Static Analysis of Processes for No Read-Up and No Write-Down

Chiara Bodei, Pierpaolo Degano,[1]
Flemming Nielson, Hanne Riis Nielson[2]

[1] Dipartimento di Informatica, Università di Pisa
Corso Italia 40, I-56100 Pisa, Italy
{chiara,degano}@di.unipi.it
[2] Computer Science Department, Aarhus University
Ny Munkegade, DK-8000 Aarhus C, Denmark
{fn,hrn}@daimi.au.dk

Abstract. We study a variant of the *no read-up/no write-down* security property of Bell and LaPadula for processes in the π-calculus. Once processes are given levels of security clearance, we statically check that a process at a high level never sends names to processes at a lower level. The static check is based on a Control Flow Analysis for the π-calculus that establishes a super-set of the set of names to which a given name may be bound and of the set of names that may be sent and received along a given channel, taking into account its directionality. The static check is shown to imply the natural dynamic condition.

1 Introduction

System security is receiving more and more attention but obtaining precise answers is often undecidable [10]. However, static analysis provides a repertoire of automatic and decidable methods for analysing properties of programs, and these can often be used as the basis for establishing security properties. We use here Control Flow Analysis that is a static technique for predicting safe and computable approximations to the set of values that the objects of a program may assume during its execution. To circumvent the undecidability issues the analysis "errs on the safe side" by never omitting values that arise, but perhaps including values that never arise in the semantics. The approach is related to Data Flow Analysis and Abstract Interpretation and naturally leads to a general treatment of semantic correctness and the existence of best solutions. A more widely used alternative for calculi of computation is to use Type Systems; they also allow a clear statement of semantic correctness (sometimes called type soundness) and allow to study whether or not best solutions exist (in the form of principal types). The interplay between Type Systems and Control Flow Analysis is not yet fully understood, but simple Type Systems and simple Control Flow Analyses seem to be equally expressive; however, a main difference is that the Control Flow Analysis guarantees that best solutions always exist whereas many Type Systems do not admit principal types (and occasionally the issue is left open when presenting the type system).

Here we elaborate on our proposal made in [5], that presents a Control Flow Analysis for the π-calculus, which is a model of concurrent communicating processes based on naming, and where we applied it to statically check that a process has *no leaks*, i.e. that a process confines a set of values, devised to be secret, within itself. Our new analysis is more accurate than the one in [5], because a more careful check is made on the input and the output prefixes, so as to identify unreachable code. The result of our Control Flow Analysis establishes a super-set of the set of names to which a given name may be bound and of the sets of names that may be sent and received along a given channel, when used by a process with clearance l. These super-sets give rise to *solutions* of the form (ρ, σ) and we formulate the Control Flow Analysis as a specification of the correctness of a candidate solution. This takes the form of judgements of the form $(\rho, \sigma) \models^{l}_{me} P$ (where me will be explained later and l is the level of security clearance), and a set of clauses that operate on them. We show that best solutions always exist and we establish the semantic correctness of solutions in the form of a subject-reduction result.

We apply our analysis for statically checking a dynamic version of the *no read-up/no write-down* property of Bell and LaPadula [4, 11, 12]: a process classified at a high level cannot write any value to a process of low level, while communications in any other direction is permitted. This requirement is part of a security model, based on a multi-level access control, see [4, 10]. We first define a static check on solutions (ρ, σ), called *discreetness*, for when a process respects the classification hierarchy. Then we show that a discreet process enjoys the dynamic version of the no read-up/no write-down property.

Overview. Section 2 gives the syntax and the operational semantics of our version of the π-calculus with clearance levels. Our Control Flow Analysis is in Section 3, together with the semantic correctness of solutions. The no read-up/no write-down property is then studied in Section 4. Some proofs are omitted or only sketched because of lack of space.

2 The π-calculus

Syntax. In this section we briefly recall the π-calculus [21], a model of concurrent communicating processes based on the notion of *naming*. The formulation of our analysis requires a minor extension to the standard syntax of the π-calculus, namely assigning "channels" to the binding occurrences of names within restrictions and "binders" to the binding occurrences of names within input prefixes; as will emerge later, this is because of the α-conversion allowed by the structural congruence, and the syntactic extension will allow to compute a super-set of the actual links that a name can denote. Also, we need a further extension to assign a security level to π-calculus processes.

More precisely, we introduce a finite set $\mathcal{L} = \{\#\} \cup \{0, \cdots, k\}$ of level labels, with metavariable l, consisting both of natural numbers and of the distinguished label $\#$, intended as the label of the environment, which is intuitively assumed to have “no level”. The set $\mathcal{L}$ is ordered (see Fig. 1) with the usual $\leq$ relation on natural numbers and assuming that $\forall l \in \mathcal{L} \setminus \{\#\}, l \not\leq \#$ and $\# \not\leq l$.

Definition 1. *Let $\mathcal{N}$ be a infinite set of* names *ranged over by $a, b, \cdots, x, y$ and let τ be a distinguished element such that $\mathcal{N} \cap \{\tau\} = \emptyset$. Also let $\mathcal{B}$ be a non-empty set of* binders *ranged over by $\beta, \beta', \cdots$; and let $\mathcal{C}$ be a non-empty set of* channels *ranged over by $\chi, \chi', \cdots$; moreover let $\mathcal{B} \cup \mathcal{C}$ be the set of* markers. *Then* processes, *denoted by $P, P_1, P_2, Q, R, \cdots \in \mathcal{P}$ are built from names according to the syntax*

$$P ::= \mathbf{0} \mid \mu.P \mid P + P \mid P|P \mid (\nu x^{\chi})P \mid [x = y]P \mid !P \mid \langle P \rangle^{l}$$

where μ may either be $x(y^{\beta})$ for input, *or $\overline{x}y$ for* output *or τ for* silent *moves, and where $l \in \mathcal{L} \setminus \{\#\}$. Hereafter, the trailing $\mathbf{0}$ will be omitted (i.e. we write π instead of $\pi.\mathbf{0}$).*

In this paper we consider the *early* operational semantics defined in SOS style. The intuition of process constructors is the standard one. Indeed, $\langle P \rangle^{l}$ behaves just as P but expresses that P has level l, where $l \in \mathcal{L} \setminus \{\#\}$. The labels of transitions are τ for silent actions, xy for free input, $\overline{x}y$ for free output, and $\overline{x}(y)$ for bound output. We will use μ as a metavariable for the labels of transitions. We recall the notion of free names $fn(\mu)$, bound names $bn(\mu)$, and names $n(\mu) = fn(\mu) \cup bn(\mu)$ of a label μ. Also two partial functions, sbj and obj, are defined that give, respectively, the subject x and the object y of input and output actions, i.e. the channel x on which y is transmitted.

