.\mathbf{EVT} \\
& + lookUp(j, r^+, r^-).\overline{r^-}\langle\rangle.\mathbf{EVT}
\end{aligned}
$$

We denote by $\widetilde{x} : k$ a variable vector $\widetilde{x}$ of length k,
$\mathbf{EVT} = \mathbf{EmptyValTbl}(put, getId, getAlias, [\forall f \in \mathbb{F}.getId_f, getArgIds_f], lookUp)$,
$\mathbf{EVT}' = \mathbf{EmptyValTbl}(put, getId', getAlias', [\forall f \in \mathbb{F}.getId'_f, getArgIds'_f], lookUp')$,
and $\mathbf{Mutex}(lock, unlock) \triangleq \overline{lock}\langle\rangle.unlock().\mathbf{Mutex}(lock, unlock)$.

The remaining components of **M** are (1) the handlers for input and output, **InputInterpreter** and **OutputInterpreter**, which ensure that all names used as channels, and values passed over them, have appropriate table entries and (2) the deconstructor equation handlers $\mathbf{EquationInterpreter}_E$, used to match a deconstructor application against deconstructor equations.

3.3.1 Value Table

IDs of names and constructors are added to the value table by communication over *put* and put_f, respectively; they can be retrieved over *getId* and $getId_f$. The *lookUp* is used to verify that an ID is in the table. Subcomponents of constructors can be retrieved over *getArgIds*.

The table is built by appending table cells to the initial empty table. Such a cell is either a **NameRecord** or a $\mathbf{ConsTermRecord}_f$ ($f \in \mathbb{F}$, see Figure 2 on the next page). All requests to retrieve information have among their parameters the names r^+, r^-, which are used for the response. Each cell checks if it should handle the request, and if not, passes it on to the next cell. If no cell handles the request, the **EmptyValTbl** eventually signals on r^-.

Figure 2 Defining equation for value entries.

$$
\begin{aligned}
&\mathbf{NameRecord}(\\
&\quad i, n, a,\\
&\quad \mathit{getId}, \mathit{getId}', \mathit{getAlias}, \mathit{getAlias}',\\
&\quad [\forall f \in \mathbb{F}.\mathit{getId}_f, \mathit{getId}'_f, \mathit{getArgIds}_f,\\
&\quad \mathit{getArgIds}'_f], \mathit{lookUp}, \mathit{lookUp}'\\
&) \triangleq \mathit{getId}(m, r^+, r^-).\\
&\quad \textbf{if } m = n \textbf{ then } \overline{r^+}\langle i\rangle.\mathbf{NE}\\
&\quad \textbf{else } \overline{\mathit{getId}'}\langle m, r^+, r^-\rangle.\mathbf{NE}\\
&\quad + \mathit{getAlias}(j, r^+, r^-).\\
&\quad \textbf{if } j = i \textbf{ then } \overline{r^+}\langle a\rangle.\mathbf{NE}\\
&\quad \textbf{else } \overline{\mathit{getAlias}'}\langle j, r^+, r^-\rangle.\mathbf{NE}\\
&\quad + \textstyle\sum_{f\in\mathbb{F}}(\\
&\qquad \underline{\mathit{getId}_f}(\widetilde{j} : \mathrm{ar}[f], r^+, r^-).\\
&\qquad \overline{\mathit{getId}'_f}\langle \widetilde{j}, r^+, r^-\rangle.\mathbf{NE}\\
&\qquad + \underline{\mathit{getArgIds}_f}(j, r^+, r^-).\\
&\qquad \overline{\mathit{getArgIds}'_f}\langle j, r^+, r^-\rangle.\mathbf{NE})\\
&\quad + \mathit{lookUp}(j, r^+, r^-).\\
&\quad \textbf{if } j = i \textbf{ then } \overline{r^+}\langle\rangle.\mathbf{NE}\\
&\quad \textbf{else } \overline{\mathit{lookUp}'}\langle j, r^+, r^-\rangle.\mathbf{NE}
\end{aligned}
$$

The shorthands **NE** and $\mathbf{CTE}_f$ stand for recursive invocation with unchanged parameters. The shorthand **if** $x_1, \cdots, x_k = y_1, \cdots, y_k$ **then** P **else** Q ($k \geq 2$) stands for **if** $x_1 = y_1$ **then** (**if** $x_1, \cdots, x_k = y_1, \cdots, y_k$ **then** P **else** Q) **else** Q.

$$
\begin{aligned}
&\mathbf{ConsTermRecord}_f(\\
&\quad i, \widetilde{i}' : \mathrm{ar}[f],\\
&\quad \mathit{getId}, \mathit{getId}', \mathit{getAlias}, \mathit{getAlias}',\\
&\quad [\forall f \in \mathbb{F}.\mathit{getId}_f, \mathit{getId}'_f, \mathit{getArgIds}_f,\\
&\quad \mathit{getArgIds}'_f], \mathit{lookUp}, \mathit{lookUp}'\\
&) \triangleq \mathit{getId}(m, r^+, r^-).\\
&\quad \overline{\mathit{getId}'}\langle m, r^+, r^-\rangle.\mathbf{CTE}_f\\
&\quad + \mathit{getAlias}(j, r^+, r^-).\\
&\quad \overline{\mathit{getAlias}'}\langle j, r^+, r^-\rangle.\mathbf{CTE}_f\\
&\quad + \mathit{getId}_f(\widetilde{j} : \mathrm{ar}[f], r^+, r^-).\\
&\quad \textbf{if } \widetilde{j} = \widetilde{i}' \textbf{ then } \overline{r^+}\langle i\rangle.\mathbf{CTE}_f\\
&\quad \textbf{else } \overline{\mathit{getId}'_f}\langle \widetilde{j}, r^+, r^-\rangle.\mathbf{CTE}_f\\
&\quad + \mathit{getArgIds}_f(j, r^+, r^-).\\
&\quad \textbf{if } j = i \textbf{ then } \overline{r^+}\langle \widetilde{i}'\rangle.\mathbf{CTE}_f\\
&\quad \textbf{else } \overline{\mathit{getArgIds}'_f}\langle j, r^+, r^-\rangle.\mathbf{CTE}_f\\
&\quad + \textstyle\sum_{g\,\in\,\mathbb{F},\ g\neq f}(\\
&\qquad \underline{\mathit{getId}_g}(\widetilde{j} : \mathrm{ar}[g], r^+, r^-).\\
&\qquad \overline{\mathit{getId}'_g}\langle \widetilde{j}, r^+, r^-\rangle.\mathbf{CTE}_f\\
&\qquad + \underline{\mathit{getArgIds}_g}(j, r^+, r^-).\\
&\qquad \overline{\mathit{getArgIds}'_g}\langle j, r^+, r^-\rangle.\mathbf{CTE}_f)\\
&\quad + \mathit{lookUp}(j, r^+, r^-).\\
&\quad \textbf{if } j = i \textbf{ then } \overline{r^+}\langle\rangle.\mathbf{CTE}_f\\
&\quad \textbf{else } \overline{\mathit{lookUp}'}\langle j, r^+, r^-\rangle.\mathbf{CTE}_f
\end{aligned}
$$

3.3.1.1 Name Registrar Name records maintain the IDs and *aliases* of names. The latter are the channels used internally for communication, to make sure all objects passed are value IDs. The *name registrar* (Figure 3 on the following page) first checks if the supplied name is a value ID, supplied by a "malicious" observer: this is an error, and the requester gets no reply. If it is not a value ID, it checks if the name is in the name table; in that case it returns its ID, otherwise it adds the name together with its new ID and alias to the name table, adds the ID to the ID table (since all IDs used must be there), and responds with the ID.

3.3.1.2 Constructor Term Registrars The registrar for a constructor f (Figure 3) is a variation of the name registrar: it checks that each component value is in the ID table (and thus is a valid value in the encoding), and if they are, either returns the ID of the constructed term (if it is already there) or puts in an association between the constructed term and its (new) ID, puts the ID in the ID table, and returns it.

Figure 3 Name and constructor term registrars

$$
\begin{array}{l}
\mathbf{NameRegistrar}(\\
\quad lock, unlock, put, getId, lookUp, new \\
) \triangleq\ ! \ \ new(n, r).\overline{lock}\langle\rangle \\
\qquad \overline{lookUp}\langle n, \nu\, s^+, \nu\, s^-\rangle. \\
\qquad (\ \ s^+().\overline{unlock}\langle\rangle \\
\qquad + s^-(). \\
\qquad\quad \overline{getId}\langle n, \nu\, t^+, \nu\, t^-\rangle. \\
\qquad\quad (\ \ t^+(i).\overline{unlock}\langle\rangle.\overline{r}\langle i\rangle \\
\qquad\quad + t^-().\overline{put}\langle\nu\, i, n, \nu\, a\rangle. \\
\qquad\quad\quad \overline{unlock}\langle\rangle.\overline{r}\langle i\rangle))
\end{array}
$$

$$
\begin{array}{l}
\mathbf{ConsTermRegistrar}_f(\\
\quad lock, unlock, put, getId, lookUp, new \\
) \triangleq\ ! \ \ new(i'_1, \ldots, i'_{\mathrm{ar}[f]}, r).lock(). \\
\qquad \overline{lookUp}\langle i'_1, \nu\, s_1^+, \nu\, s_1^-\rangle. \\
\qquad (\ \ s_1^-().\overline{unlock}\langle\rangle \\
\qquad + s_1^+(). \\
\qquad\quad \vdots \\
\qquad\quad\quad \overline{lookUp}\langle i'_{\mathrm{ar}[f]}, \nu\, s^+_{\mathrm{ar}[f]}, \nu\, s^-_{\mathrm{ar}[f]}\rangle. \\
\qquad\quad\quad (\ \ s^-_{\mathrm{ar}[f]}().\overline{unlock}\langle\rangle \\
\qquad\quad\quad + s^+_{\mathrm{ar}[f]}(). \\
\qquad\quad\quad\quad \overline{getId}\langle i'_1, \ldots, i'_{\mathrm{ar}[f]}, \nu\, t^+, \nu\, t^-\rangle. \\
\qquad\quad\quad\quad (\ \ t^+(i).\overline{unlock}\langle\rangle.\overline{r}\langle i\rangle \\
\qquad\quad\quad\quad + t^-().\overline{put}\langle\nu\, i, i'_1, \ldots, i'_{\mathrm{ar}[f]}\rangle. \\
\qquad\quad\quad\quad\quad \overline{unlock}\langle\rangle.\overline{r}\langle i\rangle)) \ldots)
\end{array}
$$

Figure 4 Equation interpreter.

$$
\begin{array}{l}
\mathbf{EquationInterpreter}_E(lock, unlock, [\forall f \in \mathbb{F}.getArgIds_f], lookUp, apply_E) \triangleq \\
\quad ! \ \ apply_E(i_1, \ldots, i_{\mathrm{ar}[E]}, r).lock().\overline{lookUp}\langle i_1, \nu\, s_1^+, \nu\, s_1^-\rangle \\
\qquad (\ \ s_1^-().\overline{unlock}\langle\rangle \\
\qquad + s_1^+(). \\
\qquad\quad \vdots \\
\qquad\quad\quad \overline{lookUp}\langle i_{\mathrm{ar}[E]}, \nu\, s^+_{\mathrm{ar}[E]}, \nu\, s^-_{\mathrm{ar}[E]}\rangle. \\
\qquad\quad\quad (\ \ s^-_{\mathrm{ar}[f]}().\overline{unlock}\langle\rangle \\
\qquad\quad\quad + s^+_{\mathrm{ar}[f]}().(\!|G_1, i_1, \ldots, G_{\mathrm{ar}[E]}, i_{\mathrm{ar}[E]}, \emptyset|\!)) \cdots)
\end{array}
$$

3.3.1.3 Equation Interpreters Given an equation $E : d(G_1, \ldots, G_{\mathrm{ar}[E]}) \triangleq x$, an interpreter for E is given a vector of $\mathrm{ar}[E]$ actual value IDs, and tries to match each value ID against the corresponding value pattern G_i of the equation (Figure 4). If it succeeds, the value ID corresponding to x of the equation is returned over the result channel r. If it fails, no result is given.

Given $E : d(G_1, \ldots, G_{\mathrm{ar}[E]}) \triangleq x$, where the $(\!|H_1, j_1, \ldots, H_k, j_k, S|\!)$-construct, $k \geq 0$, handles the matching of actual component values against components of the equation. Its definition is recursive.

1. If $k = 0$, then $(\!|S|\!) = \overline{unlock}\langle\rangle.\overline{r}\langle x\rangle$.
2. If $k \geq 1$ and $H_1 = z$ for some variable $z \notin S$,
   $(\!|H_1, j_1, \ldots, H_k, j_k, S|\!) = (\!|H_2, j_2, \ldots, H_k, j_k, S \cup \{z\}|\!)\{z := j_1\}$
3. If $k \geq 1$ and $H_1 = z$ for some variable $z \in S$,
   $(\!|H_1, j_1, \ldots, H_k, j_k, S|\!) = \mathbf{if}\ z = j_1\ \mathbf{then}\ (\!|H_2, j_2, \ldots, H_k, j_k, S|\!)$
   $\mathbf{else}\ \overline{unlock}\langle\rangle$

4. If $k \geq 1$ and $H_1 = f(H'_1, \ldots, H'_{\mathrm{ar}[f]})$ for some constructor f and arguments H'_1, …, $H'_{\mathrm{ar}[f]}$, then:
$(\!|H_1, j_1, \ldots, H_k, j_k, S|\!) = \overline{getArgIds_f}\langle j_1, \nu\, t^+, \nu\, t^-\rangle.$
$(\ t^-().\overline{unlock}\langle\rangle$
$+\ t^+(j'_1, \ldots, j'_{\mathrm{ar}[f]}).$
$(\!|H'_1, j'_1, \ldots, H'_{\mathrm{ar}[f]}, j'_{\mathrm{ar}[f]}, H_2, j_2, \ldots, H_k, j_k, S|\!))$
where j'_1, …, $j'_{\mathrm{ar}[f]}$ are fresh.

3.3.1.4 Communication Interpreters The communication interpreters handle the *input* and *output* requests to **M**. Both look up the *alias* of the subject channel in the name table; the input handler returns this so the encoded input prefix can perform the input and bind the object variable in the correct context; the output handler looks up the object ID and performs the output on behalf of the encoded output prefix, and synchronises on the result channel when done.

InputInterpreter(
$lock, unlock, getAlias, input$
) $\triangleq$! $input(i, r).lock().$
$\overline{getAlias}\langle i, \nu\, s^+, \nu\, s^-\rangle.$
$(\ s^-().\overline{unlock}\langle\rangle$
$+\ s^+(a).\overline{unlock}\langle\rangle.a(x).\overline{r}\langle x\rangle)$

OutputInterpreter(
$lock, unlock, getAlias, lookUp, output$
) $\triangleq$! $\overline{output}\langle i, j, r\rangle.lock().$
$\overline{getAlias}\langle i, \nu\, s^+, \nu\, s^-\rangle.$
$(\ s^-().\overline{unlock}\langle\rangle$
$+\ s^+(a).\overline{lookUp}\langle j, \nu\, t^+, \nu\, t^-\rangle$
$(\ t^-().\overline{unlock}\langle\rangle$
$+\ t^+().\overline{unlock}\langle\rangle.\overline{a}\langle j\rangle.\overline{r}\langle\rangle)))$

3.4 Firewall

For every free name n of **M**, we introduce a distinct fresh name n' via which external observers may interact with **M** and thus indirectly also with any encoded πT-process. The firewall **F** receives requests on the channels named with those fresh names and encodes them to requests on the corresponding internal channels.

