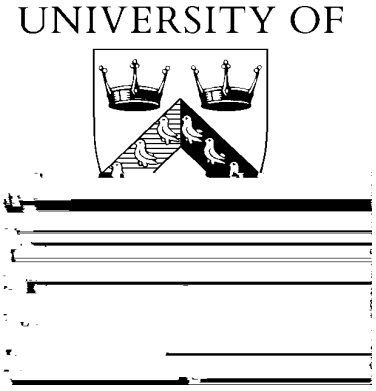
UNIVERSITY OF SUSSEX

COMPUTER SCIENCE



The Security Picalculus and Non-interference

Matthew Hennessy

Report 05/2000 revised November 2000

Computer Science
School of Cognitive and Computing Sciences
University of Sussex
Brighton BN1 9QH

ISSN 1350–3170

independent server, together with a private return channel r, generated specifically for this purpose. A response is awaited from the service, on the reply channel r, which is then forwarded on the original return channel y.

Numerous typing

respect to Γ, interacting with P. After an interaction on the channel $\mathsf{r}\ \mathsf{q}$ the process evolves to P_1:

$$(\mathsf{n}\ \ \mathsf{w}\, r : \mathrm{R})\ \ \mathsf{s}!\langle v, r\rangle \mid r?(z)\, \mathsf{b}!(z)\,,$$

for some value v and channel b sent by the observer. At this stage both the observer and the observed process P_1 can still be typed relative to Γ, as both v and b must have been known to the observer, and therefore be typeable in Γ. However now the observed process generates a new channel r, with type $\mathrm{R} = \{\mathsf{r}\langle\mathbf{int}\rangle, \mathsf{w}\langle\mathbf{int}\rangle\}$. But because of the type associated with s in Γ, r is only sent to the observer with the subtype consisting of the one capability $\mathsf{w}\langle\mathbf{int}\rangle$. Subsequently the observer 984 0 Td( ) 97 Td[(resp

which indicates that in addition P uses channels with security level *at most* σ. Similarly we have a typing relation

$$\Gamma \models^{\sigma} P$$

indicating that P uses channels with *at least* security level σ. Indeed we can go further, designing relations such as $\Gamma \vdash_{r\sigma} P$ or $\Gamma \models^{w\sigma} P$ where the *read capabilities* or the *write capabilities* of processes are independently constrained. For all of these typing relations Subject Reduction is easily established.

Section 3 is the heart of the paper. First the behavioural preorders and equivalences are defined, by adapting the standard framework, [14, 9], to the security π-calculus. We obtain the relations

$$\Gamma \rhd_{\sigma} P \simeq_{may}$$

and

$$\Gamma \rhd_{\sigma} P \simeq_{must}$$

indicating that P and can not be distinguished, relative to *may/must* experiments respectively, by any testing process T such that $\Gamma \vdash_{\sigma} T$, that is any test running at security level *at most* σ, relative to the type environment. This is followed by an exposition of the Context LTS, actions in context, and their properties. Sub-section 3.3 then contains an alternative characterisation of $\simeq_{may}$ in terms of sequences of actions in context, while in Subsection 3.4 we give the much more complicated characterisation of $\simeq_{must}$.

One benefit of having behavioural equivalences relativised to security levels is that non-interference results can be stated succinctly. Section 4 contains two such statements, and their proofs. The first gives conditions ensure that

$$\Gamma \rhd_{\sigma} P \simeq_{may} \quad \text{implies } \Gamma \rhd_{\sigma} P \mid H \simeq_{may} \quad \mid K.$$

It turns out to be sufficient to require that the *read capabilities* of P and be bounded above by σ, that is $\Gamma \vdash_{r\sigma} P,$, and that the *write capabilities* of H and K be bounded below by some $\delta \not\preceq \sigma$, that is $\Gamma \models^{w\delta} H, K$.

This is quite a general non-interference result. For example in the case where is P and K is the empty process $\mathbf{0}$ we obtain

$$\Gamma \rhd_{\sigma} P \simeq_{may} P \mid H$$

indicating that, under the conditions of the theorem, the process H can not interfere with the behaviour of P.

This result is not true for the *must* equivalence. As explained in Section 4, this is because our types allow contention between processes run-

ning

be the least sets, and $<:$, *consistent* the least relations, which satisfy:

$$\frac{}{\mathbf{B}_\rho \in \mathit{Type}_\sigma}\ \rho \preceq \sigma \quad \text{(RT-BASE)}$$

$$\frac{\mathrm{A} \in \mathit{Type}_\sigma}{\mathsf{w}_\sigma\langle \mathrm{A}\rangle \in \mathit{Cap}_\sigma}\ \text{(RT-WR)} \qquad \frac{\mathrm{A} \in \mathit{Type}_\rho}{\mathsf{r}_\rho\langle \mathrm{A}\rangle \in \mathit{Cap}_\sigma}\ \sigma \preceq \rho\ \text{(RT-RD)}$$

$$\frac{S \subseteq_{fin} \mathit{Cap}_\sigma}{S \in \mathit{Type}_\sigma}\ S \text{ consistent}\ \text{(RT-WRRD)} \qquad \frac{\mathrm{A}_i \in \mathit{Type}_\sigma \quad (\forall i)}{(\mathrm{A}_1, \ldots, \mathrm{A}_k) \in \mathit{Type}_\sigma}\ \text{(RT-TUP)}$$