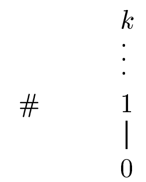

Fig. 1. Levels of processes ($i \leq i+1$) and of the environment.

Kind	μ	$fn(\mu)$	$bn(\mu)$	$sbj(\mu)$	$obj(\mu)$
Silent move	τ	$\emptyset$	$\emptyset$		
Free input and output	$xy, \overline{x}y$	$\{x, y\}$	$\emptyset$	x	y
Bound output	$\overline{x}(y)$	$\{x\}$	$\{y\}$	x	y

Functions fn, bn and n are extended in the obvious way to processes.

Congruence. Below we shall need the *structural congruence* $\equiv$ on processes, defined as in [22], apart from the last rule, where restrictions can be exchanged only if the restricted names are different, because otherwise $(\nu x^{\chi})P \equiv (\nu x^{\chi'})P$. Then $\equiv$ is the least congruence satisfying:

- if P and Q are α-equivalent ($P =_{\alpha} Q$) then $P \equiv Q$;

$$Act : \vdash^{l} \mu.P \xrightarrow[\epsilon,\epsilon]{\mu} P, \;\; \mu \neq x(y^{\beta}) \qquad Ein : \vdash^{l} x(y^{\beta}).P \xrightarrow[\lambda,\epsilon]{xw} P\{w/y\}$$

$$Par : \frac{\vdash^{l} P_1 \xrightarrow[\lambda,L]{\mu} Q_1}{\vdash^{l} P_1|P_2 \xrightarrow[\lambda,L]{\mu} Q_1|P_2}, bn(\mu) \cap fn(P_2) = \emptyset \qquad Sum : \frac{\vdash^{l} P_1 \xrightarrow[\lambda,L]{\mu} Q_1}{\vdash^{l} P_1 + P_2 \xrightarrow[\lambda,L]{\mu} Q_1}$$

$$Res : \frac{\vdash^{l} P \xrightarrow[\lambda,L]{\mu} Q}{\vdash^{l} (\nu x^{\chi})P \xrightarrow[\lambda,L]{\mu} (\nu x^{\chi})Q}, x \notin n(\mu) \qquad Open : \frac{\vdash^{l} P \xrightarrow[\epsilon,L]{\overline{x}y} Q}{\vdash^{l} (\nu y^{\chi})P \xrightarrow[\chi,L]{\overline{x}(y)} Q}, y \neq x$$

$$Close : \frac{\vdash^{l} P_1 \xrightarrow[\chi,L_1]{\overline{x}(y)} Q_1, \vdash^{l} P_2 \xrightarrow[\chi,L_2]{xy} Q_2}{\vdash^{l} P_1|P_2 \xrightarrow[\epsilon,\epsilon]{\tau} (\nu y^{\chi})(Q_1|Q_2)}, y \notin fn(P_2) \qquad Com : \frac{\vdash^{l} P_1 \xrightarrow[\epsilon,L_1]{\overline{x}y} Q_1, \vdash^{l} P_2 \xrightarrow[\epsilon,L_2]{xy} Q_2}{\vdash^{l} P_1|P_2 \xrightarrow[\epsilon,\epsilon]{\tau} Q_1|Q_2}$$

$$Match : \frac{\vdash^{l} P \xrightarrow[\lambda,L]{\mu} Q}{\vdash^{l} [x=x]P \xrightarrow[\lambda,L]{\mu} Q} \qquad Var : \frac{P' \equiv P, \vdash^{l} P \xrightarrow[\lambda,L]{\mu} Q \equiv Q'}{\vdash^{l} P' \xrightarrow[\lambda,L]{\mu} Q'}$$

$$Lev : \frac{\vdash^{l} P \xrightarrow[\lambda,L]{\mu} Q}{\vdash^{l'} \langle P \rangle^{l} \xrightarrow[\lambda,Ll]{\mu} \langle Q \rangle^{l}}$$

Table 1. Early transition system for the π-calculus with security levels.

- $(\mathcal{P}/_{\equiv}, +, \mathbf{0})$ and $(\mathcal{P}/_{\equiv}, |, \mathbf{0})$ are commutative monoids;
- $!P \equiv P|!P$.
- $(\nu x^{\chi})(\nu y^{\chi'})P \equiv (\nu y^{\chi'})(\nu x^{\chi})P$ if $x \neq y$, $(\nu x^{\chi})(P_1|P_2) \equiv (\nu x^{\chi})P_1|P_2$ if $x \notin fn(P_2)$, and $(\nu x^{\chi})P \equiv P$ if $x \notin fn(P)$;

Semantics. Table 1 shows the annotated *early* transition system of the π-calculus. The transitions have the form $\vdash^{l} P \xrightarrow[\lambda,L]{\mu} Q$, with $\lambda \in \mathcal{C} \cup \{\epsilon\}$, $l \in \mathcal{L}$, $L \in \mathcal{L}^*$. The string of level labels L records the clearances passed through while deducing the transition, while the index l on $\vdash$ represents the current level. Note that the sequence of security levels of the sender and of the receiver are discarded when a communication is derived, leading to a transition of the form $\vdash^{l} P \xrightarrow[\epsilon,\epsilon]{\tau} Q$ (see the rules *Com* and *Close* in Tab. 1). As far as the label $\lambda \in \mathcal{C} \cup \{\epsilon\}$ is concerned, we have that ϵ is used in all cases, apart from extrusions or when input transitions have to be used as a premise of a *Close* rule. In that case the label is χ and records the actual channel to be associated with the object of the input. Rule *Match* takes care of matching; we have formulated it as a transition rather than as a structural law in order to simplify the technical development (compare [5]). Rule Var ensures that all the rules and axioms can also be used upon all its variants.