Its defining equation is as follows:

$$\begin{aligned}\mathbf{F} = !\ \ & (\ new'(n,r).\overline{new}\langle n, r\rangle + \Sigma_{f\in\mathbb{F}} new'_f(\widetilde{i}, r).\overline{new_f}\langle\widetilde{i}, r\rangle \\ & + input'(i,r).\overline{input}\langle i, r\rangle + output'(i,j,r).\overline{output}\langle i, j, r\rangle \\ & + \Sigma_{E\in\mathbb{E}} apply'_E(\widetilde{i}, r).\overline{apply_E}\langle\widetilde{i}, r\rangle)\end{aligned}$$

4 Full Abstraction

The relevance of may-testing equivalence for analysing security protocols has been stated elsewhere (see eg [AG99]). For this reason, we keep this section relatively technical apart from stating the rough idea of the long and complex proof of the full abstraction result.

Definition 4.1 1. An *observer* is an process that may use a distinguished name \$. An action on channel \$ is a *success signal*.

2. A *test* is a parallel composition $P \mid O$ of an process P and an observer O.

3. An process P *may pass* a test $P \mid O$ if some sequence of τ-steps of $P \mid O$ has a state in which O signals success. Formally, we denote this property by P may O.

4. Any two processes P and Q are *may-testing equivalent* if P may O if and only if Q may O for every observer O – in other words, the tests P and Q may pass are the same. We denote this property by $P \eqsim_{\text{may}} Q$.

Theorem 4.2 *Given any πT-processes P and Q, $P \eqsim_{\text{may}} Q$ iff $[\![P]\!] \eqsim_{\text{may}} [\![Q]\!]$.*

Statement. Given any πT-processes P and Q, $P \eqsim_{\text{may}} Q$ iff $[\![P]\!] \eqsim_{\text{may}} [\![Q]\!]$.

4.1 Preservation of May Tests

The most difficult step in proving Theorem 4.2 consists of proving a result that is analogous to the one proved in [BPV04], namely that $(\!|\_|\!)$ preserves may-tests. We formulate this result with respect to a modified notion of may-testing. This adaptation is necessary since the preservation of may-tests has to be considered in the context of integrity management. The problem with the ordinary notion would be that this context is meant to consume all visible actions of encoded processes and observers, which would include any success signal by the encoded observer.

Definition 4.3 An encoded πT-process $(\!|[\![P]\!]|\!)$ *may pass* the test posed by an encoded πT-observer $(\!|[\![O]\!]|\!)$ *as mediated by integrity management* if some sequence of τ-steps of $(\!|[\![P]\!]|\!) \mid \mathbf{M} \mid (\!|[\![O]\!]|\!)$ has a state in which some encoded success signal (of the form $(\!|[\![\$(\_).\mathbf{0}]\!]|\!)$ inside the observer component) becomes un-guarded. We denote this property by $(\!|[\![P]\!]|\!)$ $\text{may}_{\mathbf{M}}$ $(\!|[\![O]\!]|\!)$.

Theorem 4.4 *(Preservation of May-Tests) Given any πT process P and any πT observer O, P* may *O if and only if $(\!|[\![P]\!]|\!)$ $\text{may}_{\mathbf{M}}$ $(\!|[\![O]\!]|\!)$.*

This result is analogous to earlier work [BPV04] but, at the same time, it is by far more difficult. The reason is that, unlike [BPV04], the global context in the form of the integrity manager has to be taken into account. The solution consists of working with a set-theoretical abstraction of the integrity manager that allows us to consider only factions of it as we exploit the syntax-directed way in which the pure encoding $(\!|\_|\!)$ is defined. In this way we are able to re-instantiate the concept of an ancestor relation introduced in [BPV04] to prove both a forward and a backward operational correspondence between unencoded and encoded process-observer couplings. The preservation of may tests is then an easy corollary once we also have an operational correspondence between abstract and concrete integrity management. The detailed proof can be found in the appendix.

4.2 Trace Characterisation of πT-May-Testing Equivalence

The second step in proving Theorem 4.2 consists of characterising may-testing equivalence of πT-processes in terms of traces of their encodings under $(\!|\_|\!)$.

A *trace* θ is a finite sequence of actions. Any given process P *has* some trace θ if (a) θ is empty or (b) there is a (not necessarily maximal) sequence $P \xrightarrow{\alpha_1} P_1 \ldots \xrightarrow{\alpha_k} P_k$, $k \geq 1$, such that $\theta = \alpha_1 \ldots \alpha_k$. Moreover, given any non-empty trace $\theta = \alpha_1 \ldots \alpha_k$, and any name n, *n occurs freely at position i* in θ, $1 \leq i \leq k$, if (i) $n \in \text{fn}[\alpha_i]$ and (ii) $n \notin \text{bn}[\alpha_j]$ for all $j \in \{1, \ldots, i-1\}$, where $\text{fn}[\alpha]$ is given by $\text{fn}[\alpha] = \text{nm}[\alpha] \setminus \text{bn}[\alpha]$ for every α.

Definition 4.5 A trace θ is *faithful (to integrity management)* if every free occurrence of any name from $\mathrm{K}_{\mathbf{M}}$ in θ is an occurrence in input subject position.

We denote the faithful traces of any given π-calculus process P by $\mathrm{ftr}(P)$. Given any $\pi\mathrm{T}$ process P, we write $\mathrm{ftr}(P)$ for $\mathrm{ftr}(\llparenthesis P \rrparenthesis \,|\, \mathbf{M})$.

Given any trace θ, we denote by $\theta^{\not\tau}$ the trace that results from removing all τ-actions from θ; given any set of traces Θ, we denote by $\Theta^{\not\tau}$ the set given by $\Theta^{\not\tau} = \{\theta^{\not\tau} \,|\, \theta \in \Theta\}$. We call $\theta^{\not\tau}$ the *τ-reduced* trace of θ.

Theorem 4.6 *(π-calculus Trace Characterisation of πT-May-Testing Equivalence) For any two πT processes P and Q, $P \eqsim_{\mathrm{may}} Q$ if and only if $\mathrm{ftr}^{\not\tau}(P) = \mathrm{ftr}^{\not\tau}(Q)$.*

Proof. Let P and Q be πT-processes. "$\Leftarrow$": Suppose $\mathrm{ftr}^{\not\tau}(P) = \mathrm{ftr}^{\not\tau}(Q)$. Further, let O be a πT-observer and suppose, for a contradiction, P may O while Q $\not{\text{may}}$ O. Then, by Theorem 4.4, $\llparenthesis P \rrparenthesis$ $\mathrm{may}_{\mathbf{M}}$ $\llparenthesis O \rrparenthesis$ while $\llparenthesis Q \rrparenthesis$ $\not{\mathrm{may}}_{\mathbf{M}}$ $\llparenthesis O \rrparenthesis$. Then there must be a τ-reduced trace θ of $\llparenthesis P \rrparenthesis \,|\, \mathbf{M}$ such that $\llparenthesis O \rrparenthesis$ has a complementary τ-reduced trace whereby it enters a success state. Because $\llparenthesis O \rrparenthesis$ is an encoded πT-process, θ is faithful to integrity management, whence $\mathrm{ftr}^{\not\tau}(P) = \mathrm{ftr}^{\not\tau}(Q)$ entails that θ is also a τ-reduced trace of $\llparenthesis Q \rrparenthesis \,|\, \mathbf{M}$, so $\llparenthesis Q \rrparenthesis \,|\, \mathbf{M}$ may $\llparenthesis O \rrparenthesis \not\downarrow$. By symmetry, there is a also contradiction if one assumes Q may O while not P may O. Hence, $\mathrm{ftr}(P) = \mathrm{ftr}(Q)$ entails $P \eqsim_{\mathrm{may}} Q$.

"$\Rightarrow$": Suppose $P \eqsim_{\mathrm{may}} Q$ and, for a contradiction, $\theta \in \mathrm{ftr}^{\not\tau}(P)$ while $\theta \notin \mathrm{ftr}^{\not\tau}(Q)$. Without loss of generality, we may assume that θ satisfies the following well-formedness properties:

i. There is a uniquely-attributable response output for every input.

ii. Corresponding inputs and response outputs are adjacent to each other.

The validity of these properties can be established by exploiting that any outside observer will interact with $\llparenthesis P \rrparenthesis \,|\, \mathbf{M}$ only via $\mathbf{M}$. As for (i), inputs that are malformed in a way such that $\mathbf{M}$ will never respond can simply be discarded from θ and the ensuing trace is still in $\mathrm{ftr}^{\not\tau}(P)$. Also, channels that are used more than once for responding can be split as many times as necessary by supplying fresh names in the corresponding inputs. As for (ii), this property can be achieved by pulling response actions forward as necessary.

The next step consists of transforming θ into a sequential πT-process $\pi\mathrm{T}(\theta)$. This operation is recursively defined by the clauses displayed in Table 2 on the following page. The substitution operator on any trace $\alpha_1 \ldots \alpha_k$ is defined on the free names of the α_i's, $1 \leq i \leq k$, where alpha conversion is invoked to avoid any capture due to any name bindings in $\alpha_1 \ldots \alpha_i$. By induction on the length of θ, using that $\theta \in \mathrm{ftr}^{\not\tau}(P)$, it holds that P may $\pi\mathrm{T}(\theta)$. Hence, $P \eqsim_{\mathrm{may}} Q$ entails Q may $\pi\mathrm{T}(\theta)$. Hence, again by induction on the length of θ, it must hold that $\theta \in \mathrm{ftr}^{\not\tau}(Q)\not\downarrow$.

By symmetry, there is also a contradiction if one assumes $\theta \in \mathrm{ftr}^{\not\tau}(Q)$ while $\theta \notin \mathrm{ftr}^{\not\tau}(P)$. Hence, $P \eqsim_{\mathrm{may}} Q$ entails $\mathrm{ftr}^{\not\tau}(P) = \mathrm{ftr}^{\not\tau}(Q)$. □

4.3 Finalising the Proof of Theorem 4.2

Proof of Theorem 4.2. As a direct consequence of Theorem 4.6. Suppose P and Q are πT-processes. The key is that $\llparenthesis P \rrparenthesis \,|\, \mathbf{M}$ and $\llparenthesis Q \rrparenthesis \,|\, \mathbf{M}$, being embedded in the context $(\nu\, \mathrm{K}_{\mathbf{M}})([\,] \,|\, \mathbf{F})$, will experience every trace of interaction with any outside π-calculus observer as complementary to a faithful trace of their own. □

Table 2 Defining clauses for πT-operator.

$$\pi\mathrm{T}(new(n,r)\ \ (\nu\,\emptyset)\,\overline{r}\langle i\rangle\ \ \theta') \ = \ \pi\mathrm{T}(\theta'\{i := n\})$$
$$\pi\mathrm{T}(new(n,r)\ \ (\nu\,\{i\})\,\overline{r}\langle i\rangle\ \ \theta') \ = \ (\nu\,n)\,\pi\mathrm{T}(\theta'\{i := n\})$$
$$\pi\mathrm{T}(new_f(V_1,\ldots,V_{\mathrm{ar}[f]},r)\ \ (\nu\,N)\,\overline{r}\langle i\rangle\ \ \theta') \ = \ \pi\mathrm{T}(\theta'\{i := f(V_1,\ldots,V_{\mathrm{ar}[f]})\})$$
$$\pi\mathrm{T}(apply_E(V_1,\ldots,V_{\mathrm{ar}[E]},r)\ \ (\nu\,\emptyset)\,\overline{r}\langle i\rangle\ \ \theta')$$
$$= \ \textbf{let}\ x = d(V_1,\ldots,V_{\mathrm{ar}[E]})\ \textbf{in}\ \pi\mathrm{T}(\theta'\{i := x\})$$
where d is the deconstructor of E and x is fresh
$$\pi\mathrm{T}(input(V,r)\ \ \overline{r}\langle i\rangle\ \ \theta') \ = \ V(x).\pi\mathrm{T}(\theta'\{i := x\}),\ x \text{ fresh}$$
$$\pi\mathrm{T}(output(V,W,r)\ \ \overline{r}\langle\rangle\ \ \theta') \ = \ \overline{V}\langle W\rangle.\pi\mathrm{T}(\theta')$$
$$\pi\mathrm{T}(\epsilon) \ = \ \$(\_).\mathbf{0}$$

It might be instructive to note that at some places a value expression designated by V or W appears where in the original trace there is only an ID. That is to take account of substitutions that might have been effected at earlier stages in the transformation process.

Proof. (Idea) The proof has two main steps:

1. This step consists of proving that may-tests are preserved, that is to say, proving that the encodings of two πA processes satisfy the same tests if the tests are encodings of πA tests.

2. This step mainly consists of using the result from Step 1 in obtaining a π-calculus trace characterisation of πT-may-testing equivalence. This property is reminiscent of trace characterisations of may-testing equivalence in other settings (see eg [DH84]). All what is left is then to take the firewall into account to obtain the final full-abstraction result.

The full proof can be found in [BPV05][1]. □

5 Related Work

Arbitrary equations between data terms are the most crucial feature of the applied π-calculus that is lacking from πT. Also, we require that deconstructors are only used in let-expressions. πT is therefore in between the applied π-calculus and the spi calculus in that it resembles Borgström, Briais and Nestmann's parameterised spi calculus [BBN04]. These simplifications entail a loss of expressiveness when describing security protocols. The advantage is that we can still give a straightforward Structural Operational Semantics (SOS) for agents without having to first define a complex evaluation semantics for value terms. This simplifies our proofs to a great extent.

Full abstraction in the kind of encoding considered here is difficult to achieve. Milner's encoding of the λ-calculus [Mil92] is not fully abstract. Sangiorgi [San93] demonstrates full abstraction for an encoding from the higher order (HO) π-calculus. The main difference is that in HOπ the language does not contain an equality test for HO values. In πT there is such a test for data terms. This means that in the encoding the interior of a term must, so to speak, be open for inspection, and this makes our encoding and proof very different.

The applied π-calculus, the spi calculus [AG99], Burrows, Abadi and Needham's logic of authentication [BAN89], which is nowadays known as BAN logic, and a number of other

modelling and analysis techniques for security protocols rest on the Dolev-Yao assumption [DY83]. This means that the underlying crypto-system is considered unbreakable, so that the protocol logic on top of that is the only concern. A central aspect of all of these lines of work is that an analysis makes it necessary to explicitly represent the knowledge that an observer of a supposedly secure system can accumulate over time. This situation is similar in mobile process approaches: Observer knowledge has there been incorporated in notions of bisimulation and testing within the framework of the spi calculus; Borgström and Nestmann give a good overview of bisimulation for the spi calculus in [BN03]; Boreale, De Nicola and Pugliese have, besides treating bisimulation, shown how to take account of may-testing for the spi calculus in terms of knowledge-enriched traces [BDP02]. In contrast to all of them our characterisation seems to be unique in that all observer knowledge is internalised. That is to say, it appears on the calculus level, not on the meta-level. In this sense it could be considered much closer to traditional process algebra methodology than the above-mentioned approaches.