(U-WR)	$\mathsf{w}_\sigma\langle \mathrm{A}\rangle$	$<:\ \mathsf{w}_\sigma\langle \mathrm{B}\rangle$	if $\mathrm{B} <: \mathrm{A}$
(U-RD)	$\mathsf{r}_\sigma\langle \mathrm{A}\rangle$	$<:\ \mathsf{r}_\sigma\langle \mathrm{B}\rangle$	if $\mathrm{A} <: \mathrm{B}$
(U-BASE)	$\mathbf{B}_\sigma$	$<:\ \mathbf{B}_\rho$	if $\sigma \preceq \rho$
(U-RES)	$\{cap_i\}_{i\in I}$	$<:\ \{cap'_j\}_{j\in J}$	if $(\forall j)(\exists i)\ cap_i <: cap'_j$
(U-TUP)	$(\mathrm{A}_1, \ldots, \mathrm{A}_k)$	$<:\ (\mathrm{B}_1, \ldots, \mathrm{B}_k)$	if $(\forall i)\ \mathrm{A}_i <: \mathrm{B}_i$

The set of capabilities Cap is *consistent* if

- $\mathsf{w}_\sigma\langle \mathrm{A}\rangle,\ \mathsf{w}_\rho\langle \mathrm{B}\rangle \in \mathrm{Cap}$ implies $\sigma = \rho$ and A is B
- $\mathsf{r}_\sigma\langle \mathrm{A}\rangle,\ \mathsf{r}_\sigma\langle \mathrm{B}\rangle \in \mathrm{Cap}$ implies A is B
- $\mathsf{w}_\sigma\langle \mathrm{A}\rangle,\ \mathsf{r}_\rho\langle \mathrm{B}\rangle \in \mathrm{Cap}$ implies $\mathrm{A} <: \mathrm{B}$.

These types correspond very closely to the *I-types* of [10]; the rule (RT-RD) ensures that only *write-ups* are allowed, from low-level processes to high-level processes. But we allow multiple read capabilities, which will enable us to be more flexible with respect to allowing/disallowing reading from a channel at different security levels. However subtyping is more restrictive; unlike [10] they can only be sub-typed at the same security level; $\mathsf{r}_\sigma\langle \mathrm{A}\rangle <: \mathsf{r}_\rho\langle \mathrm{B}\rangle$ only if $\sigma = \rho$. Nevertheless this is compensated for in the existence of multiple read capabilities.

Example 2.2.

- The set $\{\mathsf{w}_{\mathsf{bot}}\langle \mathbf{int}\rangle, \mathsf{r}_{\mathsf{bot}}\langle \mathbf{int}\rangle, \mathsf{r}_{\mathsf{top}}\langle \mathbf{int}\rangle\}$ is a bot-level channel type, an element of $\mathit{Type}_{\mathsf{bot}}$; that is channels of this type may be transmitted on bot-level channels. In turn these channels may be written to by a bot-level process or read by either a bot-level or a top-level process.
- The type $\{\mathsf{w}_{\mathsf{bot}}\langle \mathbf{int}\rangle, \mathsf{r}_{\mathsf{top}}\langle \mathbf{int}\rangle\}$ restricts reading from the channel to top-level processes, although bot-level ones can write to it.

FIGURE 3 Typing Rules

$$\frac{\Gamma(u) <: \mathrm{A}}{\Gamma \vdash u : \mathrm{A}}\ (\text{T-ID}) \qquad \frac{bv \in \mathbf{B}_\sigma}{\Gamma \vdash bv : \mathbf{B}_\sigma}\ (\text{T-BASE}) \qquad \frac{\Gamma \vdash v_i : A_i \quad (\forall i)}{\Gamma \vdash (v_1, \ldots, v_k) : (\mathrm{A}_1, \ldots, \mathrm{A}_k)}\ (\text{T-TUP})$$

$$\frac{\Gamma, X : \mathrm{A} \vdash P \quad \Gamma \vdash u : \mathsf{r}_\sigma\langle \mathrm{A} \rangle}{\Gamma \vdash u?(X : \mathrm{A})\ P}\ (\text{T-IN}) \qquad \frac{\Gamma \vdash u : \mathsf{w}_\sigma\langle \mathrm{A} \rangle \quad \Gamma \vdash v : \mathrm{A}}{\Gamma \vdash u!\langle v \rangle}\ (\text{T-OUT})$$

$$\frac{\Gamma \vdash u : \mathrm{A}, v : \mathrm{B} \quad \Gamma \vdash \quad \Gamma \sqcap \{u : \mathrm{B}, v : \mathrm{A}\} \vdash P}{\Gamma \vdash \mathsf{if}\ u = v\ \mathsf{th}\ \mathsf{n}\ P\ \ \mathsf{ls}}\ (\text{T-EQ})$$

$$\frac{\Gamma, a : \mathrm{A} \vdash P}{\Gamma \vdash (\mathsf{n}\ \mathsf{w}\, a : \mathrm{A})\ P}\ (\text{T-NEW}) \qquad \frac{\Gamma \vdash P, \quad}{\Gamma \vdash P \mid \quad , *P, \mathbf{0}}\ (\text{T-STR})$$

$\Gamma <: \Delta$ if for all u in the domain of Δ, $\Gamma(u) <: \Delta(u)$. We will normally abbreviate the simple environment $\{u : \mathrm{A}\}$ to $u : \mathrm{A}$ and moreover use $v : \mathrm{A}$ to denote its obvious generalisation to values; this is only well-defined when the value v has the same structure as the type A.

The first typing system is given in Figure 3, where the judgements take the form

$$\Gamma \vdash P$$

Intuitively this means that the process P uses all channels as input/output devices in accordance with their types, as given in Γ. It is the standard typing system for the π-calculus, [16], adapted to our types; note that the security levels on the capabilities do not play any role.