3 Control Flow Analysis

The result of our analysis for a process P (with respect to an additional marker environment me for associating names with markers and a label l recording a clearance) is a pair (ρ, σ): the abstract environment ρ gives information about which channels names can be bound to, while the abstract communication environment $\sigma = \langle \sigma_{in}, \sigma_{out} \rangle$ gives information about the channels sent and received by the sub-processes of P with clearance l. Besides the usage of security levels, our present solutions refine those in [5]. In fact, the σ component controls the values that pass along a channel, more accurately than there. We now make the above more precise.

3.1 Validation

To *validate* the correctness of a proposed solution (ρ, σ) we state a set of clauses operating upon judgments of the form:

$$(\rho, \sigma) \models^{l}_{me} P$$

The purpose of me, l, ρ, σ is clarified by:

- $me : \mathcal{N} \to (\mathcal{B} \cup \mathcal{C})$ is the *marker environment* that associates names (in particular the free names of a process) with the appropriate channel or binder where the name was introduced; so $me(x)$ will be the marker (in $\mathcal{B}$ or $\mathcal{C}$) where the current name x is bound.
- $l \in \mathcal{L}$ keeps track of the current security level that the process under validation has.
- $\rho : \mathcal{B} \to \wp(\mathcal{C})$ is the *abstract environment* that associates binders with the set of channels that they can be bound to; more precisely, $\rho(\beta)$ must include the set of channels that β could evaluate to.
  By setting $\forall \chi : \rho(\chi) = \{\chi\}$ we shall allow to regard the abstract environment as a function $\rho : (\mathcal{B} \cup \mathcal{C}) \to \wp(\mathcal{C})$.
- $\sigma_{in}, \sigma_{out} : \mathcal{L} \to (\mathcal{C} \to \wp(\mathcal{C}))$ constitute the *abstract communication environment*. They give the set of the channels that can be bound to the possible objects of an input and an output action[1] respectively, performed by the sub-processes labelled by l, on a given channel χ.

---

[1] The relation between the abstract communication environment σ and the abstract channel environment κ in [5] is $\forall \chi \in \mathcal{C} : \kappa(\chi) = \bigcup_{l \in \mathcal{L}} (\sigma_{in}(l)(\chi) \cup \sigma_{out}(l)(\chi))$ in case of least solutions.

$(\rho, \sigma) \models^{l}_{me} \mathbf{0}$	iff $true$
$(\rho, \sigma) \models^{l}_{me} \tau.P$	iff $(\rho, \sigma) \models^{l}_{me} P$
$(\rho, \sigma) \models^{l}_{me} \overline{x}y.P$	iff $(\rho(me(y)) \neq \emptyset \wedge \rho(me(x)) \neq \emptyset) \Rightarrow (\rho, \sigma) \models^{l}_{me} P \wedge$ $\forall \chi \in \rho(me(x)) : \rho(me(y)) \subseteq \sigma_{out}(l)(\chi)$
$(\rho, \sigma) \models^{l}_{me} x(y^{\beta}).P$	iff $\bigcup_{l' \in \mathcal{L}, \chi \in \rho(me(x))} \sigma_{out}(l')(\chi) \neq \emptyset \Rightarrow (\rho, \sigma) \models^{l}_{me[y \mapsto \beta]} P \wedge$ $\forall \chi \in \rho(me(x)) : \bigcup_{l' \in \mathcal{L}} \sigma_{out}(l')(\chi) \subseteq \sigma_{in}(l)(\chi) \wedge$ $\forall \chi \in \rho(me(x)) : \sigma_{in}(l)(\chi) \subseteq \rho(\beta)$
$(\rho, \sigma) \models^{l}_{me} P_1 + P_2$	iff $(\rho, \sigma) \models^{l}_{me} P_1 \wedge (\rho, \sigma) \models^{l}_{me} P_2$
$(\rho, \sigma) \models^{l}_{me} P_1 \vert P_2$	iff $(\rho, \sigma) \models^{l}_{me} P_1 \wedge (\rho, \sigma) \models^{l}_{me} P_2$
$(\rho, \sigma) \models^{l}_{me} (\nu x^{\chi})P$	iff $(\rho, \sigma) \models^{l}_{me[x \mapsto \chi]} P$
$(\rho, \sigma) \models^{l}_{me} [x = y]P$	iff $(\rho(me(x)) \cap \rho(me(y)) \neq \emptyset \vee me(x) = me(y))$ $\Rightarrow (\rho, \sigma) \models^{l}_{me} P$
$(\rho, \sigma) \models^{l}_{me} !P$	iff $(\rho, \sigma) \models^{l}_{me} P$
$(\rho, \sigma) \models^{l}_{me} \langle P \rangle^{l'}$	iff $(\rho, \sigma) \models^{l'}_{me} P \wedge \sigma_{in}(l') \sqsubseteq \sigma_{in}(l) \wedge \sigma_{out}(l') \sqsubseteq \sigma_{out}(l)$

Table 2. Control flow analysis for the π-calculus.

Note that we use a marker environment because the identity of names is not preserved under α-conversions (see rules Ein and Var). In particular, it would not suffice to "α-rename the program apart" because as in [5] this property is not preserved under reduction.