6 Conclusion

We have presented a new encoding from the value-enriched mobile process calculus πT into the polyadic π-calculus. The cornerstone of the translation is an integrity manager that acts as a clearing house for all operations that involve translated values. It is protected by a firewall so that it cannot be impersonated by a hostile environment. Similar techniques were used in [AFG98]. This encoding solves the long open full abstraction problem for πT-like calculi.

Our encoding is not compositional since it has the integrity manager and the firewall as a global context. While compositionality has been put forward as a criterion for whether an encoding from one calculus to the other is good [Pal97], it can be argued (see [Nes00]) that these criteria are too strong for practical purposes, and that by allowing a top-level context (but keeping the inner encoding compositional), many practically or theoretically motivated encodings turn out to be "good".

One way to interpret our result is that the π-calculus can provide as much "security" in the sense of protected data values as the πT-calculus and its instantiations such as the spi calculus. The means to achieve this are an integrity manager and a firewall. Thus we could claim to have proved that integrity management plus firewalling is a viable security philosophy. Modulo possible termonological differences this belief may actually be common; our contribution is to prove the correctness of a formal statement of it.

References

[AF01] M. Abadi and C. Fournet. Mobile Values, New Names, and Secure Communication. In *Principles of Programming Languages*, pages 104–115. ACM, 2001. POPL symposium proceedings.

[AFG98] A. Abadi, C. Fournet, and G. Georges. Secure Implementation of Channel Abstractions. In *Logic in Computer Science*, pages 105–116. IEEE, 1998. LICS symposium proceedings.

[AG99] M. Abadi and A. Gordon. A Calculus for Cryptographic Protocols: The Spi Calculus. *Information and Computation*, 148(1):1–70, 1999.

[BAN89] A. Burrows, M. Abadi, and R. Needham. A logic of authentication. *Proceedings of the Royal Society of London A*, 426:233–271, 1989.

[BBN04] J. Borgström, S. Briais, and U. Nestmann. Symbolic Bisimulation in the Spi Calculus. In *Concurrency Theory*, LNCS 3170, pages 161–176, 2004. Concur conference proceedings.

[BDP02] M. Boreale, R. De Nicola, and R. Pugliese. Proof Techniques for Cryptographic Processes. *SIAM Journal on Computing*, 31(3):947–986, 2002.

[BN03] J. Borgström and U. Nestmann. On Bisimulations for the Spi Calculus. Technical Report IC/2003/34, EPFL I&C, 2003.

[BPV04] M. Baldamus, J. Parrow, and B. Victor. Translating Spi Calculus to π-Calculus Preserving May-Tests. In *Logic in Computer Science*, pages 22–31. IEEE, 2004. LICS symposium proceedings.

[BPV05] Michael Baldamus, Joachim Parrow, and Björn Victor. A Fully Abstract Encoding of the Applied π-Calculus. Technical Report 2005-004, Department of Information Technology, Uppsala University, February 2005.

[DH84] R. De Nicola and M. Hennessy. Testing Equivalences for Processes. *Theoretical Computer Science*, 34:83–133, 1984.

[DY83] D. Dolev and A. Yao. On the Security of Public-Key Protocols. *IEEE Transactions on Information Technology*, 29(2):198–208, 1983.

[Mil92] Robin Milner. Functions as Processes. *Journal of Mathematical Structures in Computer Science*, 2(2):119–141, 1992.

[MPW92] R. Milner, J. Parrow, and D. Walker. A Calculus of Mobile Processes, Parts I/II. *Information and Computation*, 100:1–77, 1992.

[Nes00] Uwe Nestmann. What Is a 'Good' Encoding of Guarded Choice? *Information and Computation*, 156:287–319, 2000.

[Pal97] C. Palamidessi. Comparing the Expressive Power of the Synchronous and the Asynchronous π-Calculus. In *Principles of Programming Languages*. ACM, 1997.

[San93] D. Sangiorgi. From π-Calculus to Higher-Order π-Calculus – and Back. In *Theory and Practice of Software Development*, LNCS 668, pages 151–161. Springer, 1993.

[SW03] D. Sangiorgi and D. Walker. *The π-Calculus: A Theory of Mobile Processes*. Cambridge University Press, 2003.

Proving that May-Tests are Preserved

Statement. Given any πT process P and any πT observer O, P may O if and only if $(\!|P|\!)\ \mathrm{may}_{\mathbf{M}}\ (\!|O|\!)$.

Proof. This property is an immediate corollary of Theorem A.7 and Lemma A.14 below. □

A.1 Preservation of May-Tests with Respect to Abstract Integrity Management

A.1.1 Preliminaries

A.1.1.1 π-Calculus Preliminaries

Definition A.1 *Strong (early) bisimilarity* on π-calculus processes is the largest relation $\sim$ such that $P \sim Q$ implies:

i. Whenever $P \xrightarrow{\alpha} P'$ where $\mathrm{bn}[\alpha] \cap \mathrm{fn}[Q] = \emptyset$, then $Q \xrightarrow{\alpha} \sim^{-1} P'$.

ii. Whenever $Q \xrightarrow{\alpha} Q'$ where $\mathrm{bn}[\alpha] \cap \mathrm{fn}[P] = \emptyset$, then $P' \xrightarrow{\alpha} \sim Q'$.

A classical congruence result concerning strong bisimilarity is the following one:

Lemma A.2 *Strong bisimilarity over the polyadic π-calculus is a non-input congruence (cf. [SW03]).*

Proof. We can prove this property by adapting any already existing proof of the fact that strong bisimilarity is a non-input congruence. The adaptation is necessary because of the slightly non-standard semantics of the **if-then-else**-operator as compared to π-calculus matching and mismatching. The technical details are omitted here since they are entirely straightforward. □

Lemma A.3 *For any π-calculus process P, $!\,P \sim P \mid !\,P$.*

Proof. We can prove this property in the same vein as Lemma A.2. □

A.1.1.2 Set-Theoretical Preliminaries We understand a *mapping* f set-theoretically, that is to say, as a set of argument/values pairs that is *image-unique* in the sense that $\langle x, y_1\rangle \in f$ and $\langle x, y_2\rangle \in f$ entails $y_1 = y_2$. This stipulation allows us to apply set-theoretic notions to mappings. A *finite mapping*, for instance, is a mapping where the number of argument/value pairs is finite. Even the set-theoretic union of any number of mappings is defined as long as image-uniqueness is preserved. We write $x\colon y$ for any argument/value pair $\langle x, y\rangle$. Also, given any mapping f, the *domain* $\mathrm{dom}[f]$ or *codomain* $\mathrm{cod}[f]$ of f is given by $\mathrm{dom}[f] = \{x \mid \text{there is a } y \text{ such that } x\colon y \in f\}$ or $\mathrm{cod}[f] = \{y \mid \text{there is an } x \text{ such that } x\colon y \in f\}$, respectively. We write $f\!\downarrow x$ or $f\!\uparrow y$ for the property that $x \in \mathrm{dom}[f]$ or $y \in \mathrm{cod}[f]$, respectively. Moreover, we write $f - x$ or $f + x\colon y$ for the mapping given by $f - x = \{x'\colon y \in f \mid x' \neq x\}$ or $f + x\colon y = f \cup \{x\colon y\}$, respectively. Any use of the $f + x\colon y$ notation carries the implicit side condition with it that $f \not\downarrow x$ or $f[x] = y$.

A.1.2 Abstract Integrity Management

We introduce an operator id on values that yields a unique ID $\mathrm{id}[V]$ for any value V. We assume $\mathrm{id}[V] \neq V$ for any πT-value V.

An *abstract identity record* is a pair of one of the following forms:

1. $\mathrm{id}[n] : \_$, where $\_$ is an arbitrary but fixed don't-care value
2. $\mathrm{id}[f(V_1, \ldots, V_{\mathrm{ar}[f]})] : \langle f, \mathrm{id}[V_1], \ldots, \mathrm{id}[V_{\mathrm{ar}[f]}]\rangle$

An *abstract integrity manager thread state* is a pair of one of the following forms:

1. $r : new(n)$
2. $r : new_f(i_1, \ldots, i_{\mathrm{ar}[f]})$
3. $r : input(i)$
4. $r : output(i, j)$
5. $r : apply_E(i_1, \ldots, i_{\mathrm{ar}[E]})$
6. $r : \overline{r}\langle\rangle$, $r : \overline{r}\langle \mathrm{id}[V]\rangle$

An *abstract integrity manager state faction* is a tuple of the form $\langle \delta, \phi \rangle$ where δ and ϕ are finite mappings consisting of abstract identity records or abstract integrity manager thread states, respectively. The tuple must be *well-formed* in the following sense:

i. The ID of each sub-value of any constructor term value must be allocated whenever the complex value's ID in and of itself has been allocated: $\delta{\downarrow}\, \mathrm{id}[f(V_1, \ldots, V_{\mathrm{ar}[f]})]$ implies $\delta{\downarrow}\, \mathrm{id}[V_i]$ $(i = 1, \ldots, \mathrm{ar}[f])$.

ii. Every value that is to be returned must have been allocated: $\phi{\uparrow}\, \overline{r}\langle \mathrm{id}[V]\rangle$ implies $\delta{\downarrow}\, \mathrm{id}[V]$.

We denote abstract integrity manager state factions by Σ, Ω; the set $\mathrm{nm}[\langle \delta, \phi \rangle]$ is given by

$$\begin{aligned}
\mathrm{nm}[\langle \delta, \phi \rangle] = \quad & \mathrm{dom}[\delta] \\
& \cup\, \mathrm{dom}[\phi] \\
& \cup\, \{n \mid \phi{\uparrow}\, new(n)\} \\
& \cup\, \{i_1, \ldots, i_{\mathrm{ar}[f]} \mid \phi{\uparrow}\, new_f(i_1, \ldots, i_{\mathrm{ar}[f]}) \text{ for some } f\} \\
& \cup\, \{i \mid \phi{\uparrow}\, input(i)\} \\
& \cup\, \{i, j \mid \phi{\uparrow}\, input(i, j)\} \\
& \cup\, \{i_1, \ldots, i_{\mathrm{ar}[E]} \mid \phi{\uparrow}\, apply_E(i_1, \ldots, i_{\mathrm{ar}[E]}) \text{ for some } E\} \\
& \cup\, \{\mathrm{id}[V] \mid \phi{\uparrow}\, \overline{r}\langle \mathrm{id}[V]\rangle \text{ for some } r\}
\end{aligned}$$

Transitions over abstract integrity manager state factions are given by the clauses listed in Table 3 on page 35.

Lemma A.4 *The transitions defined in Table 3 preserve well-formed abstract integrity manager state factions.*

Proof. This proof is straightforward. Hence it may be left implicit. □

An *abstract encoding state faction* is a triple of the form $\langle N, P, \Sigma \rangle$ where P is a π-calculus process. We denote such triples by $P \overset{\nu\, N}{\bowtie} \Sigma$, or by $P \bowtie \Sigma$ if N is empty. The names in N are understood to be bound as shared private names of P and Σ. We do not distinguish abstract encoding state factions that differ only up to alpha-conversion of bound names. An abstract encoding state faction must be *well-formed* in the following sense:

i. $\langle \delta, \phi \rangle$ is well-formed.

ii. All return channels in the abstract integrity manager are privately shared: $\mathrm{dom}[\phi] \subseteq N$.

Its important to notice that N does not contain any private value IDs. This information is already implicit in N. The set $\mathrm{fn}[P \overset{\nu N}{\bowtie} \Sigma]$ is given by $\mathrm{fn}[P \overset{\nu N}{\bowtie} \Sigma] = (\mathrm{fn}[P] \cup \mathrm{nm}[\Sigma]) \setminus N$.

Transitions of abstract encoding state factions are given by the clauses displayed in Table 4 on page 36.

Lemma A.5 *The transitions defined in Table 4 preserve well-formed abstract encoding state factions.*

Proof. Just like the proof of Lemma A.4, this proof is straightforward. Hence it may also be left implicit. □

Definition A.6 An encoded πT-process $(\!|P|\!]$ *may pass* the test posed by an encoded πT-observer $(\!|O|\!)$ *as mediated by abstract integrity management* if some sequence of τ-steps starting in $(\!|P|\!]\,|\,(\!|O|\!) \bowtie \langle \emptyset, \emptyset \rangle$ has a state in which some encoded success signal (of the form $(\!|\$(\_).\mathbf{0}|\!)$ inside the observer component) becomes un-guarded. We denote this property by $(\!|P|\!] \; \mathrm{may}_{\bowtie} \; (\!|O|\!)$.

Theorem A.7 *Given any πT process P and any πT observer O, P* may *O if and only if $(\!|P|\!] \; \mathrm{may}_{\bowtie} \; (\!|O|\!)$.*

The rest of Subsection A.1 is concerned with proving this theorem.

A.1.3 Basic Properties of Abstract Integrity Management

Definition A.8 *Strong bisimilarity* on abstract encoding state factions is the largest relation $\sim$ such that $P \overset{\nu N}{\bowtie} \Sigma \sim Q \overset{\nu M}{\bowtie} \Omega$ implies:

i. Whenever $P \overset{\nu N}{\bowtie} \Sigma \to P' \overset{\nu N'}{\bowtie} \Sigma'$, then $Q \overset{\nu M}{\bowtie} \Omega \to\!\sim^{-1} P' \overset{\nu N'}{\bowtie} \Sigma'$.

ii. Whenever $Q \overset{\nu N}{\bowtie} \Omega \to Q' \overset{\nu M}{\bowtie} \Omega'$, then $P \overset{\nu N}{\bowtie} \Sigma \to\!\sim Q' \overset{\nu M'}{\bowtie} \Omega'$.

Lemma A.9

1. *Whenever $P \sim Q$, then $P \overset{\nu N}{\bowtie} \Sigma \sim Q \overset{\nu N}{\bowtie} \Sigma$ for all N and all Σ.*
2. $P \overset{\nu N}{\bowtie} \Sigma \sim P \overset{\nu M}{\bowtie} \Sigma$

Proof. We can prove this property by showing that the binary relation $\{\langle P \overset{\nu N}{\bowtie} \Sigma, Q \overset{\nu N}{\bowtie} \Sigma\rangle \mid P \sim Q\}$ or $\{\langle P \overset{\nu N}{\bowtie} \Sigma, P \overset{\nu M}{\bowtie} \Sigma\rangle \mid \top\}$, respectively, is a strong bisimulation on abstract encoding state factions. □

A.1.4 Abstract Operational Correspondence

A.1.4.1 Ancestor Relation The core of the proof of Theorem A.7 consists of a relation over unencoded processes and derivatives of encoded processes, where the derivatives are coupled with abstract encoding state factions. Following [BPV04], we call this relation the *ancestor relation*. The crucial difference to the ancestor relation from [BPV04] consists of the fact that this relation takes the abstract integrity management context into account. Its defining clauses can be found in Table 5 on page 37 and Table 6 on page 38. Some notation that is used in these tables and later on is as follows:

1. $\blacktriangleleft$ is the ancestor relation; $\blacktriangleright$ is its inverse.