We can also design a type inference system which not only ensures that channels are used according to their types but also controls the security levels of the channels used. One such system is given in Figure 4, where the judgements now take the form

$$\Gamma \vdash_\sigma P$$

This indicates that not only is P well-typed as before but in addition it uses channels with security level *at most* σ. (This corresponds to the typing system used in [10].) The only difference is in the input/output rules, where the security level of the channels used are checked. For example $\Gamma \vdash_\sigma a!\langle v \rangle$ only if in Γ the channel a can be assigned a security level $\delta \preceq \sigma$, in addition to having the appropriate output capability in Γ.

We can also design a typing system

$$\Gamma \Vdash^\sigma P$$

$a?(X:\mathrm{B})\,R$, the move is $a?(X)\,R \xrightarrow{a?v} R\{\!\{{}^{v}\!/_{X}\}\!\}$ and $\Delta \vdash_{\sigma} P$. From the typing rules we have $\Delta \vdash a : \mathsf{r}_{\delta}\langle \mathrm{B} \rangle$ for some $\delta \preceq \sigma$ and $\Delta, X:\mathrm{B} \vdash_{\sigma} R$. From the former we know that there exists some $\mathrm{A} <: \mathrm{B}$ such that $\mathsf{r}_{\delta}\langle \mathrm{A} \rangle \in \Delta(a)$; from the latter, and Subsumption, we have $\Delta, X:\mathrm{A} \vdash_{\sigma} R$. A standard Substitution Lemma can now be applied for any v such that $\Delta \sqcap v:\mathrm{A}$ is well-defined to obtain $\Delta \sqcap v:\mathrm{A} \vdash_{\sigma} R\{\!\{{}^{v}\!/_{X}\}\!\}$.

□

3 Behavioural Theories

In this section we develop two behavioural theories of typed processes, based on the general testing theories of [14, 9]. In the first section we adapt the original definitions from [14, 9] to our language. This is followed by a subsection defining the Context LTS alluded to in the Introduction. Two further subsections use this LTS to determine the *may* and *must* versions of our behavioural equivalence.

3.1 Testing Processes

A *test* or *observer* is a process with an occurrence of a new reserved resource name ω, used to report success. We let T to range over tests, with the typing rule $\Gamma \vdash_{\sigma} \omega!\langle\rangle$ for all Γ. When placed in parallel with a process P, a test may interact with P, producing an output on ω if some desired behaviour of P has been observed. We write

$$P \textbf{ may } T$$

$T \mid P \xrightarrow{\tau}{}^{*} R$ for some R such that R can *report success*, i.e. $R \xrightarrow{\omega!\langle\rangle}$. The stronger relation

$$P \textbf{ must } T$$

holds when in every computation

$$T \mid P \xrightarrow{\tau} R_1 \xrightarrow{\tau} \ldots \xrightarrow{\tau} R_n \xrightarrow{\tau} \ldots$$

there is some R_k, $k \geq 0$, which can report success.

We can obtain a testing based behavioural preorder between processes by demanding that they react in a similar manner to a given class of tests. Here we choose the class of tests which are well-typed and use channels from *at most* a given security level σ; that is we require that processes react in the same manner to all tests T such that $\Gamma \vdash_{\sigma} T$.

Definition 3.1 (Testing Preorders). We write $\Gamma \rhd_{\sigma} P \sqsubseteq_{may}$ if for every test *finite* T such that $\Gamma \vdash_{\sigma} T$, P **may** T implies **may** T.

Similarly $\Gamma \rhd_\sigma P \sqsubseteq_{must}$ means that for every such T, P **must** T implies **must** T.

We use $\simeq_{may}$ and $\simeq_{must}$ denote the related equivalence relations.

So for example setting σ to be bot, $\Gamma \rhd_{\mathsf{bot}} P \simeq_{may}$ means that in the type environment Γ, P and are indistinguishable by low-level observers, from a may testing point of view.

For technical reasons we have limited tests to be *finite*, that contain no occurrence of the recursive operator $*$. It is well-known (see [9]) that this does not lead to any less distinguishing power.

3.2 The Context Labelled Transition System

It is well-known, [14, 9], that testing equivalences are closely related to the ability of processes to perform sequences of actions. We have explained in the Introduction that here we need to relativise these sequences to security levels and to a pair of typing environments, one for the observer and one for the process being observed.

The rules for the Context LTS, are given in Figure 5. The judgements take the form

$$\Gamma; \Delta \rhd P \xrightarrow{\mu}_\sigma \Gamma'; \Delta' \rhd P'$$

This judgement should be

process $a!\langle v\rangle$ at any type B such that $\Gamma \vdash a : \mathsf{r}_\delta\langle \mathrm{B}\rangle$, where $\delta \preceq \sigma$. However to eliminate much potential nondeterminism in the LTS our rule dictates that for a given $\delta \preceq \sigma$ the observer receives v at the *minimum* B such that $\Gamma \vdash a : \mathsf{r}_\delta\langle \mathrm{B}\rangle$; this is the type A such that $\mathsf{r}_\delta\langle \mathrm{A}\rangle \in \Gamma(a)$.

Note that in the output actions we do not record the types of the bound names. These we only required in Figure 2 in order to implement communication between processes; see the rule (L-COM). Here we do not need to formalise, at least directly, communication between the process P and its observer.
We can describe precisely the form these judgements in can take:

LEMMA 3.2. *Suppose* $\Gamma; \Delta \triangleright P \xrightarrow{\mu}_\sigma \Gamma'; \Delta' \triangleright P'$.