The analysis is in Tab. 2. As we are analysing a process P from scratch, we assume that the initial clearance label is #. All the rules for validating a compound process require that the components are validated, apart from the rules for output, input and matching. The rules for output and input require a preliminary check to decide whether the continuation P needs to be validated and this makes the analysis more accurate than the one in [5]. In case of output, one has to make sure that the (set of channels associated with the) object is bound to some channels and similarly for the subject. In the case of input we control that the (set of channels associated with the) subject has some channels to read; actually, we ensure that some value can be sent along the subject. The last conjunct of the rule for output takes care of the clearance l of the process under analysis. The channels that can be bound to the object of an output action along channel χ must be included in $\sigma_{out}(l)(\chi)$. Analogously for the case of input, where we ensure that $\sigma_{in}(l)(\chi)$ and $\rho(\beta)$ contain all the outputs on $\chi \in \rho(me(x))$, regardless of the clearance level l of the sending process. The condition for matching says that P needs to be validated if there is at least one channel to which both x and y can evaluate; note that both can evaluate to $\emptyset$ and thus we need to check whether they actually denote the same channel by allowing $me(x) = me(y)$. The rule for the process $\langle P \rangle^{l'}$ simply says that the channels that can be read and written by it must be included in those read or written by its surrounding process, labelled l. It makes use of the following definition.

Definition 2. *The set of proposed solutions can be partially ordered by setting* $(\rho, \sigma) \sqsubseteq (\rho', \sigma')$ *iff* $\forall \beta \in \mathcal{B} : \rho(\beta) \subseteq \rho'(\beta)$, $\forall \chi \in \mathcal{C}, \forall l \in \mathcal{L} : \sigma_{in}(l)(\chi) \subseteq \sigma'_{in}(l)(\chi)$ *and* $\forall \chi \in \mathcal{C}, \forall l \in \mathcal{L} : \sigma_{out}(l)(\chi) \subseteq \sigma'_{out}(l)(\chi)$.

It is immediate that this suffices for making the set of proposed solutions into a complete lattice; using standard notation we write $(\rho, \sigma) \sqcup (\rho', \sigma')$ for the binary least upper bound (defined pointwise), $\sqcap \mathcal{I}$ for the greatest lower bound of a set $\mathcal{I}$ of proposed solutions (also defined pointwise), and $(\perp, \perp)$ for the least element (where $\perp$ maps everything[2] to $\emptyset$).

Example 1. Consider the following process

$$S = !(R \mid Q \mid P) =$$

$$!(\langle \overline{a}b.\overline{a}b.\overline{b}c \rangle^{l_R} \mid \langle a(x^{\beta_x}).\overline{x}x \rangle^{l_Q} \mid \langle a(y^{\beta_y}).y(z^{\beta_z}).([y = z]\overline{y}a \ + \ y(w^{\beta_w})) \rangle^{l_P}),$$

where the marker environment me is such that $me(fv) = \chi_{fv}$ for all the free names $fv \in \{a, b, c\}$. The pair (ρ, σ) is defined as follows, where the bound names are $bv \in \{x, y, z, w\}$ and the level labels are $l \in \{\#, l_R, l_Q, l_P\}$:

$$\rho(\beta_{bv}) = \begin{cases} \{\chi_b\} & \text{if } bv = x, y \\ \{\chi_a, \chi_b, \chi_c\} & \text{if } bv = z, w \end{cases} \qquad \text{(Recall that } \rho(\chi) = \{\chi\}\text{)}$$

$$\sigma_{in}(l)(\chi_a) = \begin{cases} \{\chi_b\} & \text{if } l = \# \\ \emptyset & \text{if } l = l_R \\ \{\chi_b\} & \text{if } l = l_Q, l_P \end{cases} \qquad \sigma_{out}(l)(\chi_a) = \begin{cases} \{\chi_b\} & \text{if } l = \#, l_R \\ \emptyset & \text{if } l = l_Q, l_P \end{cases}$$

$$\sigma_{in}(l)(\chi_b) = \begin{cases} \{\chi_a, \chi_b, \chi_c\} & \text{if } l = \# \\ \emptyset & \text{if } l = l_R, l_Q \\ \{\chi_a, \chi_b, \chi_c\} & \text{if } l = l_P \end{cases} \qquad \sigma_{out}(l)(\chi_b) = \begin{cases} \{\chi_a, \chi_b, \chi_c\} & \text{if } l = \# \\ \{\chi_c\} & \text{if } l = l_R \\ \{\chi_b\} & \text{if } l = l_Q \\ \{\chi_a\} & \text{if } l = l_P \end{cases}$$

$$\sigma_{in}(l)(\chi_c) = \emptyset, \ \text{ if } l = \#, l_R, l_Q, l_P \qquad \sigma_{out}(l)(\chi_c) = \emptyset, \ \text{ if } l = \#, l_R, l_Q, l_P$$

A simple check shows that $(\rho, \sigma) \models^{\#}_{me} S$.

3.2 Existence of solution

So far we have only considered a procedure for validating whether or not a proposed solution (ρ, σ) is in fact acceptable. We now show that there always exists a least choice of (ρ, σ) that is acceptable in the manner of Tab. 2.

[2] However, note that $\perp_\rho : \mathcal{B} \to \wp(\mathcal{C})$ viewed as $\perp_\rho : (\mathcal{B} \cup \mathcal{C}) \to \wp(\mathcal{C})$ has $\perp_\rho(\beta) = \emptyset$ for $\beta \in \mathcal{B}$ but $\perp_\rho(\chi) = \{\chi\}$ for $\chi \in \mathcal{C}$ (rather than $\perp_\rho(\chi) = \emptyset$).

Definition 3. *A set $\mathcal{I}$ of proposed solutions is a* Moore family *if and only if it contains $\sqcap\mathcal{J}$ for all $\mathcal{J} \subseteq \mathcal{I}$ (in particular $\mathcal{J} = \emptyset$ and $\mathcal{J} = \mathcal{I}$).*

This is sometimes called the model intersection property and is fundamental for many approaches to program analysis [7]. When $\mathcal{I}$ is a Moore family it contains a greatest element ($\sqcap\emptyset$) as well as a least element ($\sqcap\mathcal{I}$). The following theorem then guarantees that there always is a least solution to the specification in Tab. 2.

Theorem 1. *The set $\{(\rho, \sigma) \mid (\rho, \sigma) \models^{l}_{me} P\}$ is a Moore family for all me, l, P.*

Proof. By induction on P.