2. - $\delta \oplus n = \delta + \mathrm{id}[n]\colon \_$
   - $\delta \oplus f(V_1, \ldots, V_{\mathrm{ar}[f]})$
     $= \delta \oplus V_1 \oplus \cdots \oplus V_{\mathrm{ar}[f]} + \mathrm{id}[f(V_1, \ldots, V_{\mathrm{ar}[f]})]\colon \langle f, \mathrm{id}[V_1], \ldots, \mathrm{id}[V_{\mathrm{ar}[f]}]\rangle$
   - $\delta \oplus (V_1, \ldots, V_k) = \delta \oplus V_1 \oplus \cdots \oplus V_k$

3. $\Rightarrow$ is the reflexive and transitive closure of $\rightarrow$ (when we are dealing with a reduction relation) or $\xrightarrow{\tau}$ (when we are dealing with a labelled transition relation).

4. $P \xrightarrow{\widehat{\alpha}} P'$ stands for $P \xrightarrow{\alpha} P'$, given that α is visible, and for the property: (a) $P \xrightarrow{\tau} P'$ or (b) $P = P'$, given that $\alpha = \tau$.

5. $P > \phi \Rightarrow P'$, where ϕ is some predicate stands for the property: $P \Rightarrow P'$ under the assumption of ϕ.

Lemma A.10 *The derivations defined in Tables 5 and 6 preserve well-formed abstract encoding state factions.*

Proof. Similarly to the proofs of Lemmas A.5 and A.4, this proof is straightforward. Hence it may also be left implicit. □

Lemma A.11 $P\{\widetilde{x} := \widetilde{V}\} \blacktriangleleft (\!|P|\!)\{\widetilde{x} := \mathrm{id}[\widetilde{V}]\} \bowtie \langle \emptyset \oplus \widetilde{V}, \emptyset\rangle$

Proof. We prove this property by straightforward structural induction on P. □

A.1.4.2 Forward Abstract Operational Correspondence

Theorem A.12

i. Whenever $Q \overset{\nu M}{\bowtie} \Sigma \blacktriangleright P \xrightarrow{n(W)} P'$ *where* $n \notin M$ *and* $M \cap \mathrm{nm}[W] = \emptyset$, *then* $Q \overset{\nu M}{\bowtie} \Sigma \Rightarrow Q' \overset{\nu M'}{\bowtie} \langle \epsilon', \psi' + r : \mathit{input}(\mathrm{id}[n])\rangle$ *where* $Q' \overset{\nu M'}{\bowtie} \langle \epsilon', \psi' - r + r : \overline{r}\langle \mathrm{id}[W]\rangle\rangle \rightarrow\blacktriangleright P'$.

ii. Whenever $Q \overset{\nu M}{\bowtie} \Sigma \blacktriangleright P \xrightarrow{(\nu K)\,\overline{n}\langle W\rangle} P'$ *where* $n \notin M$, *then* $Q \overset{\nu M}{\bowtie} \Sigma \Rightarrow Q' \overset{\nu M'}{\bowtie} \langle \epsilon', \psi' + r : \mathit{output}(\mathrm{id}[n], \mathrm{id}[W])\rangle$ *where* $Q' \overset{\nu M'}{\bowtie} \langle \epsilon', \psi' - r + r : \overline{r}\langle\rangle\rangle \rightarrow\blacktriangleright P'$ *and* $K \subseteq M'$.

iii. Whenever $Q \overset{\nu M}{\bowtie} \Sigma \blacktriangleright P \xrightarrow{\tau} P'$, *then* $Q \overset{\nu M}{\bowtie} \Sigma \Rightarrow\blacktriangleright P'$.

Proof. As it is usually the case with operational correspondences where encodings are involved, the forward direction is a complementary reduct of the backward direction, the reason being that we do not have to take interleavings of different interpretation threads into account. For this reason, we leave the proof implicit. □

A.1.4.3 Backward Abstract Operational Correspondence

Theorem A.13

i. Whenever $P \blacktriangleleft Q \overset{\nu M}{\bowtie} \langle \epsilon, \psi + r : \mathit{input}(\mathrm{id}[n]) \rangle$, *where* $n \notin M$, *then* $P \xrightarrow{n(W)} \blacktriangleleft Q \overset{\nu M}{\bowtie} \langle \epsilon \oplus W, \psi - r + r : \overline{r}\langle \mathrm{id}[W] \rangle \rangle$ *for all* W *where* $M \cap \mathrm{nm}[W] = \emptyset$.

ii. Whenever $P \blacktriangleleft Q \overset{\nu M}{\bowtie} \langle \epsilon, \psi + r : \mathit{output}(\mathrm{id}[n], \mathrm{id}[W]) \rangle$, *where* $n \notin M$, *then* $P \xrightarrow{(\nu K)\,\overline{n}\langle W \rangle} \blacktriangleleft Q \overset{\nu M}{\bowtie} \langle \epsilon, \psi - r + r : \overline{r}\langle\langle\rangle\rangle \rangle$ *for some* K *where* $K \subseteq M$.

iii. Whenever $P \blacktriangleleft Q \overset{\nu M}{\bowtie} \Omega \to Q' \overset{\nu M'}{\bowtie} \Omega'$, *then* $P \xrightarrow{\widehat{\tau}} \sim \blacktriangleleft Q' \overset{\nu M'}{\bowtie} \Omega'$.

Proof. We prove this theorem by induction on the lexicographical order on the measure

$$(\text{size of } P, \text{ length of the inference used to establish } P \blacktriangleleft Q \overset{\nu M}{\bowtie} \Omega).$$

In carrying out the induction we consider (i)-(iii) simultaneously: The individual cases are as shown below. In all cases let $\Omega = \langle \epsilon, \psi \rangle$ and let $P'[W]$ or P' be the sought-after derivative of P. In case (iii) let $\Omega' = \langle \epsilon', \psi' \rangle$. We organise the presentation according to the last clause used in the inference used to establish $P \blacktriangleleft Q \overset{\nu M}{\bowtie} \Omega$.

(◀-**0**) This situation is actually *impossible* as (i)-(iii) carry additional premises with them that can not be satisfied if $P \blacktriangleleft Q \overset{\nu M}{\bowtie} \Omega$ has been inferred with (◀-**0**). Henceforth we designate all similar "cases" as impossible.

(◀-IN) In this case P is of the form $(T(y).P_1)\{\widetilde{x} := \widetilde{V}\}$ where $y \notin \widetilde{x}$ and r is fresh. We proceed by considering (i)-(iii) separately.

i. In this case $Q \overset{\nu M}{\bowtie} \Omega$ is bisimilar to $r(y).(\!|P_1|\!)\{\widetilde{x} := \mathrm{id}[\widetilde{V}]\} \overset{\nu\{r\}}{\bowtie} \langle \emptyset \oplus n, \emptyset + r : \mathit{input}(\mathrm{id}[n]) \rangle$ since (a) $r : \mathit{input}(\mathrm{id}[n])$ can have been inserted in the abstract integrity manager state faction only through the second-to-last step executed in interpreting the prefix $T(y)\{\widetilde{x} := \widetilde{V}\}$, (b) the residual of $(\!|T|\!)_r\{\widetilde{x} := \mathrm{id}[\widetilde{V}]\}$ has become bisimilar to **0** in this state of the interpretation (cf. Lemma A.9) and (c), as another property that holds in this state of the interpretation, any names in M that are different from r are not used in the interpretation anymore (ibid.). The bisimilarity entails $T\{\widetilde{x} := \widetilde{V}\} = n$ for some n, so

$$(T(y).P_1)\{\widetilde{x} := \widetilde{V}\} \xrightarrow{n(W)} P_1\{\widetilde{x}, y := \widetilde{V}, W\}$$

for all W. Further, by (◀-IN′) and (◀-EXT):

$$P_1\{\widetilde{x}, y := \widetilde{V}, W\} \blacktriangleleft r(y).(\!|P_1|\!)\{\widetilde{x} := \mathrm{id}[\widetilde{V}]\} \overset{\nu\{r\}}{\bowtie} \Big\langle \emptyset \oplus \widetilde{V} \oplus n \oplus W, \emptyset + r : \overline{r}\langle \mathrm{id}[W] \rangle \Big\rangle.$$

As W is arbitrary, this establishes the required conclusion if we set $P' = P_1\{\widetilde{x}, y := \widetilde{V}, W\}$, $M = \{r\}$, $\epsilon = \emptyset \oplus n$ and $\psi = \emptyset$.

ii. This case is impossible since abstract integrity manager thread states of the form $r : \mathit{output}(i, j)$ are not generated whenever any input prefix is interpreted.

iii. In this case $Q \overset{\nu M}{\bowtie} \Omega$ has a transition of the form $Q \overset{\nu M}{\bowtie} \Omega \to Q' \overset{\nu M'}{\bowtie} \Omega'$ where this transition belongs to interpreting the prefix $T(y)\{\widetilde{x} := \widetilde{V}\}$. Hence we can apply ($\blacktriangleleft$-IN) again to infer $(T(y).P_1)\{\widetilde{x} := \widetilde{V}\} \blacktriangleleft Q' \overset{\nu M'}{\bowtie} \Omega'$. This establishes the required conclusion if we set $P' = P$.

($\blacktriangleleft$-IN$'$) In this case $P \blacktriangleleft Q \overset{\nu M}{\bowtie} \Omega$ is given as

$$P_1\{\widetilde{x}, y := \widetilde{V}, W\} \blacktriangleleft r(y).(\!|P_1|\!)\{\widetilde{x} := \mathrm{id}[\widetilde{V}]\} \overset{\nu\{r\}}{\bowtie} \Big\langle \emptyset \oplus \widetilde{V} \oplus W, \emptyset + r : \overline{r}\langle \mathrm{id}[W]\rangle \Big\rangle$$

where $y \notin \widetilde{x}$ and r is fresh. Any sub-"cases" triggered by (i) and (ii) are impossible, so it remains to consider (iii). In this case the only possible transition by $Q \overset{\nu M}{\bowtie} \Omega$ is

$$r(y).(\!|P_1|\!)\{\widetilde{x} := \mathrm{id}[\widetilde{V}]\} \overset{\nu\{r\}}{\bowtie} \Big\langle \emptyset \oplus \widetilde{V} \oplus W, \emptyset + r : \overline{r}\langle \mathrm{id}[W]\rangle \Big\rangle \\ \to (\!|P_1|\!)\{\widetilde{x}, y := \mathrm{id}[\widetilde{V}, W]\} \overset{\nu\{r\}}{\bowtie} \Big\langle \emptyset \oplus \widetilde{V} \oplus W, \emptyset \Big\rangle.$$

By Lemma A.11 and ($\blacktriangleleft$-EXT), it is thus possible to establish the required conclusion by setting $P' = P$.

($\blacktriangleleft$-OUT), ($\blacktriangleleft$-OUT$'$) These cases are similar to the ones triggered by ($\blacktriangleleft$-IN) or ($\blacktriangleleft$-IN$'$), respectively.

($\blacktriangleleft$-|) In this case $P \blacktriangleleft Q \overset{\nu M}{\bowtie} \Omega$ is given as

$$P_1 \mid P_1 \blacktriangleleft Q_1 \mid Q_2 \overset{\nu M_1 \cup M_2}{\bowtie} \Big\langle \epsilon_1 \cup \epsilon_2, \psi_1 \cup \psi_2 \Big\rangle$$

where $P_i \blacktriangleleft Q_1 \overset{\nu M_i}{\bowtie} \langle \epsilon_i, \psi_i \rangle$ $(i = 1, 2)$, $\mathrm{fn}[Q_1 \overset{\nu M_1}{\bowtie} \langle \epsilon_1, \psi_1 \rangle] \cap M_2 = \emptyset$, $M_1 \cap \mathrm{fn}[Q_2 \overset{\nu M_2}{\bowtie} \langle \epsilon_2, \psi_2 \rangle] = \emptyset$ and $M_1 \cap M_2 = \emptyset$. We proceed by considering (i)-(iii) separately. In re-applying ($\blacktriangleleft$-|) we never check the required side conditions explicitly since we may in any case use alpha conversion to establish them.

i. In this case we exploit that return channels are always restricted. This property entails $r \in M$, so by $M_1 \cap M_2 = \emptyset$ we can conclude that $\psi_i + r : \mathit{input}(\mathrm{id}[n]) \in \psi_i$ for either $i = 1$ or $i = 2$. These two cases are symmetric to each other. Hence we may consider only the one where $i = 1$. In this case, by the inductive hypothesis, for all W where $M_1 \cap \mathrm{nm}[W] = \emptyset$ there is a $P_1'[W]$ such that

$$P_1 \xrightarrow{n(W)} P_1'[W] \blacktriangleleft Q_1 \overset{\nu M_1}{\bowtie} \Big\langle \epsilon_1 \oplus W, \psi_1 - r + r : \overline{r}\langle \mathrm{id}[W]\rangle \Big\rangle.$$

Hence, by ($\blacktriangleleft$-|) together with elementary set theory,

$$P_1 \mid P_2 \xrightarrow{n(W)} P_1'[W] \mid P_2 \\ \blacktriangleleft Q_1 \mid Q_2 \overset{\nu M_1 \cup M_2}{\bowtie} \Big\langle (\epsilon_1 \cup \epsilon_2) \oplus W, (\psi_1 \cup \psi_2) - r + r : \overline{r}\langle \mathrm{id}[W]\rangle \Big\rangle$$

for all W where $(M_1 \cup M_2) \cap \mathrm{nm}[W] = \emptyset$, using that this property implies $M_1 \cap \mathrm{nm}[W] = \emptyset$. This establishes the required conclusion if we set $P'[W] = P_1'[W] \mid P_2$.

ii. This case is similar to the previous one.

iii. In this case we proceed by sub-case analysis on the last clause used in inferring the transition by $Q \overset{\nu M}{\bowtie} \langle \epsilon, \psi \rangle$.

($\bowtie$-PAR$_1$) In this case we exploit that encodings never perform any transition that results from any direct internal communication across any $|$-operator that is inherited from the encoded process, so $Q \xrightarrow{\tau} Q'$ has been inferred from $Q_i \xrightarrow{\tau} Q'_i$ for either $i = 1$ or $i = 2$. These two cases are symmetric to each other. Hence we may consider only the one where $i = 1$. In this case, by the induction hypothesis,

$$P_1 \xrightarrow{\widehat{\tau}} P'_1 \sim R \blacktriangleleft Q'_1 \overset{\nu M_1}{\bowtie} \Big\langle \epsilon_1, \psi_1 \Big\rangle$$

for some P'_1 and some R. Hence, by ($\blacktriangleleft$-$|$) and Lemma A.2 – and also by ($|_1$) provided that $P_1 \xrightarrow{\widehat{\tau}} P'_1$ is a proper transition:

$$P_1 \mid P_2 \xrightarrow{\widehat{\tau}} P'_1 \mid P_2 \sim R \mid P_2 \blacktriangleleft Q'_1 \mid Q_2 \overset{\nu M_1 \cup M_2}{\bowtie} \Big\langle \epsilon_1 \cup \epsilon_2, \psi_1 \cup \psi_2 \Big\rangle.$$

This establishes the required conclusion if we set $P' = P'_1 \mid P_2$.

($\bowtie$-PAR$_2$) In this case we exploit that the transition by Ω is labelled with τ, so it has been inferred with $\langle new' \rangle$, $\langle new'_{\mathrm{C}} \rangle$, $\langle apply' \rangle$ or $\langle \mathrm{COM} \rangle$. We proceed by sub-case analysis.