$\mu = \tau$: *Here* $\Gamma' = \Gamma$ *and* $\Delta' = \Delta$.
$\mu = a?v$: *Here* $\Gamma' = \Gamma$ *while* $\Delta' = \Delta \sqcap v : \mathrm{A}$ *for some type* A *such that* $\Gamma \vdash v : \mathrm{B}$, $a : \mathsf{w}_\delta\langle \mathrm{B}\rangle$, *for some* $\delta \preceq \sigma$ *and* $\mathrm{B} <: \mathrm{A}$
$\mu = (\tilde{c})a!v$: *Here* $\Delta' = \Delta, \tilde{c} : \tilde{C}$ *for some sequence of types* $\tilde{C}$ *such that* $\Delta, \tilde{c} : \tilde{C} \vdash v : \mathrm{A}$, *while* $\Gamma' = \Gamma \sqcap v : \mathrm{A}$ *for some* A *such that* $\mathsf{r}_\delta\langle \mathrm{A}\rangle \in \Gamma(a)$, *where* $\delta \preceq \sigma$.

Proof. Straightforward rule induction on $\Gamma; \Delta \triangleright P \xrightarrow{\mu}_\sigma \Gamma'; \Delta' \triangleright P'$. □

However we are only interested in a subset of the possible judgements which can be derived from the rules in Figure 5. We say that the two type environments Γ and Δ are *compatible* if

- $\Gamma \sqcap \Delta$ exists
- $\mathrm{domain}(\Delta) \subseteq \mathrm{domain}(\Gamma)$.

The main property of this relation is given by:

LEMMA 3.3. *Suppose* Γ *and* Δ *are compatible. Then* $\Gamma \vdash a : \mathsf{w}_\rho\langle \mathrm{A}\rangle$ *and* $\Delta \vdash a : \mathsf{r}_{\rho'}\langle \mathrm{A}'\rangle$ *imply* $\mathrm{A} <: \mathrm{A}'$ *and* $\rho \preceq \rho'$.

Proof. Simple calculation. □

The triple $\Gamma; \Delta \triangleright P$ is said to be a *configuration* if

- Γ and Δ are compatible
- $\Delta \vdash P$.

When this is the case we will refer to the judgment $\Gamma; \Delta \triangleright P \xrightarrow{\mu}_\sigma \Gamma'; \Delta' \triangleright P'$ as an *action in context*.

Configurations are preserved by these actions:

LEMMA 3.4. *If* $\Gamma;\Delta \triangleright P \xrightarrow{\mu}_{\sigma} \Gamma';\Delta' \triangleright P'$ *is an* action in context *then* $\Gamma';\Delta' \triangleright P'$ *is a configuration.*

Proof. From Lemma 3.2 we know exactly the form Γ' and Δ' can take, depending on μ. In each case it is straightforward to show that they are compatible. The simplest way to show that $\Delta' \vdash P'$ is to use rule induction on $\Gamma;\Delta \triangleright P \xrightarrow{\mu}_{\sigma} \Gamma';\Delta' \triangleright P'$. □

In future we will limit our attention to judgements $\Gamma;\Delta \triangleright P \xrightarrow{\mu}_{\sigma} \Gamma';\Delta' \triangleright P'$, which are *actions in context.* This has important consequences, in the case when μ is an output action $(\tilde{c}:\tilde{C})a!v$. It means that the only new names gained by the observer, that is names in the domain of Γ' which are not in that of Γ, are $\tilde{c}$. In other words if w is an identifier in v which does not occur in $\tilde{c}$ the observer already knows about it. However the action may increase the type at which the observer knows w. It is also worth noting that the two rules (C-IN) and (C-OUT) are apriori partial; that is (C-IN) can only be applied if $\Delta \sqcap v:\mathrm{A}$ is well-defined while (C-OUT) requires $\Gamma \sqcap v:\mathrm{A}$ to be well-defined. However it is easy to show that for *actions in context* these environments are in fact well-defined whenever the corresponding premises hold. Moreover in (C-IN) the side-condition $\mathrm{B} <: \mathrm{A}$ may be omitted as it is always satisfied.

We can also determine the circumstances under which the unconstrained actions, from Figure 5, can give rise to *actions in context.*

LEMMA 3.5. *Suppose* $P \xrightarrow{\mu}$ *and let* $\Gamma;\Delta \triangleright P$ *be a configuration.*

$\mu = \tau$: *Here* $\Gamma;\Delta \triangleright P \xrightarrow{\tau}_{\sigma} \Gamma;\Delta \triangleright$

$\mu = a?v$: *Here if* $\Gamma \vdash v:\mathrm{B}$, $a:\mathsf{w}_{\delta}\langle\mathrm{B}\rangle$, *where* $\delta \preceq \sigma$ *then* $\Gamma;\Delta \triangleright P \xrightarrow{a?v}_{\sigma}$ $\Gamma;\Delta \sqcap v:\mathrm{A} \triangleright$ *for some* A *such that* $\mathrm{B} <: \mathrm{A}$.

$\mu = (\tilde{c}:\tilde{C})a!v$: *Here if* *vules*

- Suppose $P \xrightarrow{\mu}$ is inferred using (L-OPEN), that is

$$(\mathsf{n}\ \mathsf{w}\, b:\mathrm{B})\ P' \xrightarrow{(b:\mathrm{B})(\tilde{c}:\tilde{\mathrm{C}})a!v}$$

because $P' \xrightarrow{(\tilde{c}:\tilde{\mathrm{C}})a!v}$.