There is also a constructive procedure for obtaining the least solution; it has a low polynomial complexity. Essentially, establishing $(\rho, \sigma) \models^{l}_{me} P$ amounts to checking a number of individual constraints. In the full paper we define a function $\mathcal{G}_C[\![P]\!]_{me}$ for explicitly extracting these constraints, proceeding by induction on the structure of processes. This is not entirely straightforward because of the conditional analysis of the continuation process in the case of output, input and matching. The resulting constraints can be solved in low polynomial time.

3.3 Correctness

We state now some auxiliary results that will allow us to establish semantic correctness of our analysis. They are all independent of the semantics and only rely on Tab. 2; their proofs are all by induction.

Lemma 1. *Assume that $\forall x \in fn(P) : me_1(x) = me_2(x)$; then $(\rho, \sigma) \models^{l}_{me_1} P$ if and only if $(\rho, \sigma) \models^{l}_{me_2} P$.*

Lemma 2. *Assume that $me(y) = me(z)$; then $(\rho, \sigma) \models^{l}_{me} P$ if and only if $(\rho, \sigma) \models^{l}_{me} P\{z/y\}$.*

Corollary 1. *Assume that $z \notin fn(P)$ and $\eta \in \mathcal{B} \cup \mathcal{C}$; then $(\rho, \sigma) \models^{l}_{me[y \mapsto \eta]} P$ if and only if $(\rho, \sigma) \models^{l}_{me[z \mapsto \eta]} P\{z/y\}$.*

Lemma 3. *Assume that $P \equiv Q$; then $(\rho, \sigma) \models^{l}_{me} P$ iff $(\rho, \sigma) \models^{l}_{me} Q$.*

Lemma 4. *Assume that $(\rho, \sigma) \models^{l}_{me} P$ and $me(w) \in \rho(me(z)) \subseteq \mathcal{C}$; then $(\rho, \sigma) \models^{l}_{me} P\{w/z\}$.*

Subject reduction. To establish the semantic correctness of our analysis we rely on the definition of the early semantics in Tab. 1 as well as on the analysis in Tab. 2. The subject reduction result below applies to *all* the solutions of the analysis, and hence in particular to the least. The operational semantics only rewrites processes at "top level" where it is natural to demand that all free names are bound to channels (rather than binders); this is formalised by the condition $me[fn(-)] \subseteq \mathcal{C}$ that occurs several times. Note that item (*3b*) corresponds to "bound" input, mainly intended to be used to match a corresponding bound output in the rule *Close* of the semantics; therefore the name y read along link x must be fresh in P, i.e. $y \notin fn(P)$.

Theorem 2. *If* $me[fn(P)] \subseteq \mathcal{C}$, $(\rho, \sigma) \models^{l}_{me} P$ *and* $\vdash^{l} P \xrightarrow[\lambda, L]{\mu} Q$ *we have:*

(1) If $\mu = \tau$ *then*
$\lambda = \epsilon$, $(\rho, \sigma) \models^{l}_{me} Q$, *and* $me[fn(Q)] \subseteq \mathcal{C}$

(2a) If $\mu = \overline{x}y$ *then*
$\lambda = \epsilon$, $(\rho, \sigma) \models^{l}_{me} Q$, $me[fn(Q)] \subseteq \mathcal{C}$ *and* $\forall l' \in Ll.\, me(y) \in \sigma_{out}(l')(me(x))$

(2b) If $\mu = \overline{x}(y)$ *then*
$\lambda = \chi$ *for some* $\chi \in \mathcal{C}$, $(\rho, \sigma) \models^{l}_{me[y \mapsto \chi]} Q$, $(me[y \mapsto \chi])[fn(Q)] \subseteq \mathcal{C}$, *and* $\forall l' \in Ll.\, \chi \in \sigma_{out}(l')(me(x))$

(3a) If $\mu = xy$, $\lambda = \epsilon$, $me(y) \in \mathcal{C}$ *and*
$me(y) \in \bigcup_{l' \in \mathcal{L}} (\sigma_{in}(l')(me(x)) \cup \sigma_{out}(l')(me(x)))$ *then*
$(\rho, \sigma) \models^{l}_{me} Q$, $me[fn(Q)] \subseteq \mathcal{C}$, *and* $\forall l' \in Ll.\, me(y) \in \sigma_{in}(l')(me(x))$

(3b) If $\mu = xy$, $\lambda = \chi$, $\chi \in \bigcup_{l' \in \mathcal{L}} (\sigma_{in}(l')(me(x)) \cup \sigma_{out}(l')(me(x)))$ *and* $y \notin fn(P)$ *then*
$(\rho, \sigma) \models^{l}_{me[y \mapsto \chi]} Q$, $(me[y \mapsto \chi])[fn(Q)] \subseteq \mathcal{C}$, *and*
$\forall l' \in Ll.\, \chi \in \sigma_{in}(l')(me(x))$

Proof. A lengthy proof by induction on the construction of

$$\vdash^{l} P \xrightarrow[\lambda, L]{\mu} Q$$

and with subcases depending on whether case *(1)*, *(2a)*, *(2b)*, *(3a)* or *(3b)* applies. The proof makes use of Lemmata 1, 2, 3 and 4.

4 Multi-level Security

System security is typically based on putting objects and subjects into security classes and preventing information from flowing from higher levels to lower ones. Besides the no-leaks property studied in [5], here we offer another evidence that Control Flow Analysis helps in statically detecting useful information on security.

The literature reports a security property called no read-up/no write-down [11, 12]. The security requirement is that a process classified at a high level cannot write any value to a process of low level, while the converse is allowed. These requirements are part of a security model, based on a multi-level access control, see [4, 10]. The no read-up/no write-down property is commonly studied for a set of processes put in parallel (see, e.g. [29]). We follow this view and consider in the following only processes of the form $\langle P_0 \rangle^{l_0} | \langle P_1 \rangle^{l_1} | \ldots | \langle P_n \rangle^{l_n}$, where each process P_i has no labelling construct inside.

A dynamic notion. Now we are ready to introduce the dynamic version of the no read-up/no write-down property.