$\langle new' \rangle$, $\langle new'_{\mathrm{C}} \rangle$, $\langle apply' \rangle$. In this case the transition by Ω belongs to a unique abstract integrity management thread which is referred to by a unique return channel r. These channels are always privately shared, so $r \in M_i$ for either $i = 1$ or $i = 2$. These two cases are symmetric to each other. Hence we may consider only the one where $i = 1$. In this case $\langle new' \rangle$, $\langle new'_{\mathrm{C}} \rangle$ and $\langle apply' \rangle$ respectively entail that there is a transition of the form $\langle \epsilon_1, \psi_1 \rangle \xrightarrow{\tau} \langle \epsilon'_1, \psi'_1 \rangle$ such that $\epsilon' = \epsilon'_1 \cup \epsilon_2$ and $\psi' = \psi'_1 \cup \psi_2$. Hence, by the induction hypothesis,

$$P_1 \xrightarrow{\widehat{\tau}} P'_1 \sim R \blacktriangleleft Q_1 \overset{\nu M_1}{\bowtie} \Big\langle \epsilon'_1, \psi'_1 \Big\rangle$$

for some P'_1 and some R. Hence, by ($\blacktriangleleft$-$|$) and Lemma A.2 – and also by ($|_1$) provided that $P_1 \xrightarrow{\widehat{\tau}} P'_1$ is a proper transition:

$$P_1 \mid P_2 \xrightarrow{\widehat{\tau}} P'_1 \mid P_2 \blacktriangleleft R \mid P_2 Q_1 \mid Q_2 \overset{\nu M_1 \cup M_2}{\bowtie} \Big\langle \epsilon'_1 \cup \epsilon_2, \psi'_1 \cup \psi_2 \Big\rangle.$$

This establishes the required conclusion if we set $P' = P'_1 \mid P_2$.

$\langle \mathrm{COM} \rangle$ In this case the transition by Ω belongs to two abstract integrity management threads – one for input, one for output – which are referred to by unique return channels r or s, respectively. By $M_1 \cap M_2 = \emptyset$, either both r and s belong to either M_1 or M_2, or $r \in M_1$ and $s \in M_2$, or $r \in M_2$ and $s \in M_1$. By symmetry, we may consider only the following cases:

$r \in M_1$ and $s \in M_1$. This case can be treated in a similar way as the one triggered by $\langle new' \rangle$, $\langle new'_{\mathrm{C}} \rangle$ and $\langle apply' \rangle$.

$r \in M_1$ and $s \in M_2$. In this case there are an n and a W such that $r : \mathit{input}(\mathrm{id}[n]) \in \psi_1$ and $s : \mathit{output}(\mathrm{id}[n], \mathrm{id}[W]) \in \psi_2$ where $\epsilon{\downarrow}\,\mathrm{id}[n]$, $\epsilon{\downarrow}\,\mathrm{id}[W]$ and

$$\psi' = \psi - r + r : \overline{r}\langle \mathrm{id}[W] \rangle - s + s : \overline{s}\langle\rangle.$$

Because $M_1 \cap M_2 = \emptyset$, n does not occur in both M_1 and M_2. Further, by $(\mathrm{C}_{1,\,2})$, n does not occur in only one of M_1 and M_2 and, at the same time, in ψ_2 or ψ_1, respectively. Hence $n \notin M_i$ $(i = 1, 2)$. Also, by alpha-conversion, we may assume $M_1 \cap \mathrm{nm}[W] = \emptyset$. Hence, by the induction hypothesis,

$$P_1 \xrightarrow{n(W)} P_1' \blacktriangleleft Q_1 \overset{\nu\, M_1}{\bowtie} \Big\langle \epsilon_1 \oplus W, \psi_1 - r + r : \overline{r}\langle \mathrm{id}[W] \rangle \Big\rangle$$

for some P_1' and

$$P_2 \xrightarrow{(\nu\, K_2)\,\overline{n}\langle W \rangle} P_2' \blacktriangleleft Q_2 \overset{\nu\, M_2}{\bowtie} \Big\langle \epsilon_2, \psi_2 - s + s : \overline{r}\langle\rangle \Big\rangle.$$

for some P_2' and some K_2 where $K_2 \subseteq M_2$. Again by alpha conversion, we may assume $\mathrm{fn}[P_1] \cap K_2 = \emptyset$. Hence, together with $(\textsc{close}_1)$, we may derive

$$P_1 \mid P_2 \xrightarrow{\tau} (\nu\, K_2)(P_1' \mid P_2').$$

Further, by $(\blacktriangleleft\text{-}|)$ and elementary set-theory, we may derive

$$P_1' \mid P_2' \blacktriangleleft Q \overset{\nu\, M}{\bowtie} \Big\langle \epsilon, \psi' \Big\rangle \tag{1}$$

using $M_1 \cap M_2 = \emptyset$ and the fact that the well-formedness of $Q_2 \overset{\nu\, M_2}{\bowtie} \langle \epsilon_2, \psi_2 \rangle$ implies $\epsilon_2 = \epsilon_2 \oplus W$, so $\epsilon = \epsilon \oplus W$. Still further, $K_2 \subseteq M$ and, at the same time, $Q \overset{\nu\, M}{\bowtie} \langle \epsilon, \psi' \rangle$ is well-formed. Hence, applying $(\blacktriangleleft\text{-}\nu_1)$ and $(\blacktriangleleft\text{-}\nu_2)$ successively with (1) as starting point does not enlarge M, so

$$(\nu\, K_2)(P_1' \mid P_2') \blacktriangleleft Q \overset{\nu\, M}{\bowtie} \Big\langle \epsilon, \psi' \Big\rangle.$$

This establishes the required conclusion if we set $P' = (\nu\, K_2)(P_1' \mid P_2')$.

$(\bowtie\text{-}\textsc{req})$, $(\bowtie\text{-}\textsc{ret}_1)$, $(\bowtie\text{-}\textsc{ret}_2)$ In this case the transition by $Q \overset{\nu\, M}{\bowtie} \Omega$ has been derived from a transition by either Q_1 or Q_2, and a transition by Ω, where the transition by Ω has been inferred with $\langle \mathit{new} \rangle$, $\langle \mathit{new}_{\mathrm{C}} \rangle$, $\langle \mathit{input} \rangle$, $\langle \mathit{output} \rangle$, $\langle \mathit{apply} \rangle$, $\langle \textsc{ret}_1 \rangle$ or $\langle \textsc{ret}_2 \rangle$. These two cases are symmetric to each other. Hence we may consider only the first one. In each sub-case of this case there is a transition of the form $Q_1 \overset{\nu\, M_1}{\bowtie} \langle \epsilon_1, \psi_1 \rangle \xrightarrow{\tau} Q_1' \overset{\nu\, M_1'}{\bowtie} \langle \epsilon_1, \psi_1' \rangle$ that has also been inferred with $\langle \mathit{new} \rangle$, $\langle \mathit{new}_{\mathrm{C}} \rangle$, $\langle \mathit{input} \rangle$, $\langle \mathit{output} \rangle$, $\langle \mathit{apply} \rangle$, $\langle \textsc{ret}_1 \rangle$ or $\langle \textsc{ret}_2 \rangle$ respectively, and where (a) $Q' = Q_1' \mid Q_2$, (b) $M' = M_1' \,\dot\cup\, M_2$ since we may in any case use alpha conversion to ensure $M_1' \cap M_2 = \emptyset$ and (c) $\psi' = \psi_1' \cup \psi_2$. Hence, by the induction hypothesis,

$$P_1 \xrightarrow{\widehat{\tau}} P_1' \sim R \blacktriangleleft Q_1' \overset{\nu\, M_1'}{\bowtie} \Big\langle \epsilon_1, \psi_1' \Big\rangle$$

for some P_1' and some R. Hence, by ($\blacktriangleleft$-|) and Lemma A.2 – and also by ($|_1$) provided that $P_1 \xrightarrow{\widehat{\tau}} P_1'$ is a proper transition:

$$P_1 \mid P_2 \xrightarrow{\widehat{\tau}} P_1' \mid P_2 \sim R \mid P_2 \blacktriangleleft Q_1' \mid Q_2 \;{}^{\nu\, M_1' \cup M_2}\!\bowtie \Big\langle \epsilon_1 \cup \epsilon_2, \psi_1' \cup \psi_2 \Big\rangle.$$

This establishes the required conclusion if we set $P' = P_1' \mid P_2$.

($\blacktriangleleft$-ν_1) In this case $P \blacktriangleleft Q \;{}^{\nu\, M}\!\bowtie \Sigma$ is given as

$$(\nu\, m)\, P_1 \blacktriangleleft (\nu\, m)\, Q_1 \;{}^{\nu\, M}\!\bowtie \Big\langle \epsilon, \psi \Big\rangle$$

where $P_1 \blacktriangleleft Q_1 \;{}^{\nu\, M}\!\bowtie \langle \epsilon, \psi \rangle$, $\psi \not\upharpoonright \mathit{new}(m)$ and $\epsilon \not\downarrow \mathrm{id}[m]$. We proceed by considering (i)-(iii) separately.

i. In this case $r : \mathit{input}(\mathrm{id}[n]) \in \psi$ for some r and some n where $n \notin M$. Hence, by the induction hypothesis, for all W where $M \cap \mathrm{nm}[W] = \emptyset$ there is a $P_1'[W]$ such that

$$P_1 \xrightarrow{n(W)} P_1'[W] \blacktriangleleft Q_1 \;{}^{\nu\, M}\!\bowtie \Big\langle \epsilon \oplus W, \psi - r + r : \overline{r}\langle \mathrm{id}[W] \rangle \Big\rangle.$$

A straightforward induction on the inference used to establish $P_1 \blacktriangleleft Q_1 \;{}^{\nu\, M}\!\bowtie \langle \epsilon, \psi \rangle$ proves that $r : \mathit{input}(\mathrm{id}[n]) \in \psi$ entails $\epsilon{\downarrow}\,\mathrm{id}[n]$, so $\epsilon \not\downarrow \mathrm{id}[m]$ entails $m \neq n$. Also, by alpha-conversion, we may assume $m \notin \mathrm{nm}[W]$, so (ν) entails

$$(\nu\, m)\, P_1 \xrightarrow{n(W)} (\nu\, m)\, P_1'[W]. \tag{2}$$

Further, $\psi \not\upharpoonright \mathit{new}(m)$ implies $\psi - r + r : \overline{r}\langle \mathrm{id}[W] \rangle \not\upharpoonright \mathit{new}(m)$, and $\epsilon \not\downarrow \mathrm{id}[m]$ and $m \notin \mathrm{nm}[W]$ together imply $\epsilon \oplus W \not\downarrow \mathrm{id}[m]$, so ($\blacktriangleleft$-$\nu_1$) entails

$$(\nu\, m)\, P_1'[W] \blacktriangleleft (\nu\, m)\, Q_1 \;{}^{\nu\, M}\!\bowtie \Big\langle \epsilon \oplus W, \psi - r + r : \overline{r}\langle \mathrm{id}[W] \rangle \Big\rangle. \tag{3}$$

Taken together, (2) and (3) establish the required conclusion if we set $P'[W] = (\nu\, m)\, P_1'[W]$.

ii. In this case $r : \mathit{output}(\mathrm{id}[n], \mathrm{id}[W]) \in \psi$ for some r, some n and some W where $n \notin M$. Hence, by the induction hypothesis:

$$P_1 \xrightarrow{(\nu\, K_1)\, \overline{n}\langle W \rangle} P_1' \blacktriangleleft Q_1 \;{}^{\nu\, M}\!\bowtie \Big\langle \epsilon, \psi - r + r : \overline{r}\langle\rangle \Big\rangle.$$

for some P_1' and some K_1 where $K_1 \subseteq M$. A straightforward induction on the inference used to establish $P_1 \blacktriangleleft Q_1 \;{}^{\nu\, M}\!\bowtie \langle \epsilon, \psi \rangle$ proves that $r : \mathit{output}(\mathrm{id}[n], \mathrm{id}[W]) \in \psi$ entails $\epsilon{\downarrow}\,\mathrm{id}[n]$ and $\epsilon{\downarrow}\,\mathrm{id}[W]$, so $\epsilon \not\downarrow \mathrm{id}[m]$ entails $m \notin n + \mathrm{nm}[W]$, so ($\nu$) entails

$$(\nu\, m)\, P_1 \xrightarrow{(\nu\, K_1)\, \overline{n}\langle W \rangle} (\nu\, m)\, P_1'. \tag{4}$$

Further, $\psi \not\upharpoonright \mathit{new}(m)$ implies $\psi - r + r : \overline{r}\langle \mathrm{id}[W] \rangle \not\upharpoonright \mathit{new}(m)$, and $\epsilon \not\downarrow \mathrm{id}[m]$ and $m \notin \mathrm{nm}[W]$ together imply $\epsilon \oplus W \not\downarrow \mathrm{id}[m]$, so ($\blacktriangleleft$-$\nu_1$) entails

$$(\nu\, m)\, P_1' \blacktriangleleft (\nu\, m)\, Q_1 \;{}^{\nu\, M}\!\bowtie \Big\langle \epsilon, \psi - r + r : \overline{r}\langle\rangle \Big\rangle. \tag{5}$$

Taken together, (4) and (5) establish the required conclusion if we set $P' = (\nu\, m)\, P_1'$.

iii. In this case we distinguish as to whether $Q \overset{\nu M}{\bowtie} \Sigma \rightarrow Q' \overset{\nu M'}{\bowtie} \Sigma'$ makes m privately shared or not. If it does not, then a straightforward analysis of the possible cases proves that

$$Q' \overset{\nu M'}{\bowtie} \left\langle \epsilon', \psi' \right\rangle = (\nu\, m)\, Q_1' \overset{\nu M'}{\bowtie} \left\langle \epsilon', \psi' \right\rangle \tag{6}$$

for some Q_1' where $Q_1 \overset{\nu M}{\bowtie} \Sigma \rightarrow Q_1' \overset{\nu M'}{\bowtie} \langle \epsilon', \psi' \rangle$; if it does, then

$$Q' \overset{\nu M'}{\bowtie} \left\langle \epsilon', \psi' \right\rangle = Q_1' \overset{\nu M_1'+m}{\bowtie} \left\langle \epsilon', \psi' \right\rangle \tag{7}$$

for some some Q_1' and some M_1' where $Q_1 \overset{\nu M}{\bowtie} \Sigma \rightarrow Q_1' \overset{\nu M_1'}{\bowtie} \langle \epsilon', \psi' \rangle$.

In the case of (6), by the inductive hypothesis,

$$P_1 \overset{\widehat{\tau}}{\rightarrow} P_1' \sim R \blacktriangleleft Q_1' \overset{\nu M'}{\bowtie} \left\langle \epsilon', \psi' \right\rangle$$

for some P_1' and some R. Because m has not become privately shared in $Q' \overset{\nu M'}{\bowtie} \langle \epsilon', \psi' \rangle$ while $\epsilon \not\downarrow \mathrm{id}[m]$ and $\psi \not\uparrow new(n)$, $\psi' \not\uparrow new(n)$ and $\epsilon' \not\downarrow \mathrm{id}[m]$. Hence, by ($\blacktriangleleft$-$\nu_1$) and Lemma A.2 – and also by (ν) provided that $P_1 \overset{\widehat{\tau}}{\rightarrow} P_1'$ is a proper transition:

$$(\nu\, m)\, P_1 \overset{\widehat{\tau}}{\rightarrow} (\nu\, m)\, P_1' \sim (\nu\, m)\, R \blacktriangleleft (\nu\, m)\, Q_1' \overset{\nu M'}{\bowtie} \left\langle \epsilon', \psi' \right\rangle.$$

This establishes the required conclusion if we set $P' = (\nu\, m)\, P_1'$.