$\Delta \vdash P$ implies $\Delta, b:\mathrm{B} \vdash P'$ and the existence of of $\Gamma \sqcap \Delta$ also ensures that of $\Gamma \sqcap \Delta, b:\mathrm{B}$. In short the (weaker) inductive hypothesis holds of $\Gamma; \Delta, b:\mathrm{B} \triangleright P'$ and therefore by induction we can obtain the action in context $\Gamma; \Delta, b:\mathrm{B} \triangleright P' \xrightarrow{(\tilde{c})a!v}_{\sigma}$. An application of (C-OPEN) gives the required $\Gamma; \Delta \triangleright P \xrightarrow{(b)(\tilde{c})a!v}_{\sigma}$

□

Note that in *actions in context* $\Gamma; \Delta \triangleright P \xrightarrow{\mu}_{\sigma} \Gamma'; \Delta' \triangleright$ the resulting environments, $\Delta'; \Gamma'$, are not in general determined by Γ and Δ. The change in the environment of the observed process, the change from Δ to Δ', is determined by the declared types of new names introduced by the process. For example consider

$$P_1 = (\mathsf{n}\ \mathsf{w}\, c:\mathrm{C}_1)\ a!$$

$s = (\tilde{c})a!v \cdot s'$*:* Here Γ *after*$_\sigma$ s is only defined if $\mathsf{r}_\delta\langle \mathrm{A}\rangle \in \Gamma(a)$ for some $\delta \preceq \sigma$, in which case it is $(\Gamma \sqcap v : \mathrm{A})$ *after*$_\sigma$ s'.

LEMMA 3.6. *If* $\Gamma; \Delta \triangleright P \xrightarrow{s}_\sigma \Gamma'; \Delta' \triangleright \quad$ *, where* Γ *is a single-level environment, then* Γ after$_\sigma$ s *is defined and* $\Gamma' = \Gamma$ after$_\sigma$ s.

Proof. By induction on the derivation of $\Gamma; \Delta \triangleright P \xrightarrow{s}_\sigma \Gamma'; \Delta' \triangleright \quad$. □

3.3 May testing

In this section we give a characterisation of the relation $\Gamma \triangleright_\sigma P \sqsubseteq_{may} \quad$.

Actions in context are generalised to (asynchronous) *traces in context* as follows:

DEFINITION 3.7 (TRACES). Let $\Gamma; \Delta \triangleright P \stackrel{s}{\Longrightarrow}_\sigma \Gamma'; \Delta' \triangleright P'$ be the least relation such that:

$$\frac{\Gamma; \Delta \triangleright P \xrightarrow{\tau}_\sigma \Gamma'; \Delta' \triangleright P' \qquad \Gamma'; \Delta' \triangleright P' \stackrel{s}{\Longrightarrow}_\sigma \Gamma''; \Delta'' \triangleright P''}{\Gamma; \Delta \triangleright P \stackrel{s}{\Longrightarrow}_\sigma \Gamma''; \Delta'' \triangleright P''}\ (\text{TR-}\tau) \qquad \frac{}{\Gamma; \Delta \triangleright P \stackrel{\epsilon}{\Longrightarrow}_\sigma \Gamma; \Delta \triangleright P}\ (\text{TR-}\epsilon)$$

$$\frac{\Gamma; \Delta \triangleright P \xrightarrow{\alpha}_\sigma \Gamma'; \Delta' \triangleright P' \qquad \Gamma'; \Delta' \triangleright P' \stackrel{s}{\Longrightarrow}_\sigma \Gamma''; \Delta'' \triangleright P''}{\Gamma; \Delta \triangleright P \stackrel{\alpha \cdot s}{\Longrightarrow}_\sigma \Gamma''; \Delta'' \triangleright P''}\ (\text{TR-}\alpha)$$

$$\frac{\Gamma \vdash v : \mathrm{A} \qquad \Gamma; \Delta \sqcap v : \mathrm{A} \sqcap a : \mathsf{w}_\delta\langle \mathrm{A}\rangle \triangleright P \mid a!\langle v\rangle \stackrel{s}{\Longrightarrow}_\sigma \Gamma''; \Delta'' \triangleright P''}{\Gamma; \Delta \triangleright P \stackrel{a?v \cdot s}{\Longrightarrow}_\sigma \Gamma''; \Delta'' \triangleright P''}\ \delta \preceq \sigma\ (\text{TR-ASYNC})$$

Note that there is some redundancy here. The rule (TR-α), where α is an input action $a?v$, can actually be derived from (TR-ASYNC) and (TR-τ).

We now show how interactions between a process P and a σ-level observer T, that is a computation from $T \mid P$, can be decomposed into a *trace in context* from P and the complementary sequence from T. It will become clear that it is sufficient to only consider $\mathsf{n}\ \mathsf{w}\mathit{free}$ observers, that is observers which contain no occurrence of the binders $(\mathsf{n}\ \mathsf{w}\, a)$.

THEOREM 3.8 (TRACE DECOMPOSITION). *Let* $\Gamma;\Delta\triangleright P$ *be a configuration and suppose* $T \mid P \xrightarrow{\tau}{}^{*} R$ *for some* $\mathsf{n}\ \mathsf{w} f$*ree observer* T *such that* $\Gamma \vdash_{\sigma} T$. *Then there exists a* trace in context

$$\Gamma;\Delta \triangleright P \stackrel{s}{\Longrightarrow}_{\sigma} \Gamma';\Delta' \triangleright P'$$

and a derivation $T \stackrel{\overline{s}}{\Longrightarrow} T'$, *where* R *has the form* $(\mathsf{n}\ \mathsf{w}\,\tilde{c}:\tilde{C})\ (T' \mid P')$.

Proof. By induction on the derivation of $T \mid P \xrightarrow{\tau}{}^{*} R$. Consider the non-trivial case when this is of the form $T \mid P \xrightarrow{\tau}\xrightarrow{\tau}{}^{*} R$. There are essentially three cases:

- Output from T to P. In this case we have $T \xrightarrow{a!v} T_1$, $P \xrightarrow{a?v} P_1$ and $T_1 \mid P_1 \xrightarrow{\tau}{}^{*} R$.