We assume that the environment is always willing to listen to P, i.e. its sub-processes at any level can perform free outputs to the environment. On the contrary, some parts of P are reachable only if the environment supplies some information to a sub-process R with a particular clearance. To formalize the intentions of the environment, we use a function

$$\varsigma : \mathcal{L} \to (\mathcal{C} \to \wp(\mathcal{C}))$$

that associates a label l and a channel χ with the set of channels that the environment considers secure to communicate to R.

Definition 4. *Given P, me_P, ς, a* granted step *is $(P, me_P) \xrightarrow[\lambda, L]{\mu} (Q, me_Q)$, and is defined whenever*

1. $\vdash^{\#} P \xrightarrow[\lambda, L]{\mu} Q$, *and*
2. *if* $\mu = xy$, *then* $\begin{cases} (a)\ me_P(y) \in \varsigma(\#)(me_P(x)) & \text{if } \lambda = \epsilon \\ (b)\ \lambda \in \varsigma(\#)(me_P(x)) \text{ and } y \notin fn(P) & \text{if } \lambda = \chi \end{cases}$

where $me_Q = \begin{cases} me_P & \text{if } \lambda = \epsilon \\ me_P[obj(\mu) \mapsto \chi] & \text{if } \lambda = \chi \end{cases}$

A granted computation $(P, me_P) \Longrightarrow^* (Q, me_Q)$ *is made of granted steps.*

The definition of our version of the no read-up/no write-down property follows. Essentially, it requires that in all the communications performed by a process, the sender R_o has a clearance level lower than the clearance level of the receiver R_i.

Definition 5. *A process P is* no read-up/no write-down *(*nru/nwd *for short) with respect to ς, me_P if and only if the following holds:*

whenever $(P, me_P) \Longrightarrow^* (P', me_{P'}) \xrightarrow[\epsilon, L]{\tau} (Q, me_Q)$ where the last granted step is a communication (between R_o and R_i) that has been deduced with either

(a) the rule Com, using the premises $\vdash^l R_o \xrightarrow[\epsilon, L_o]{\overline{x}y} R'_o$ and $\vdash^l R_i \xrightarrow[\epsilon, L_i]{xy} R'_i$, or

(b) the rule Close, using the premises $\vdash^l R_o \xrightarrow[\chi, L_o]{\overline{x}(y)} R'_o$ and $\vdash^l R_i \xrightarrow[\chi, L_i]{xy} R'_i$,

then no element of L_o is strictly greater than any element of L_i.

A static notion. We define now a static property that guarantees that a process is nru/nwd. Besides finding a solution (ρ, σ) for a process P, we require that the channels read along χ should include those that the environment is willing to supply, expressed by ς. The last condition below requires that the same channel cannot be used for sending an object from a process with high level l'' to a process with low level l'.

Definition 6. *Let P, me be such that $me[fn(P)] \subseteq \mathcal{C}$. Then P is* discreet *(w.r.t. ς, me) if and only if there exists (ρ, σ) such that*

1. $(\rho, \sigma) \models^{\#}_{me} P$
2. $\forall l \in \mathcal{L}, \chi \in \mathcal{C} : \sigma_{in}(l)(\chi) \supseteq \varsigma(l)(\chi)$
3. $\forall l', l'' \in \mathcal{L},\ l' < l''$ *and* $\forall \chi \in \mathcal{C} : \sigma_{out}(l'')(\chi) \cap \sigma_{in}(l')(\chi) = \emptyset$.

The property of discreetness can be checked in low polynomial time by building on the techniques mentioned in Section 3.2. Below, we show that the property of being discreet is preserved under granted steps.

Lemma 5 (Subject reduction for discreetness).

If P is discreet with respect to ς, me_P, and $(P, me_P) \overset{\mu}{\underset{\lambda, L}{\Longrightarrow}} (Q, me_Q)$, then Q is discreet with respect to ς, me_Q.

Proof. Theorem 2 suffices for proving that $me_Q[fn(Q)] \subseteq \mathcal{C}$ and $(\rho, \sigma) \models^{\#}_{me} Q$. The proof of the second and third items is immediate, because the solution does not change. The only delicate point for the application of Theorem 2 is when the granted step is an input. Consider first, the case in which $\lambda = \epsilon$. It suffices to make sure that $me(y) \in \mathcal{C}$ and that $me(y) \in \sigma_{in}(\#)(me(x))$. Condition 2 of Def. 6 guarantees that $\sigma_{in}(\#)(me(x)) \supseteq \varsigma(\#)(me(x))$. In turn $me(y) \in \varsigma(\#)(me(x)) \subseteq \mathcal{C}$ because the step is granted. The case when $\lambda = \chi$ is just the same, while in the other cases the proof is trivial.

The following lemma further illustrates the links between the transitions of processes and the results of our static analysis. It will be used in proving that discreetness is sufficient to guarantee that P enjoys the nru/nwd property.

Lemma 6. *If $\vdash^{l} P \xrightarrow[\lambda, L]{\mu} Q$ has been deduced with premise $\vdash^{l'} R \xrightarrow[\lambda', L']{\mu} R'$, and $(\rho, \sigma) \models^{l}_{me} P$, then there exists me' such that $(\rho, \sigma) \models^{l'}_{me'} R$.*

Proof. The proof is by induction on the derivation of the transition.

Theorem 3.
If P is discreet (w.r.t. ς, me), then P is nru/nwd (w.r.t. ς, me).

Proof. By Lemma 5 it is enough to check that, if $(P, me_P) \overset{\tau}{\underset{\epsilon, L}{\Longrightarrow}} (Q, me_Q)$, then Q is nru/nwd, with τ being a communication between R_o and R_i, defined as in Def. 5. Assume, per absurdum, that an element $l_o \in L_o$ is strictly greater than an element element $l_i \in L_i$. By Lemma 6, $(\rho, \sigma) \models^{l}_{me} R_o$ and $(\rho, \sigma) \models^{l}_{me} R_i$. Consider first the case (a) of Def. 5. The analysis and Theorem 2 tell us that $\forall l' \in L_o.\ me(y) \in \sigma_{out}(l')(me(x))$ and $\forall l' \in L_i.\ me(y) \in \sigma_{in}(l')(me(x))$. But this contradicts item 3 of Def. 6, because, in particular, $me(y) \in \sigma_{out}(l_o)(me(x))$ as well as $me(y) \in \sigma_{in}(l_i)(me(x))$.
As for case (b) of Def. 5, just replace χ for $me(y)$ and proceed as in case (a).