In the case of (7), by the inductive hypothesis,

$$P_1 \overset{\widehat{\tau}}{\rightarrow} P_1' \sim R \blacktriangleleft Q_1' \overset{\nu M_1'}{\bowtie} \left\langle \epsilon', \psi' \right\rangle$$

for some P_1'. Because m has become privately shared in $Q' \overset{\nu M'}{\bowtie} \langle \epsilon', \psi' \rangle$ while $\psi \not\uparrow new(n)$ and $\epsilon \not\downarrow \mathrm{id}[m]$, $\psi' \uparrow new(n)$ and $\epsilon' \not\downarrow \mathrm{id}[m]$. Hence, by ($\blacktriangleleft$-$\nu_2$) and Lemma A.2 – and also by (ν) provided that $P_1 \overset{\widehat{\tau}}{\rightarrow} P_1'$ is a proper transition:

$$(\nu\, m)\, P_1 \overset{\widehat{\tau}}{\rightarrow} (\nu\, m)\, P_1' \sim (\nu\, m)\, R \blacktriangleleft Q_1' \overset{\nu M_1'+m}{\bowtie} \left\langle \epsilon', \psi' \right\rangle.$$

This establishes the required conclusion if we set $P' = (\nu\, m)\, P_1'$.

($\blacktriangleleft$-ν_2) In this case $P \blacktriangleleft Q \overset{\nu M}{\bowtie} \Sigma$ is given as

$$(\nu\, m)\, P_1 \blacktriangleleft Q \overset{\nu M_1+m}{\bowtie} \left\langle \epsilon, \psi \right\rangle$$

where $P_1 \blacktriangleleft Q \overset{\nu M_1}{\bowtie} \langle \epsilon, \psi \rangle$ and (a) $\psi \uparrow new(m)$ or (b) $\epsilon \downarrow \mathrm{id}[m]$. We proceed by considering (i)-(iii) separately.

i. In this case $r : \mathit{input}(\mathrm{id}[n]) \in \psi$ for some r and for some n where $n \notin M$, so $n \notin M_1$. Hence, by the induction hypothesis, for all W where $M_1 \cap \mathrm{nm}[W] = \emptyset$ there is a $P_1'[W]$ such that

$$P_1 \xrightarrow{n(W)} P_1'[W] \blacktriangleleft Q \overset{\nu\, M_1}{\bowtie} \Big\langle \epsilon \oplus W, \psi - r + r : \overline{r}\langle \mathrm{id}[W] \rangle \Big\rangle.$$

By $n \notin M$: $n \neq m$. Also, by alpha-conversion, we may assume $m \notin \mathrm{nm}[W]$, so (ν) entails

$$(\nu\, m)\, P_1 \xrightarrow{n(W)} (\nu\, m)\, P_1'[W]. \tag{8}$$

for all W where $(M_1 + m) \cap \mathrm{nm}[W] = \emptyset$, using that this property implies $M_1 \cap \mathrm{nm}[W] = \emptyset$. Further, as for re-establishing (a) or (b), (a) $\psi\!\uparrow \mathit{new}(m)$ and $\psi[r] = \mathit{input}(\mathrm{id}[n])$ together imply $(\psi - r + r : \overline{r}\langle \mathrm{id}[W] \rangle)\!\uparrow \mathit{new}(m)$; (b) $\epsilon\!\downarrow \mathrm{id}[m]$ implies $\epsilon \oplus W\!\downarrow \mathrm{id}[m]$. Hence ($\blacktriangleleft$-$\nu_2$) entails

$$(\nu\, m)\, P_1'[W] \blacktriangleleft Q \overset{\nu\, M_1 + m}{\bowtie} \Big\langle \epsilon \oplus W, \psi - r + r : \overline{r}\langle \mathrm{id}[W] \rangle \Big\rangle. \tag{9}$$

Taken together, (8) and (9) establish the required conclusion if we set $P'[W] = (\nu\, m)\, P_1'[W]$.

ii. In this case $r : \mathit{output}(\mathrm{id}[n], \mathrm{id}[W]) \in \psi$ for some r, some n and some W where $n \notin M$, so $n \notin M_1$. Hence, by the induction hypothesis,

$$P_1 \xrightarrow{(\nu\, K_1)\, \overline{n}\langle W \rangle} P_1' \blacktriangleleft Q \overset{\nu\, M_1}{\bowtie} \Big\langle \epsilon, \psi - r + r : \overline{r}\langle\rangle \Big\rangle. \tag{10}$$

for some P_1' and some K_1 where $K_1 \subseteq M_1$. By $n \notin M$: $n \neq m$. Also, by alpha conversion, we may assume $m \notin K_1$. We proceed by distinguishing as to whether $m \in \mathrm{nm}[W] \setminus K_1$ or not.

If $m \notin \mathrm{nm}[W] \setminus K_1$, then (ν) entails

$$(\nu\, m)\, P_1 \xrightarrow{(\nu\, K_1)\, \overline{n}\langle W \rangle} (\nu\, m)\, P_1'. \tag{11}$$

Further, $\psi\!\uparrow \mathit{new}(n)$ and $\psi[r] = \mathit{output}(\mathrm{id}[n], \mathrm{id}[W])$ together entail $(\psi - r + r : \overline{r}\langle\rangle)\!\uparrow \mathit{new}(n)$, so ($\blacktriangleleft$-$\nu_2$) entails

$$(\nu\, m)\, P_1' \blacktriangleleft Q \overset{\nu\, M_1 + m}{\bowtie} \Big\langle \epsilon, \psi - r + r : \overline{r}\langle\rangle \Big\rangle. \tag{12}$$

Taken together, (11) and (12) establish the required conclusion if we set $P' = (\nu\, m)\, P_1'$.

If $m \in \mathrm{nm}[W] \setminus K_1$, then (OPEN) entails

$$(\nu\, m)\, P_1 \xrightarrow{(\nu\, K_1 + m)\, \overline{n}\langle W \rangle} P_1'. \tag{13}$$

Further, $Q \overset{\nu\, M_1 + m}{\bowtie} \Big\langle \epsilon, \psi - r + r : \overline{r}\langle\rangle \Big\rangle$ is well-formed. Hence, applying ($\blacktriangleleft$-EXT) to (10) yields

$$P_1' \blacktriangleleft Q \overset{\nu\, M_1 + m}{\bowtie} \Big\langle \epsilon, \psi - r + r : \overline{r}\langle\rangle \Big\rangle. \tag{14}$$

Taken together, (13) and (14) establish the required conclusion if we set $P' = P_1'$.

iii. In this case a straightforward analysis of the possible cases shows that

$$Q' \overset{\nu M'}{\bowtie} \left\langle \epsilon', \psi' \right\rangle = Q' \overset{\nu M_1'+m}{\bowtie} \left\langle \epsilon', \psi' \right\rangle$$

for some M_1' where $Q \overset{\nu M_1}{\bowtie} \langle \epsilon, \psi \rangle \rightarrow Q' \overset{\nu M_1'}{\bowtie} \langle \epsilon', \psi' \rangle$. Hence, by the induction hypothesis,

$$P_1 \overset{\widehat{\tau}}{\rightarrow} P_1' \sim R \blacktriangleleft Q' \overset{\nu M_1'}{\bowtie} \langle \epsilon', \psi' \rangle$$

for some P_1' and some R. Further, (a) or (b) entails $\psi' \uparrow new(m)$ or $\epsilon' \downarrow \mathrm{id}[m]$. Hence, by ($\blacktriangleleft$-$\nu_2$) and Lemma A.2 – and also by (ν) provided that $P_1 \overset{\widehat{\tau}}{\rightarrow} P_1'$ is a proper transition,

$$(\nu\, m)\, P_1 \overset{\widehat{\tau}}{\rightarrow} (\nu\, m)\, P_1' \sim (\nu\, m)\, R \blacktriangleleft Q' \overset{\nu M_1'+m}{\bowtie} \langle \epsilon', \psi' \rangle.$$

This establishes the required conclusion if we set $P' = (\nu\, m)\, P_1'$.

($\blacktriangleleft$-!) In this case $P \blacktriangleleft Q \overset{\nu M}{\bowtie} \Sigma$ is given as

$$(!\, P_1)\{\widetilde{x} := \widetilde{V}\} \blacktriangleleft\, !(\!|P_1|\!)\{\widetilde{x} := \mathrm{id}[\widetilde{V}]\} \bowtie \left\langle \emptyset \oplus \widetilde{V}, \emptyset \right\rangle.$$

Any "sub-case" triggered by (i) or (ii) is impossible. As for (iii), we exploit once more that encodings never perform any transition that results from any direct internal communication across any |-operator that is inherited from the encoded process. Hence $Q \overset{\nu M}{\bowtie} \Sigma \rightarrow Q' \overset{\nu M'}{\bowtie} \Sigma'$ has been inferred via

$$!(\!|P_1|\!)\{\widetilde{x} := \mathrm{id}[\widetilde{V}]\} \overset{\alpha}{\rightarrow} Q_1' \mid (!(\!|P_1|\!)\{\widetilde{x} := \mathrm{id}[\widetilde{V}]\})$$

from a transition of the form

$$(\!|P_1|\!)\{\widetilde{x} := \mathrm{id}[\widetilde{V}]\} \overset{\alpha}{\rightarrow} Q_1'$$

with or without participation by a complementary transition by $\left\langle \emptyset \oplus \widetilde{V}, \emptyset \right\rangle$. Hence

$$(\!|P_1|\!)\{\widetilde{x} := \mathrm{id}[\widetilde{V}]\} \bowtie \left\langle \emptyset \oplus \widetilde{V}, \emptyset \right\rangle \rightarrow Q_1' \overset{\nu M'}{\bowtie} \Sigma',$$

where $P_1\{\widetilde{x} := \widetilde{V}\} \blacktriangleleft (\!|P_1|\!)\{\widetilde{x} := \mathrm{id}[\widetilde{V}]\} \bowtie \langle \emptyset \oplus \widetilde{V}, \emptyset \rangle$ due to Lemma A.11. Hence the induction hypothesis entails

$$P_1\{\widetilde{x} := \widetilde{V}\} \overset{\widehat{\tau}}{\rightarrow} P_1' \sim R \blacktriangleleft Q_1' \overset{\nu M'}{\bowtie} \Sigma'$$

for some P_1' and some R. Considering $P_1' \sim R \blacktriangleleft Q_1' \overset{\nu M'}{\bowtie} \Sigma'$, Lemma A.2 and ($\blacktriangleleft$-|) together entail

$$P_1' \mid (!\, P_1)\{\widetilde{x} := \mathrm{id}[\widetilde{V}]\} \sim R \mid (!\, P_1)\{\widetilde{x} := \mathrm{id}[\widetilde{V}]\} \blacktriangleleft Q_1' \mid (!(\!|P_1|\!)\{\widetilde{x} := \mathrm{id}[\widetilde{V}]\}) \overset{\nu M'}{\bowtie} \Sigma'$$

(where $Q_1' \mid (!(\!|P_1|\!)\{\widetilde{x} := \mathrm{id}[\widetilde{V}]\}) = Q'$). If $P_1\{\widetilde{x} := \widetilde{V}\} \xrightarrow{\widehat{\tau}} P_1'$ is a proper transition then (!-|) entails

$$(!\,P_1)\{\widetilde{x} := \widetilde{V}\} \xrightarrow{\tau} P_1' \mid (!\,P_1)\{\widetilde{x} := \mathrm{id}[\widetilde{V}]\}$$

– This establishes the required conclusion if we set $P' = P_1' \mid (!\,P_1)\{\widetilde{x} := \mathrm{id}[\widetilde{V}]\}$. If $P_1\{\widetilde{x} := \widetilde{V}\} \xrightarrow{\widehat{\tau}} P_1'$ is not a proper transition, then $P_1' = P_1\{\widetilde{x} := \mathrm{id}[\widetilde{V}]\}$, where Lemma A.3 entails

$$(!\,P_1)\{\widetilde{x} := \mathrm{id}[\widetilde{V}]\} \sim P_1\{\widetilde{x} := \mathrm{id}[\widetilde{V}]\} \mid (!\,P_1)\{\widetilde{x} := \mathrm{id}[\widetilde{V}]\}.$$

This establishes the required conclusion if we set $P' = (P_1 \mid !\,P_1)\{\widetilde{x} := \mathrm{id}[\widetilde{V}]\}$.

(◀-**if**), (◀-**let**) These cases are straightforward. Hence they may be left implicit.

(◀-EXT) In this case $P \blacktriangleleft Q \overset{\nu\, M}{\bowtie} \Sigma$ is given as

$$P \blacktriangleleft Q \overset{\nu\, M}{\bowtie} \Big\langle \epsilon, \psi \Big\rangle$$

where $P \blacktriangleleft Q \overset{\nu\, M_1}{\bowtie} \langle \epsilon_1, \psi \rangle$ for some M_1 and some ϵ_1 such that $M_1 \subseteq M$ and $\epsilon_1 \subseteq \epsilon$. We proceed by considering (i)-(iii) separately.

i. In this case $r : \mathit{input}(\mathrm{id}[n]) \in \psi$ for some r and some n where $n \notin M$. Hence, by $M_1 \subseteq M$: $n \notin M_1$. Hence, by the induction hypothesis, for all W where $M_1 \cap \mathrm{nm}[W] = \emptyset$ there are a $P'[W]$ and an $R[W]$ such that

$$P \xrightarrow{n(W)} P'[W] \sim R[W] \blacktriangleleft Q \overset{\nu\, M_1}{\bowtie} \Big\langle \epsilon_1 \oplus W, \psi - r + r : \overline{r}\langle \mathrm{id}[W] \rangle \Big\rangle.$$

Hence, by (◀-EXT),

$$P \xrightarrow{n(W)} P'[W] \sim R[W] \blacktriangleleft Q \overset{\nu\, M}{\bowtie} \Big\langle \epsilon \oplus W, \psi - r + r : \overline{r}\langle \mathrm{id}[W] \rangle \Big\rangle$$

for all W where $M \cap \mathrm{nm}[W] = \emptyset$, using that this property entails $M_1 \cap \mathrm{nm}[W] = \emptyset$. This establishes the required conclusion.

ii. In this case $r : \mathit{output}(\mathrm{id}[n], \mathrm{id}[W]) \in \psi$ for some r, some n and some W where $n \notin M$. Hence, by $M_1 \subseteq M$: $n \notin M_1$. Hence, by the induction hypothesis

$$P \xrightarrow{(\nu\, K)\,\overline{n}\langle W \rangle} P' \sim R \blacktriangleleft Q \overset{\nu\, M_1}{\bowtie} \Big\langle \epsilon_1, \psi - r + r : \overline{r}\langle\rangle \Big\rangle$$

for some P' and some R where $K \subseteq M_1$. Hence, by (◀-EXT),

$$P \xrightarrow{(\nu\, K)\,\overline{n}\langle W \rangle} P' \sim R \blacktriangleleft Q \overset{\nu\, M}{\bowtie} \Big\langle \epsilon, \psi - r + r : \overline{r}\langle\rangle \Big\rangle$$

where $K \subseteq M$. This establishes the required conclusion.

iii. In this case we exploit that the encoding and ◀ together preserve the consistency of any descendant of any πT-process in the sense that every activity within the descendant that depends on the prior allocation of some value ID will always find this condition satisfied. Hence $Q \overset{\nu\, M}{\bowtie} \langle \epsilon, \psi \rangle \to Q \overset{\nu\, M'}{\bowtie} \langle \epsilon', \psi' \rangle$ can be reduced to a

transition of the form $Q \overset{\nu M_1}{\bowtie} \langle \epsilon_1, \psi \rangle \rightarrow Q \overset{\nu M_1'}{\bowtie} \langle \epsilon_1', \psi' \rangle$ where $M_1' \subseteq M'$ and $\epsilon_1' \subseteq \epsilon'$. Hence, by the inductive hypothesis,

$$P \xrightarrow{\widehat{\tau}} P' \sim R \blacktriangleleft Q \overset{\nu M_1'}{\bowtie} \langle \epsilon_1', \psi' \rangle$$

for some P' and some R. Hence, ($\blacktriangleleft$-EXT) entails

$$P \xrightarrow{\widehat{\tau}} P' \sim R \blacktriangleleft Q \overset{\nu M}{\bowtie} \langle \epsilon', \psi' \rangle.$$

This establishes the required conclusion.