  $\Gamma \vdash_{\sigma} T$ means $\Gamma \vdash v:\mathrm{B}$, $a:\mathsf{w}_{\delta}\langle \mathrm{B}\rangle$, for some $\delta \preceq \sigma$ and B, and so we may apply Lemma 3.5 to obtain the action in context

  $$\Gamma;\Delta \triangleright P \xrightarrow{a?v}_{\sigma} \Gamma;\Delta \sqcap v:\mathrm{A} \triangleright P_1$$

  for some $\mathrm{B} <: \mathrm{A}$. Moreover the compatibility of Γ and $\Delta \sqcap v:\mathrm{A}$ follows from that of Γ and Δ.

  Subject Reduction implies that $\Gamma \vdash_{\sigma} T_1$ and therefore we may apply induction to obtain

  $$\Gamma;\Delta \triangleright P_1 \stackrel{s'}{\Longrightarrow}_{\sigma} \Gamma';\Delta' \triangleright P' \text{ and } T_1 \stackrel{\overline{s'}}{\Longrightarrow} T'$$

  where R has the form $(\mathsf{n}\ \mathsf{w}$

THEOREM 3.9 (TRACE COMPOSITION). *Suppose* $\Gamma;\Delta\triangleright P \stackrel{s}{\Longrightarrow}_\sigma \Gamma';\Delta'\triangleright P'$ *and* $T \stackrel{\overline{s}}{\Longrightarrow} T'$ *for some* s. *Then there exists a derivation* $T \mid P \stackrel{\tau}{\longrightarrow}^* R$, *where* R *has the form* $(\mathsf{n}\ \mathsf{w}\,\tilde{c}:\tilde{C})\ (T' \mid P')$.

Proof. By induction on s. □

Refering to the statement of this theorem note that Subject Reduction ensures that $\Delta' \vdash P'$. However in general we do *not* have that $\Gamma' \models_\sigma T'$, even under the assumption $\Gamma \models_\sigma T$.

EXAMPLE 3.10. Let P, T be the processes $(\mathsf{n}\ \mathsf{w}\,c:\mathrm{C})\ a!\langle c\rangle$ and $a?(x:\mathrm{A}_2)\ x!\langle\rangle$ respectively and let Γ,Δ map a to the type $\{\mathsf{r}_{\delta_1}\langle \mathrm{A}_1\rangle,\ \mathsf{r}_{\delta_2}\langle \mathrm{A}_2\rangle,\ \mathsf{w}_{\mathsf{bot}}\langle \mathrm{C}\rangle\}$, where $\mathrm{A}_1,\ \mathrm{A}_2,\ \mathrm{C}$ are the types $\mathsf{r}_{\mathsf{bot}}\langle\rangle$, $\mathsf{w}_{\mathsf{bot}}\langle\rangle$, $\{\mathrm{A}_1,\mathrm{A}_2\}$ respectively; here we assume $\delta_i \preceq \sigma$. Then

$$\Gamma \models_\sigma T$$
$$\Delta \vdash P$$
$$\Gamma;\Delta \triangleright P \stackrel{(c)a!c}{\longrightarrow}_\sigma \Gamma, c:\mathrm{A}_1;\Delta' \triangleright \mathbf{0}$$
$$T \stackrel{a?c}{\longrightarrow} c!\langle\rangle$$

but $\Gamma, c:\mathrm{A}_1 \not\models_\sigma c!\langle\rangle$.

The problem lies, again, with the use of multi-level types.

LEMMA 3.11. *Let* Γ *be a single-level environment. Suppose* $\Gamma \models_\sigma T$ *and* $\Gamma\ \mathrm{after}_\sigma\ s$ *is defined. Then* $T \stackrel{\overline{s}}{\Longrightarrow} T'$ *implies* $\Gamma\ \mathrm{after}_\sigma\ s \models_\sigma T'$.

Proof. By induction on s. □

This Lemma may now be applied to the conditions of the Trace Composition Theorem, Theorem 3.9, to ensure when Γ is a single-level environment we can also conclude that $\Gamma' \models_\sigma T'$; here Γ' can only be Γ *after*$_\sigma$ s.

We may now state a sufficient condition to ensure two processes are related with respect to *may* testing.

DEFINITION 3.12. For any configuration $\mathcal{C}$ let $Aseq^\sigma(\mathcal{C}) = \{\, s \mid \mathcal{C} \stackrel{s}{\Longrightarrow}_\sigma \,\}$ Then we write

$$\Gamma \triangleright_\sigma (\Delta \vdash P) \ll_{may} (\Delta' \vdash\ \ ).$$

if for every appropriate Γ', $Aseq^\sigma(\Gamma,\Gamma';\Delta\triangleright P) \subseteq Aseq^\sigma(\Gamma,\Gamma';\Delta'\triangleright\ \ )$

Notice that in this definition we allow the testing environment, Γ, to be increased via Γ'; this enables tests to generate new names to send to the processes under observation.

PROPOSITION 3.13. *Suppose* $\Delta \vdash P$, , *where* Γ *and* Δ *are compatible. Then* $\Gamma \rhd_\sigma (\Delta \vdash P) \ll_{may} (\Delta' \vdash)$ *implies* $\Gamma \rhd_\sigma P \sqsubseteq_{may}$.

Proof.

Suppose $\Gamma \rhd_\sigma P \sqsubseteq_{may}$ and P **may** T, where $\Gamma \vdash T$; we must show **may** T.