Example 2. Consider again the process S validated in Example 1:

$$!(\langle \overline{a}b.\overline{a}b.\overline{b}c\rangle^{l_R} \mid \langle a(x^{\beta_x}).\overline{x}x\rangle^{l_Q} \mid \langle a(y^{\beta_y}).y(z^{\beta_z}).([y=z]\overline{y}a \ + \ y(w^{\beta_w}))\rangle^{l_P}),$$

and suppose that $l_R < l_Q < l_P$. Then it is easy to prove that the process is discreet. In particular the following five conditions hold.

- $\sigma_{out}(l_Q)(\chi_{fv}) \cap \sigma_{in}(l_R)(\chi_{fv}) = \sigma_{out}(l_Q)(\chi_{fv}) \cap \emptyset = \emptyset$;
- $\sigma_{out}(l_P)(\chi_{fv}) \cap \sigma_{in}(l_R)(\chi_{fv}) = \sigma_{out}(l_P)(\chi_{fv}) \cap \emptyset = \emptyset$;
- $\sigma_{out}(l_P)(\chi_a) \cap \sigma_{in}(l_Q)(\chi_a) = \emptyset \cap \{\chi_b\} = \emptyset$.
- $\sigma_{out}(l_P)(\chi_b) \cap \sigma_{in}(l_Q)(\chi_b) = \{\chi_a\} \cap \emptyset = \emptyset$.
- $\sigma_{out}(l_P)(\chi_c) \cap \sigma_{in}(l_Q)(\chi_c) = \emptyset \cap \emptyset = \emptyset$.

Note that the clearance levels of processes introduced here are orthogonal to the security levels of channels as defined in [5]. There channels are partitioned into secret and public and a static check is made that secret channels never pass along a public one. Therefore channels have always the same level. On the contrary, here it is possible that a channel a can be sent along b by one process but *not* by another. Discreetness cannot be checked with the analysis of [5], because in that analysis a can be either sent on b always or never. The combination of the two analyses may permit a static check of even more demanding properties.

5 Conclusions

There is a vast literature on the topics of our paper. Here we only mention very briefly some papers related to security issues.

The first studies in system security reach back to the 1970's and were mainly carried out in the area of operating systems; see the detailed survey by Landwehr [17] and Denning's book [10] reporting on the static detection of secure flow violation while analysing the code.

Recently, security classes have been formalized as types and the control of flow is based on type checking. Heintze and Riecke [15] study a non-interference property on the SLam Calculus (Secure λ-calculus). Volpano, Smith and Irvine develop a type system to ensure secure information flow in a sequential imperative language in [30], later extended by the first two authors in a concurrent, shared memory based setting [29]. Abadi studies in [1] the secrecy of channels and of encrypted messages, using the spi-calculus, an extension of the π-calculus devised for writing secure protocols. Venet [27, 28] uses Abstract Interpretation techniques to analyse processes in a fragment of the π-calculus, with particular attention to the usage of channels.

Other papers interesting for this area are [24, 26, 13, 8, 3, 9, 25, 16]. Particularly relevant are Hennessy and Riely's papers [25, 16] who give a type system for Dπ, a variant of the π-calculus with explicit sites that harbour mobile processes. Cardelli and Gordon [6] propose a type system for the Mobile Ambient calculus ensuring that a well-typed mobile computation cannot cause certain kinds of run-time faults, even when capabilities can be exchanged between processes.

The idea of static analysis for security has been followed also in the Java world, for example in the Java Bytecode Verifier [18], and in the techniques of proof-carrying code [23]. Also Abadi faces in [2] the problem of implementing secure systems and proposes to use full abstraction to check that the compile code enjoys the same security properties as the source program.

A different approach consists in dynamically checking properties. This point of view has been adopted by a certain number of information flow models [14, 19, 20, 11, 12] (to cite only a few), mainly concerned with checking (variants of) the security property we studied here. All these papers, consider the external observable behaviour only as the object of the analysis.

Here, we presented a Control Flow Analysis for the π-calculus that statically predicts how names will be bound to actual channels at run-time. The only extensions made to the syntax of processes are that a channel χ is explicitly assigned to a restricted name, and that an input action has the form $x(y^\beta)$, making explicit the rôle of the placeholder y; this change was motivated by the inclusion of α-conversion in the semantics. Our intention was to apply our analysis for detecting violations of a security property that needs security levels, so processes may carry labels expressing their clearance. The result of our analysis for a process P is a solution (ρ, σ). The abstract environment ρ gives information about which channels a binder β may be bound to, by means of communication. The abstract communication environment σ gives information about the channels sent and received by a process with clearance l. All the solution components approximate the actual solution, because they may give a super-set of the corresponding actual values.

We defined judgements of the form $(\rho, \sigma) \models^l_{me} P$ and a set of clauses that operate on them so as to validate the correctness of the solution. The additional marker environment me binds the free names of P to actual channels. The label l records the security level of P. We proved that a best solution always exists. In the full paper we shall give a constructive procedure for generating solutions that essentially generates a set of constraints corresponding to the checks necessary to validate solutions. These constraints can be solved in low polynomial time.

We used our analysis to establish the no read-up/no write-down security property of Bell and LaPadula. This property requires that a process with high clearance level never sends channels to processes with a low clearance. We defined a static check on solutions and proved that it implies the no read-up/no write-down property. Also, the check that a process P is discreet with respect to given ς,me has a polynomial time complexity. A web-based system that validates the solutions and checks discreetness can be found at the URL: `http://www.daimi.au.dk./~rrh/discreet.html`.