□

A.1.5 Finalising the Proof of Theorem A.7

Proof of Theorem A.7. As an easy consequence of Theorems A.12 and A.13. □

A.2 Equivalence of May-Tests as Mediated by Abstract and Concrete Integrity Management

Lemma A.14 *P $\text{may}_{\bowtie}$ O if and only if P $\text{may}_{\mathbf{M}}$ O*

To prove Lemma A.14, we introduce the following notational conventions:

1. *EVT* ranges over all states of the value table in which it has not spawned off any value entries(s).
2. *VE*(i) ranges over all states of any value entry that has been instantiated with i as the actual ID of the value to which the record corresponds.
3. *NE*(i, n, a) ranges over all states of any name entry that has been instantiated with i, n and a as its actual parameters.
4. $CTE_f(i_1', \ldots, i_{\text{ar}[f]}')$ ranges over all states of any complex value entry that has been been instantiated with i_1', …, $i_{\text{ar}[f]}'$ as its actual parameters.
5. *NRT*(n, r) ranges over all states of any name registration thread that has been instantiated with n and r as actual parameters.
6. $CRT_f(i_1', \ldots, i_{\text{ar}[f]}', r)$ ranges over all states of any f-constructor term interpretation thread that has been instantiated with i_1', …, $i_{\text{ar}[f]}'$ and r as actual parameters.
7. *IIT*(i, r) ranges over all states of any input interpretation thread that has been instantiated with i and r as actual parameters.
8. *OIT*(i, j, r) ranges over all states of any output interpretation thread that has been instantiated with i, j and r as actual parameters.
9. $EIT_E(i_1, \ldots, i_{\text{ar}[E]}, r)$ ranges over all states of any E-application interpretation thread that has been instantiated with i_1, …, $i_{\text{ar}[E]}$ and r as actual parameters.

10. *Mut* ranges over all states of the semaphore.

The parameters are sometimes un-important or implicit. In these case we omit them. Moreover, we denote any linkage of the form $(\nu\, N)(VE \mid P)$ by by $VE \frown P$.

Modulo garbage in the form of **0**-processes and internal names that are not used anymore, any state S of the integrity manager can then be written as shown in Figure A.2. The garbage will later be taken account of by working up to strong bisimilarity. where:

Figure 5 General schema for states of the integrity manager.

$$
\begin{aligned}
&S \\
&\quad = (\nu\, \{lock, unlock, put, getId, getAlias, [\forall f \in \mathbb{F}.put_f, getId_f, getArgIds_f], lookUp\} \\
&\qquad \cup A \cup S \cup I \\
&\quad)(\, VE_1 \frown \ldots \frown VE_{k_{\mathbf{V}}} \frown EVT \\
&\quad \mid \\
&\qquad NRT_1 \mid \ldots \mid NRT_{k_{\mathbf{N}}} \mid \mathbf{NameRegistrar}(\ldots) \\
&\quad \mid \\
&\qquad [\textstyle\prod_{f \in \mathbb{F}} CRT_{f,1} \mid \ldots \mid CRT_{f,k_{\mathbf{B},f}} \mid \mathbf{BuildInterpreter}_f(\ldots)] \\
&\quad \mid \\
&\qquad IIT_1 \mid \ldots \mid IIT_{k_{\mathbf{I}}} \mid \mathbf{InputInterpreter}(\ldots) \\
&\quad \mid \\
&\qquad OIT_1 \mid \ldots \mid OIT_{k_{\mathbf{O}}} \mid \mathbf{OutputInterpreter}(\ldots) \\
&\quad \mid \\
&\qquad [\textstyle\prod_{E \in \mathbb{E}} EIT_{E,1} \mid \ldots \mid EIT_{E,k_{\mathbf{D},E}} \mid \mathbf{DeconstructionInterpreter}_E(\ldots)] \\
&\quad \mid \\
&\qquad Mut \\
&\quad)
\end{aligned}
$$

1. $k_{\mathbf{V}}, k_{\mathbf{N}}, [\forall f \in \mathbb{F}.k_{\mathbf{B}_f}], k_{\mathbf{I}}, k_{\mathbf{O}}, [\forall E \in \mathbb{E}.k_{\mathbf{D}_E}] \geq 0$
2. A comprises all name aliases that are held in the value table.
3. $S = \emptyset$ or $S = \{s\}$ where, in the case of $S = \{s\}$, s is the return channel of the one and only call that is being served by the value table. There can only be at most one such call since the value table is protected by the semaphore.
4. I comprises all value IDs that have been allocated but, at the same time, not yet returned to the environment.

In addition to that a number of invariants hold, which we leave largely implicit. The only one that we make explicit consists of (i–iii) below. It pertains to the value table.

i. Different entries in the value table correspond to different values and, the the same time, hold distinct value IDs.

ii. If $k \geq 1$, then there exist i, n, a such that $VE_1 = NE(i, n, a)$.

iii. If $VE_{k'} = CTE_f(i, i'_1, \ldots, i'_{\mathrm{ar}[f]})$, then the value table contains a $VE_{k''}(i')$, $k'' < k'$, for any $i' \in \{i'_1, \ldots, i'_{\mathrm{ar}[f]}\}$.

Property (i) gives rise to a canonical partial mapping val_S. This mapping yields the value referred to by any value ID held in the value table in any state S. Further, the properties (i) and (iii) together entail a mapping δ_S in the sense of Section A.1.2 such that: For each value ID i that is held by some entry in the value table,

i. δ_S is defined on $\mathrm{id}[\mathrm{val}_S[i]]$ and

ii.
$$\delta_S[\mathrm{id}[\mathrm{val}_S[i]]] = \begin{cases} \text{-} & \text{if } \mathrm{val}_S[i] = n \\ \langle f, \mathrm{id}[V_1], \ldots, \mathrm{id}[V_{\mathrm{ar}[f]}]\rangle & \text{if } \mathrm{val}_S[i] = f(V_1, \ldots, V_{\mathrm{ar}[f]}). \end{cases}$$

Still further, we stipulate abstractions of the concrete interpretation thread states in S. These abstractions are specified below. In each case there are two sub-cases: In the first one the thread has not (yet) reached a state in which all what is left is to deliver a result to the caller; in the second one it has reached such as state, in which j is the value ID that is to be returned to the caller except in the case of an output interpretation, where there is no such return ID.

1. $A_S(NRT(n,r)) = \begin{cases} r : new(n,r) \\ r : \overline{r}\langle \mathrm{id}[\mathrm{val}_S[j]]\rangle \end{cases}$

2. $A_S(CRT_f(i'_1, \ldots, i'_{\mathrm{ar}[f]}, r)) = \begin{cases} r : new_f(i'_1, \ldots, i'_{\mathrm{ar}[f]}) \\ r : \overline{r}\langle \mathrm{id}[\mathrm{val}_S[j]]\rangle \end{cases}$

3. $A_S(IIT(i,r)) = \begin{cases} r : input(i,r) \\ r : \overline{r}\langle \mathrm{id}[\mathrm{val}_S[j]]\rangle \end{cases}$

4. $A_S(OIT(i,j,r)) = \begin{cases} r : output(i,j,r) \\ r : \overline{r}\langle\rangle \end{cases}$

5. $A_S(EIT_E(i_1, \ldots, i_{\mathrm{ar}[E]})) = \begin{cases} r : apply_E(i_1, \ldots, i_{\mathrm{ar}[E]}) \\ r : \overline{r}\langle \mathrm{id}[\mathrm{val}_S[j]]\rangle \end{cases}$

Process encodings and their derivatives never re-use any return channels, whence the return channel of each thread in S is unique. Hence the abstractions give rise to a mapping ϕ_S in the sense of Section A.1.2 where

$$\phi_S = \left\{ \begin{array}{l} A_S(NRT_1), \ldots, A_S(NRT_{k_{\mathbf{N}}}), \\ [\forall_{f \in \mathbb{F}} A_S(CRT_{f,1}), \ldots, A_S(NRT_{k_{\mathbf{N},f}})], \\ A_S(IIT_1), \ldots, A_S(IIT_{k_{\mathbf{I}}}), \\ A_S(OIT_1), \ldots, A_S(OIT_{k_{\mathbf{I}}}), \\ [\forall_{E \in \mathbb{E}} A_S(EIT_{E,1}), \ldots, A_S(EIT_{\mathbf{D},E})] \end{array} \right\}$$

Lemma A.15 *(Operational Correspondence of Abstract and Concrete Integrity Management)*

1. *Whenever* $\langle \delta_S, \phi_S \rangle \xrightarrow{\alpha} \langle \delta', \phi' \rangle$, *then there exists an* $S^\sharp$ *such that* $S \Rightarrow \xrightarrow{\alpha} \sim S^\sharp$ *and* $\langle \delta', \phi' \rangle = \langle \delta_{S^\sharp}, \phi_{S^\sharp} \rangle$.

2. *Whenever* $S \xrightarrow{\alpha} S'$, *then there exists an* $S^\sharp$ *such that* $S' \sim S^\sharp$ *and* $\langle \delta_S, \phi_S \rangle \xrightarrow{\widehat{\alpha}} \langle \delta_{S^\sharp}, \phi_{S^\sharp} \rangle$.

Proof. Tedious but straightforward and without any serious difficulty. The key is that the abstract integrity manager can be regarded as a complete specification of what the (concrete) integrity manager is supposed to do: A request for registering a name n leads to a name entry with a (new) ID i and a (new) alias being looked up or inserted in the value table, whereafter i is returned to the caller; a request for building a representation of a term of the form $f(V_1, \ldots, V_{\mathrm{ar}[f]}$, where pre-existing IDs of representations of V_1, $\ldots$, $V_{\mathrm{ar}[f]}$ are given as arguments, leads to a complex value entry with a (new) ID i being looked up or inserted in the value table, whereafter i is returned to the caller; and so on. Such proofs over π-calculus representations of common data structures and algorithms are standard. Specifically, the forward operational correspondence (1) means that the integrity manager satisfies everything that the specification demands; the backward operational correspondence (2) means that the integrity manager does not do anything else. Working up to strong bisimilarity is in both cases only required to remove unused shared local names and **0**-processes. □

Corollary A.16 *(Operational Correspondence of May-Testing as Mediated by Abstract and Concrete Integrity Management) Let* P *be a derivative of any encoded* πT*-process. Then:*

1. *Whenever* $P \overset{\nu N}{\bowtie} \langle \delta_S, \phi_S \rangle \rightarrow P' \overset{\nu N'}{\bowtie} \langle \delta', \phi' \rangle$, *then there there exist an* S' *and an* $S^\sharp$ *such that* $S' \sim S^\sharp$ *and, for every set* M *of names, there exists a set* M' *of names such that* $(\nu\, M)(P \mid S) \Rightarrow (\nu\, M')(P' \mid S')$ *and* $\langle \delta', \phi' \rangle = \langle \delta_{S^\sharp}, \phi_{S^\sharp} \rangle$.

2. *Whenever* $(\nu\, M)(P \mid S) \Rightarrow (\nu\, M')(P' \mid S')$, *then there exists an* $S^\sharp$ *such that* $S' \sim S^\sharp$ *and, for every set* N *of names, there exists a set* N' *of names such that* $P \overset{\nu N}{\bowtie} \langle \delta_S, \phi_S \rangle \xrightarrow{\widehat{\tau}} P' \overset{\nu N'}{\bowtie} \langle \delta_{S^\sharp}, \phi_{S^\sharp} \rangle$.

Proof of Lemma A.14. As a follow-up corollary of Corollary A.16. □

Table 3 Clauses for transitions over abstract integrity manager state factions.

$\langle new \rangle$	$\left\langle \delta, \phi \right\rangle \xrightarrow{new(n,r)} \left\langle \delta, \phi + r : new(n) \right\rangle$
$\langle new' \rangle$	$\left\langle \delta, \phi + r : new(n) \right\rangle \xrightarrow{\tau} \left\langle \delta + \mathrm{id}[n] : \_, \phi - r + r : \overline{r}\langle \mathrm{id}[n] \rangle \right\rangle$ provided that $\delta \not\downarrow n$
$\langle new_{\mathrm{C}} \rangle$	$\left\langle \delta, \phi \right\rangle \xrightarrow{new_f(i_1,\ldots,i_{\mathrm{ar}[f]},r)} \left\langle \delta, \phi + r : new_f(i_1, \ldots, i_{\mathrm{ar}[f]}) \right\rangle$
$\langle new'_{\mathrm{C}} \rangle$	$\left\langle \delta, \phi + r : new_f(\mathrm{id}[V_1], \ldots, \mathrm{id}[V_{\mathrm{ar}[f]}]) \right\rangle \xrightarrow{\tau} \left\langle \delta + \mathrm{id}[V] : \langle f, \mathrm{id}[V_1], \ldots, \mathrm{id}[V_{\mathrm{ar}[f]}] \rangle, \phi - r + r : \overline{r}\langle \mathrm{id}[V] \rangle \right\rangle \quad V = f(V_1, \ldots, V_{\mathrm{ar}[f]})$ provided that $\delta \downarrow \mathrm{id}[V_j]$ $(j = 1, \ldots, \mathrm{ar}[f])$
$\langle input \rangle$	$\left\langle \delta, \phi \right\rangle \xrightarrow{input(i,r)} \left\langle \delta, \phi + r : input(i) \right\rangle$
$\langle output \rangle$	$\left\langle \delta, \phi \right\rangle \xrightarrow{output(i,j,r)} \left\langle \delta, \phi + r : output(i, j) \right\rangle$
$\langle \mathrm{COM} \rangle$	$\left\langle \delta, \phi + r : input(\mathrm{id}[n]) + s : output(\mathrm{id}[n], \mathrm{id}[W]) \right\rangle \xrightarrow{\tau} \left\langle \delta, \phi - r + r : \overline{r}\langle \mathrm{id}[W] \rangle - s + s : \overline{s}\langle\rangle \right\rangle$ provided that $\delta \downarrow \mathrm{id}[n]$ and $\delta \downarrow \mathrm{id}[W]$
$\langle apply \rangle$	$\left\langle \delta, \phi \right\rangle \xrightarrow{apply_E(i_1,\ldots,i_k,r)} \left\langle \delta, \phi + r : apply_E(i_1, \ldots, i_k) \right\rangle$
$\langle apply' \rangle$	$\left\langle \delta, \phi + r : apply_{d(G_1,\ldots,G_k) \triangleq x}(\mathrm{id}[V_1], \ldots, \mathrm{id}[V_k]) \right\rangle \xrightarrow{\tau} \left\langle \delta, \phi - r + r : \overline{r}\langle \mathrm{id}[x\sigma] \rangle \right\rangle$ provided that $\delta \downarrow \mathrm{id}[V_j]$ $(j = 1, \ldots, k)$ and $(G_1, \ldots, G_k)\sigma = (V_1, \ldots, V_k)$
$\langle \mathrm{RET}_1 \rangle$	$\left\langle \delta, \phi + r : \overline{r}\langle\rangle \right\rangle \xrightarrow{\overline{r}\langle\rangle} \left\langle \delta, \phi - r \right\rangle$
$\langle \mathrm{RET}_2 \rangle$	$\left\langle \delta, \phi + r : \overline{r}\langle \mathrm{id}[V] \rangle \right\rangle \xrightarrow{\overline{r}\langle \mathrm{id}[V] \rangle} \left\langle \delta, \phi - r \right\rangle$