Notice that the Trace Decomposition Theorem, Theorem 3.8, is only valid for $\mathsf{n}\ \mathsf{w} f\mathit{ree}$ processes and T may in fact contain occurrences of $(\mathsf{n}\ \mathsf{w}\, n)$, intuitively generating new names with which to test the processes. However, because we only employ finite tests, it is easy to show that

$$T \equiv_{st} (\mathsf{n}\ \mathsf{w}\, \tilde{c} : \tilde{C})\ T'$$

for some $\mathsf{n}\ \mathsf{w} f\mathit{ree}$ test T', where $\equiv_{st}$ is the structural congruence generated by the equations:

$$\begin{array}{rl}
P \mid (\mathsf{n}\ \mathsf{w}\, a) & \equiv_{st} (\mathsf{n}\ \mathsf{w}\, a)\ (P \mid \ ) \ \text{ if } a \notin \mathsf{fn}(P) \\
\text{if } u = v \text{ th } \mathsf{n}\ (\mathsf{n}\ \mathsf{w}\, a)\ P \ \mathsf{ls} & \equiv_{st} (\mathsf{n}\ \mathsf{w}\, a)\ (\text{if } u = v \text{ th } \mathsf{n}\ P \ \mathsf{ls} \quad) \\
 & \qquad \text{if } a \notin \mathsf{fn}(\ ), a \neq u, v \\
u?(x)\,(\mathsf{n}\ \mathsf{w}\, a)\ P & \equiv_{st} (\mathsf{n}\ \mathsf{w}\, a)\ (u?(x)\, P) \ \text{ if } a \neq u \\
P \mid & \equiv_{st} \ \mid P
\end{array}$$

(We have omitted two obvious symmetric rules for *Cap* and input, respectively.) Moreover it is possible to show that $\equiv_{st}$ is preserved by reduction, $\xrightarrow{\tau}$, form which it follows that for any process S, S **may** T if and only S **may** T'. So it is sufficient to prove **may** T'.

Since P **may** T' we know there exists a computation $T' \mid P \xrightarrow{\tau}{}^* R$, where R can report a success. For convenience let Γ' denote $\Gamma, \tilde{c} : \tilde{C}$, an extension of Γ. Because $\Gamma'; \Delta \rhd P$ is a configuration Theorem 3.8 can be used to obtain the decomposition into a *trace in context*

$$\Gamma'; \Delta \rhd P \stackrel{s}{\Longrightarrow}_\sigma \Gamma'; \Delta' \rhd P'$$

and a sequence $T' \stackrel{\overline{s}}{\Longrightarrow} T''$, where R has the form $(\mathsf{n}\ \mathsf{w}\, \tilde{d} : \tilde{D})\ (T'' \mid P')$.

Since $Aseq^\sigma(\Gamma'; \Delta \rhd P)$

the property that P **may** $T(\Gamma, s, \sigma)$ if and only if there is some Δ such that $\Gamma; \Delta \triangleright P \stackrel{s}{\Longrightarrow}$. Note Δ will not be used in the definition and the tests will only be defined for certain combinations of Γ and s.

For convenience we only consider traces in which only simple identifiers are output, rather than vectors; that is the output actions are of the form $a!v$ or $(c)a!c$. The generalisation to general output actions of the form $(\tilde{c}:\tilde{C})a!v$ is very straightforward, but notationally complex. The definition of $T(\Gamma, s, \sigma)$ is by induction on s.

ε: $T(\Gamma, \varepsilon, \sigma)$ is $\omega!\langle\rangle$.

$a?v \cdot s$: In this case the test is defined only if

- there exists some $\delta \preceq \sigma$ such that $\Gamma \vdash a : \mathsf{w}_\delta\langle \mathrm{A}\rangle$ for some type A such that $\Gamma \vdash v : \mathrm{A}$
- $T(\Gamma, s, \sigma)$ is defined.

If this is the case then $T(\Gamma, a?v \cdot s, \sigma)$ is defined to be

$$a!\langle v\rangle \mid T(\Gamma, s, \sigma).$$

$a!v \cdot s$: Here the test is defined if

- $v \in \mathrm{domain}(\Gamma)$
- there exists some type A such that $\mathsf{r}_\delta\langle \mathrm{A}\rangle \in \Gamma(a)$ for some $\delta \preceq \sigma$
- $T(\Gamma \sqcap v : \mathrm{A}, s, \sigma)$ is defined.

For each such A let $T_{\mathrm{A}}(\Gamma, a!v \cdot s, \sigma)$ be the test

$$a?(x:\mathrm{A})\,\mathsf{if}\ x = v\ \mathsf{th}\ \mathsf{n}\ T(\Gamma \sqcap v : \mathrm{A}, s, \sigma)\ \ \mathsf{ls}\ \ \mathbf{0}$$

Then the required test is

$$T_{\mathrm{A}_1}2\mathrm{Tf}7.080080\mathrm{Td}(\mathrm{s};)\mathrm{Tj}13.08010\mathrm{Td}()\mathrm{Tj}/\mathrm{R}690.12\mathrm{Tf}8.639840\mathrm{Td}())\mathrm{T}_{\mathrm{j}}$$

where A

DEFINITION 3.18 (CONVERGENCE). We say the configuration $\mathcal{C}$ *converges*, written $\mathcal{C} \Downarrow$, if there is no infinite sequence of derivations

$$\mathcal{C} \xrightarrow{\tau} \mathcal{C}_1 \xrightarrow{\tau} \ldots \xrightarrow{\tau} \mathcal{C}_k \xrightarrow{\tau}$$

This relation is then parameterised to *sequences in context*, security levels and finite multisets of input actions, by

ε: $\mathcal{C} \Downarrow^I_\sigma$ if $(\mathcal{C} \mid \overline{I}) \Downarrow$
$s = (\tilde{c})a!v \cdot s'$: $\mathcal{C} \Downarrow^I_\sigma s$ if $\mathcal{C} \Downarrow$ and whenever $\mathcal{C} \xRightarrow{(\tilde{c})a!u}_\sigma \mathcal{C}'$, $\mathcal{C}' \Downarrow^I_\sigma s'$.
$s = a?v \cdot s'$: $\mathcal{C} \Downarrow^I_\sigma s$ if, assuming $\mathcal{C}$ has the form $\Gamma; \Delta \triangleright P$,

- Δ a[illegible]

We use $I^\sigma(\mathcal{C})$ to denote the set of input actions which the configuration $\mathcal{C}$ can perform at level σ, $\{\, a?v \mid \mathcal{C} \xrightarrow{a?v}_\sigma \,\}$. More generally we use I to denote an arbitrary *multi-set* of input actions,

Proof. (Outline) The proof follows the outline of that of Lemma 4.4.13 of [9], although the details are more complicated because of asynchrony and the use of type environments and security levels.