We have not considered here the more general notion of the no read-up/no write-down property, that assigns levels of confidentiality also to the exchanged data (i.e. the objects of input and output actions). Processes with low level clearance are then not allowed to access (i.e. they can neither send nor receive) highly classified data. The reason is that the dynamic version of this property is surprisingly more intricate than its static version. The latter, that entails the former, only requires an additional check on the second component of a solution (i.e., $\forall \chi \in \sigma_{in}(l)(\chi')$ the confidentiality level of χ, possibly read along channel χ', should be smaller than the security level of the process under check, namely l; similarly for σ_{out}).

Other properties that deserve further investigation are connected with the so-called "indirect flow" of information, i.e. on the possibility of a low level process to detect the value of some confidential datum by "observing" the behaviour of higher level processes.

Acknowledgments. We wish to thank Martín Abadi, Roberto Gorrieri and Dennis Volpano for interesting discussions, and René Rydhof Hansen for writing the web-based system that checks discreetness.

The first two authors have been partially supported by the CNR Progetto Strategico *Modelli e Metodi per la Matematica e l'Ingegneria* and by the MURST Progetto *Tecniche Formali per la Specifica, l'Analisi, la Verifica, la Sintesi e la Trasformazione di Sistemi Software*. The last two authors have been partially supported by the DART project (funded by The Danish Research Councils).

References

1. M. Abadi. Secrecy by typing in security protocols. In *Proceedings of Theoretical Aspects of Computer Software, Third International Symposioum, LNCS 1281*, pages 611–638. Springer-Verlag, 1997.
2. M. Abadi. Protection in programming-language translations. In *Proceedings of ICALP'98, LNCS 1443*, pages 868–883. Springer-Verlag, 1998.
3. R.M. Amadio. An asynchronous model of locality, failure, and process mobility. In *Proceedings of COORDINATION'97, LNCS 1282*, pages 374–391. Springer-Verlag, 1997.
4. D.E. Bell and L.J. LaPadula. Secure computer systems: Unified exposition and Multics interpretation. Technical Report ESD-TR-75-306, Mitre C., 1976.
5. C. Bodei, P. Degano, F. Nielson, and H. Riis Nielson. Control flow analysis for the π-calculus. In *Proceedings of CONCUR'98, LNCS 1466*, pages 84–98. Springer-Verlag, 1998.
6. L. Cardelli and A.D. Gordon. Types for mobile ambients. In *Proceedings of POPL'99*, page (to appear). ACM Press, 1999.
7. P. Cousot and R. Cousot. Systematic Design of Program Analysis Frameworks. In *Proceedings of POPL '79*, pages 269–282. ACM Press, 1979.
8. M. Dam. Proving trust in systems of second-order processes: Preliminary results. Technical Report LOMAPS-SICS 19, SICS, Sweden, 1997.
9. R. De Nicola, G. Ferrari, and R. Pugliese. Coordinating mobile agents via blackboards and access rights. In *Proceedings of COORDINATION'97, LNCS 1282*, pages 220–237. Springer-Verlag, 1997.
10. D.E Denning. *Cryptography and Data Security.* Addison-Wesley, Reading Mass., 1982.
11. R. Focardi. Comparing two information flow security properties. In *Proceedings of 9^{th} IEEE Computer Science Security Foundation W/S*, pages 116–122, 1996.

12. R. Focardi and R. Gorrieri. The compositional security checker: A tool for the verification of information flow security properties. *IEEE Transactions on Software Engineering*. To appear.
13. C. Fournet, C. Laneve, L. Maranget, and D. Remy. Implicit typing à la ML for the join calculus. In *Proceedings of CONCUR'97, LNCS 1243*, pages 196–212. Springer-Verlag, 1997.
14. J.A. Goguen and J. Meseguer. Security policy and security models. In *Proceedings of the 1982 IEEE Symposioum on Research on Security and Privacy*, pages 11–20. IEEE Press, 1982.
15. N. Heintze and J.G Riecke. The SLam calculus: Programming with secrecy and integrity. In *Proceedings of POPL'98*, pages 365–377. ACM Press, 1998.
16. M. Hennessy and J. Riely. Resource access control in systems of mobile agents. Technical Report 2/98, University of Sussex, 1998.
17. C. E. Landwehr. Formal models for computer security. *ACM Computing Surveys*, 13(3):247–278, 1981.
18. T. Lindholm and F. Yellin. *The Java Virtual Machine Specification*. Addison-Wesley, 1996.
19. D. McCullough. Specifications for multi-level security and hook-up property. In *Proceedings of the 1987 IEEE Symposioum on Research on Security and Privacy*. IEEE Press, 1987.
20. D. McCullough. Noninterference and the composability of security properties. In *Proceedings of the 1988 IEEE Symposium on Research on Security and Privacy*, pages 177–186. IEEE Press, 1988.
21. R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes (I and II). *Information and Computation*, 100(1):1–77, 1992.
22. R. Milner, J. Parrow, and D. Walker. Modal logics for mobile processes. *TCS*, 114:149–171, 1993.
23. G. Necula. Proof-carrying code. In *Proceedings of POPL'97*, pages 106–119. ACM Press, 1997.
24. B.C. Pierce and D. Sangiorgi. Typing and sub-typing for mobile processes. *Mathematical Structures in Computer Science*, 6(5):409–454, 1996.
25. J. Riely and M. Hennessy. A typed language for distributed mobile processes. In *Proceedings of POPL'98*, pages 378–390. ACM Press, 1998.
26. P. Sewell. Global/local subtyping and capability inference for a distributed π-calculus. In *Proceedings of ICALP'98, LNCS 1443*, pages 695–706. Springer-Verlag, 1998.
27. A. Venet. Abstract interpretation of the π-calculus. In *Analysis and Verification of Multiple-Agent Languages, LNCS 1192*, pages 51–75. Springer-Verlag, 1997.
28. A. Venet. Automatic determination of communication topologies in mobile systems. In *Static Analysis Symposium, LNCS 1503*, pages 152–167. Springer-Verlag, 1998.
29. D. Volpano and G. Smith. Secure information flow in a multi-threaded imperative language. In *Proceedings of POPL'98*, pages 355–364. ACM Press, 1998.
30. D. Volpano, G. Smith, and C. Irvine. A sound type system for secure flow analysis. *Journal of Computer Security*, 4:4–21, 1996.