Table 4 Clauses for transitions of abstract encoding state factions.

$$(\bowtie\text{-PAR}_1) \qquad \frac{P \xrightarrow{\tau} P'}{P \overset{\nu\, N}{\bowtie} \Sigma \to P' \overset{\nu\, N}{\bowtie} \Sigma}$$

$$(\bowtie\text{-PAR}_2) \qquad \frac{\Sigma \xrightarrow{\tau} \Sigma'}{P \overset{\nu\, N}{\bowtie} \Sigma \to P \overset{\nu\, N}{\bowtie} \Sigma'}$$

$$(\bowtie\text{-REQ}) \qquad \frac{P \xrightarrow{(\nu\, M)\,\overline{n}\langle\widetilde{m}\rangle} P' \quad \langle\delta,\phi\rangle \xrightarrow{n(\widetilde{m})} \langle\delta',\phi'\rangle \quad m \text{ fresh } (m \in M)}{P \overset{\nu\, N}{\bowtie} \langle\delta,\phi\rangle \to P' \overset{\nu\, N \cup M}{\bowtie} \langle\delta',\phi'\rangle}$$

$$(\bowtie\text{-RET}_1) \qquad \frac{P \xrightarrow{r()} P' \quad \Sigma \xrightarrow{\overline{r}\langle\rangle} \Sigma'}{P \overset{\nu\, N}{\bowtie} \Sigma \to P' \overset{\nu\, N}{\bowtie} \Sigma'}$$

$$(\bowtie\text{-RET}_2) \qquad \frac{P \xrightarrow{r(\mathrm{id}[V])} P' \quad \Sigma \xrightarrow{\overline{r}\langle\mathrm{id}[V]\rangle} \Sigma'}{P \overset{\nu\, N}{\bowtie} \Sigma \to P' \overset{\nu\, N}{\bowtie} \Sigma'}$$

Table 5 Clauses for ancestor relation relating πT-processes and (descendants of) their encodings.

($\blacktriangleleft$-**0**)
$$\mathbf{0} \blacktriangleleft \mathbf{0} \bowtie \left\langle \emptyset, \emptyset \right\rangle$$

($\blacktriangleleft$-IN)
$$\frac{\begin{array}{l}(\!|T(y).P_1|\!)\{\widetilde{x} := \mathrm{id}[\widetilde{V}]\} \bowtie \left\langle \emptyset \oplus \widetilde{V}, \emptyset \right\rangle \\ \Longrightarrow Q \overset{\nu M}{\bowtie} \Omega >_{T\{\widetilde{x}:=\widetilde{V}\}=n} \\ \Longrightarrow\sim r(x).(\!|P_1|\!)\{\widetilde{x} := \mathrm{id}[\widetilde{V}]\} \overset{\nu\{r\}}{\bowtie} \left\langle \emptyset \oplus \widetilde{V} \oplus n, \emptyset + r : \mathit{input}(\mathrm{id}[n]), \right\rangle \quad y \notin \widetilde{x} \quad r \text{ fresh}\end{array}}{(T(y).P_1)\{\widetilde{x} := \widetilde{V}\} \blacktriangleleft Q \overset{\nu M}{\bowtie} \Omega}$$

($\blacktriangleleft$-IN$'$)
$$\frac{y \notin \widetilde{x} \quad r \text{ fresh}}{P_1\{\widetilde{x}, y := \widetilde{V}, W\} \blacktriangleleft r(y).(\!|P_1|\!)\{\widetilde{x} := \mathrm{id}[\widetilde{V}]\} \overset{\nu\{r\}}{\bowtie} \left\langle \emptyset \oplus \widetilde{V} \oplus W, \emptyset + r : \overline{r}\langle \mathrm{id}[W] \rangle \right\rangle}$$

($\blacktriangleleft$-OUT)
$$\frac{\begin{array}{l}(\!|\overline{T}\langle U \rangle.P_1|\!)\{\widetilde{x} := \mathrm{id}[\widetilde{V}]\} \bowtie \left\langle \emptyset \oplus \widetilde{V}, \emptyset \right\rangle \\ \Longrightarrow Q \overset{\nu M}{\bowtie} \Omega >_{T\{\widetilde{x}:=\widetilde{V}\}=n} \\ \Longrightarrow\sim r().(\!|P_1|\!)\{\widetilde{x} := \mathrm{id}[\widetilde{V}]\} \\ \qquad \overset{\nu\{r\}}{\bowtie} \left\langle \emptyset \oplus \widetilde{V} \oplus n \oplus W, \emptyset + r : \mathit{output}(\mathrm{id}[n], \mathrm{id}[W]), \right\rangle \quad W = U\{\widetilde{x} := \widetilde{V}\} \quad r \text{ fresh}\end{array}}{(\overline{T}\langle U \rangle.P_1)\{\widetilde{x} := \widetilde{V}\} \blacktriangleleft Q \overset{\nu M}{\bowtie} \Omega}$$

($\blacktriangleleft$-OUT$'$)
$$\frac{y \notin \widetilde{x} \quad r \text{ fresh}}{P_1\{\widetilde{x} := \widetilde{V}\} \blacktriangleleft r(x).(\!|P_1|\!)\{\widetilde{x} := \mathrm{id}[\widetilde{V}]\} \overset{\nu\{r\}}{\bowtie} \left\langle \emptyset \oplus \widetilde{V}, \emptyset + r : \overline{r}\langle\rangle \right\rangle}$$

($\blacktriangleleft$-$|$)
$$\frac{P_i \blacktriangleleft Q_i \overset{\nu M_i}{\bowtie} \left\langle \epsilon_i, \psi_i \right\rangle \ (i = 1, 2) \quad \begin{array}{l}\mathrm{fn}\left[Q_1 \overset{\nu M_1}{\bowtie} \left\langle \epsilon_1, \psi_1 \right\rangle\right] \cap M_2 = \emptyset \\ M_1 \cap \mathrm{fn}\left[Q_2 \overset{\nu M_2}{\bowtie} \left\langle \epsilon_2, \psi_2 \right\rangle\right] = \emptyset\end{array} \quad M_1 \cap M_2 = \emptyset}{P_1 \mid P_2 \blacktriangleleft Q_1 \mid Q_2 \overset{\nu M_1 \cup M_2}{\bowtie} \left\langle \epsilon_1 \cup \epsilon_2, \psi_1 \cup \psi_2 \right\rangle}$$

Table 6 Clauses for ancestor relation relating πT-processes and (descendants of) their encodings.

($\blacktriangleleft$-ν_1)
$$\frac{P_1 \blacktriangleleft Q_1 \overset{\nu M}{\bowtie} \left\langle \epsilon, \psi \right\rangle \quad \psi \not\uparrow new(m) \text{ and } \epsilon \not\downarrow \mathrm{id}[m]}{(\nu\, m)\, P_1 \blacktriangleleft (\nu\, m)\, Q_1 \overset{\nu M}{\bowtie} \left\langle \epsilon, \psi \right\rangle}$$

($\blacktriangleleft$-ν_2)
$$\frac{P_1 \blacktriangleleft Q_1 \overset{\nu M_1}{\bowtie} \left\langle \epsilon, \psi \right\rangle \quad \psi\uparrow new(m) \text{ or } \epsilon\downarrow \mathrm{id}[m]}{(\nu\, m)\, P_1 \blacktriangleleft Q_1 \overset{\nu M_1+m}{\bowtie} \left\langle \epsilon, \psi \right\rangle}$$

($\blacktriangleleft$-!)
$$(!\, P_1)\{\widetilde{x} := \widetilde{V}\} \blacktriangleleft\, !(\!|P_1|\!)\{\widetilde{x} := \mathrm{id}[\widetilde{V}]\} \bowtie \left\langle \emptyset \oplus \widetilde{V}, \emptyset \right\rangle$$

($\blacktriangleleft$-**if**)
$$\frac{\begin{array}{l}(\!|\mathbf{if}\ T = U\ \mathbf{then}\ P_1\ \mathbf{else}\ P_2|\!)\{\widetilde{x} := \mathrm{id}[\widetilde{V}]\} \bowtie \left\langle \emptyset \oplus \widetilde{V}, \emptyset \right\rangle \\ \Longrightarrow Q \overset{\nu M}{\bowtie} \Omega \\ \Longrightarrow\sim \mathbf{if}\ \mathrm{id}[T\{\widetilde{x} := \widetilde{V}\}] = \mathrm{id}[U\{\widetilde{x} := \widetilde{V}\}]\ \mathbf{then}\ (\!|P_1|\!)\{\widetilde{x} := \mathrm{id}[\widetilde{V}]\}\ \mathbf{else}\ (\!|P_2|\!)\{\widetilde{x} := \mathrm{id}[\widetilde{V}]\} \\ \qquad \bowtie \left\langle \emptyset \oplus \widetilde{V} \oplus T\{\widetilde{x} := \widetilde{V}\} \oplus U\{\widetilde{x} := \widetilde{V}\}, \emptyset \right\rangle\end{array}}{(\mathbf{if}\ T = U\ \mathbf{then}\ P_1\ \mathbf{else}\ P_2)\{\widetilde{x} := \widetilde{V}\} \blacktriangleleft Q \overset{\nu M}{\bowtie} \Omega}$$

($\blacktriangleleft$-**let**)
$$\frac{\begin{array}{l}(\!|\mathbf{let}\ y = d(\widetilde{T})\ \mathbf{in}\ P_1|\!)\{\widetilde{x} := \mathrm{id}[\widetilde{V}]\} \bowtie \left\langle \emptyset \oplus \widetilde{V}, \emptyset \right\rangle \\ \Longrightarrow Q \overset{\nu M}{\bowtie} \Omega \\ \Longrightarrow\sim (\nu\, u)\ \left(\overline{u}\langle\rangle \mid \Pi_{d(\widetilde{G}) \triangleq y,\ \widetilde{G}\sigma = \widetilde{T}\{\widetilde{x} := \widetilde{V}\}} u().(\!|P_1|\!)\{\widetilde{x}, y := \mathrm{id}[\widetilde{V}], \mathrm{id}[y\sigma]\}\right) \\ \qquad \bowtie \left\langle \emptyset \oplus \widetilde{V} \oplus \widetilde{T}\{\widetilde{x} := \widetilde{V}\}, \emptyset \right\rangle\end{array}}{(\mathbf{let}\ y = d(\widetilde{T})\ \mathbf{in}\ P_1)\{\widetilde{x} := \widetilde{V}\} \blacktriangleleft Q \overset{\nu M}{\bowtie} \Omega}$$

($\blacktriangleleft$-EXT)
$$\frac{P \blacktriangleleft Q \overset{\nu M_1}{\bowtie} \left\langle \epsilon_1, \psi \right\rangle \quad M_1 \subseteq M \quad \epsilon_1 \subseteq \epsilon \quad Q \overset{\nu M}{\bowtie} \left\langle \epsilon, \psi \right\rangle \text{ well-formed}}{P \blacktriangleleft Q \overset{\nu M}{\bowtie} \left\langle \epsilon, \psi \right\rangle}$$

Recent technical reports from the Department of Information Technology

2004-043 Iordanis Kavathatzopoulos, Jenny Persson, and Carl Åborg: *Skattekontoret i Falun: Ett mönsterkontor*

2004-044 Erik Cedheim, Ramzi Ferchichi, Anders Jonsson, Dan Lind, Henrik Nyman, Olof Sivertsson, Andreas Widenfalk, Jöns Åkerlund, Leonid Mokrushin, and Paul Pettersson: *Kelb - A Real-Time Programming Environment for the Sony Aibo*

2004-045 Per Carlsson and Arne Andersson: *A Flexible Model for Tree-Structured Multi-Commodity Markets*

2004-046 Per Carlsson: *Market Simulations*

2004-047 *Iordanis Kavathatzopoulos, Jenny Öhman Persson, and *Carl Åborg: *Assessing Health and Moral Stress in IT-Based Work*

2004-048 Henrik Löf and Jarmo Rantakokko: *Algorithmic Optimizations of a Conjugate Gradient Solver on Shared Memory Architectures*

2004-049 Erik Borälv, Niklas Johansson, Emmanuel Papaioannou, and Athanasios Demiris: *A Design Case: Interactive Sports Content Broadcasting*

2004-050 Stefan Johansson: *High Order Summation by Parts Operator Based on a DRP Scheme Applied to 2D Aeroacoustics*

2004-051 Claes Olsson: *Comparative Study of Recursive Parameter Estimation Algorithms with Application to Active Vibration Isolation*

2004-052 Bharath Bhikkaji, Torsten Söderström, and Kaushik Mahata: *Recursive Algorithms for Estimating the Parameters in a One Dimensional Heat Diffusion System: Derivation and Implementation*

2004-053 Bharath Bhikkaji, Kaushik Mahata, and Torsten Söderström: *Recursive Algorithms for Estimating the Parameters in a One Dimensional Heat Diffusion System: Analysis*

2004-054 Lars Ferm, Per Lötstedt, and Paul Sjöberg: *Adaptive, Conservative Solution of the Fokker-Planck Equation in Molecular Biology*

2004-055 Per Lötstedt, Jonas Persson, Lina von Sydow, and Johan Tysk: *Space-Time Adaptive Finite Difference Method for European Multi-Asset Options*

2004-056 Erik Borälv: *Design and Evaluation of the CHILI System*

2004-057 Erik Borälv: *Evaluation and Reflections on the Design of the WeAidU System*

2004-058 Anna Eckerdal: *On the Understanding of Object and Class*

2005-001 Henrik Brandén and Per Sundqvist: *Preconditioners Based on Fundamental Solutions*

2005-002 Torbjörn Wigren: *MATLAB Software for Recursive Identification and Scaling Using a Structured Nonlinear Black-box Model — Revision 1*

2005-003 Claes Olsson: *Structure Flexibility Impacts on Robust Active Vibration Isolation Using Mixed Sensitivity Optimisation*

2005-004 Michael Baldamus, Joachim Parrow, and Björn Victor: *A Fully Abstract Encoding of the Applied π-Calculus*

UPPSALA
UNIVERSITET

February 2005, updated July 2005
ISSN 1404-3203
http://www.it.uu.se/