Let T be an arbitrary $\mathsf{n}\ \mathsf{w} f r e e$ test such that $\Gamma \vdash_{\sigma} T$ and suppose P **must** T. We show **must** T. To keep the argument simple we assume T is $\mathsf{n}\ \mathsf{w} f r e e$; the more general case is handled exactly as in Proposition 3.13.

Let

$$T \mid \quad (\equiv T_0) \mid \quad {}_0 \xrightarrow{\tau} \mathcal{C}_1 \xrightarrow{\tau} \ldots\ldots \mathcal{C}_k \xrightarrow{\tau} \ldots \qquad (\dagger)$$

be an arbitrary maximal computation from $T \mid$, where we may assume each $\mathcal{C}_k$ has the form $(\mathsf{n}\ \mathsf{w}\, \tilde{c_k} : \tilde{C_k})\ (T_k \mid \quad {}_k)$. We must show that for some k, $T_k \xrightarrow{\omega !\langle\rangle}$.

First suppose that the computation $(\dagger)$ is finite, ending in $\mathcal{C}_n$. Using Trace Decomposition it can decomposed into

$$\begin{array}{c} \Gamma; \Delta \triangleright \quad \stackrel{s}{\Longrightarrow} \Gamma; \Delta \triangleright \quad {}_n \\ T \stackrel{\bar{s}}{\Longrightarrow} T_n \end{array}$$

From Lemma 2.5 we can assume T_n has the form $a_1!\langle v_1\rangle \mid \ldots a_k!\langle v_k\rangle \mid T'$, where T' cannot perform any output moves. Let I denote the multi-set of input actions, $\{a_1?v_1, \ldots, a_k?v_k\}$ and D the acceptance set determined by the configuration $\mathcal{C}_n$output4use59382.15977otev

plying Lemma 3.11 it follows that $\Gamma_m \models_\sigma a!\langle v\rangle$, which by Lemma 3.5 is sufficient to ensure that $\Gamma_m; \Delta_m \triangleright P_m \xrightarrow{a?v}_\sigma$. This contradicts (2) above.

Output: Here we have $P_m \xrightarrow{(c)a!\langle v\rangle}$ and $T' \xrightarrow{a?v}$.

Again from Lemma 3.11 we know $\Gamma_m \models_\sigma T'$ and therefore $a! \in O \subseteq D$; so $_n \xrightarrow{(\tilde{c})a!w}$ for some value w. Because of the structure of our language, $T' \xrightarrow{a?v}$ implies that $T' \xrightarrow{a?w}$ is also true, and therefore we have a contradiction of the maximality of $\mathcal{C}_n$.

This completes the proof, under the assumptions that $\Gamma; \Delta \triangleright P \Downarrow^I_\sigma s$ and the computation under scrutiny, (†), is finite. However these assumptions can be taken care of in the standard manner, as in the proof of Lemma 4.4.13 of [9].

□

As in the case of *may testing* the proof of the converse depends on the ability to define well-typed tests which determine the relation $\ll^\sigma$. Here there are two possible reasons why configurations may not be related; one associated with convergence, the other with a mismatch of acceptance sets. We treat each in turn. As in the previous sub-section to avoid notational complexity we only consider simple output actions, where only single names are transmitted. We also use some of the derived notation developed in that sub-section.

Tests for Convergence. We define the terms $T_C(\Gamma, s, I, \sigma)$ by induction on s:

ε: Here $T_C(\Gamma, s, I, \sigma) = (\omega!\langle\rangle \oplus \omega!\langle\rangle) \mid \overline{I}$

$a!v \cdot s'$: Here $T_C(\Gamma, s, I, \sigma)$ is given by

$$(\mathsf{n}\ \mathsf{w}\, n)\ n!\langle\rangle \mid n?()\,\omega!\langle\rangle \mid a?(x:\mathrm{A})\ \ \mathsf{if}\ x = v\ \mathsf{th}\ \mathsf{n}\ n?()\,T_C(\Gamma \sqcap v:\mathrm{A}, s', I, \sigma)\ \mathsf{ls}\ \mathbf{0}$$

where $\mathsf{r}_\delta\langle\mathrm{A}\rangle \in \Gamma(a)$ for some $\delta \preceq \sigma$

$(c)a!c \cdot s'$: In this case $T_C(\Gamma, s, I, \sigma)$ is given by

$$(\mathsf{n}\ \mathsf{w}\, n)\ n!\langle\rangle \mid n?()\,\omega!\langle\rangle \mid a?(x:\mathrm{A})\ \ \mathsf{if}\ x \in I(\Gamma, \mathrm{A})\ \mathsf{th}\ \mathsf{n}\ \mathbf{0}\ \mathsf{ls}\ (n?()\,T_C(\Gamma, c:\mathrm{A}, s', I, \sigma))\{^x/_c\}$$

where again $\mathsf{r}_\delta\langle\mathrm{A}\rangle \in \Gamma(a)$ for some $\delta \preceq \sigma$

$a?v \cdot s'$: Here $T_C(\Gamma, s, I, \sigma)$ is only defined if $\Gamma \vdash a : \mathsf{w}_\delta\langle\mathrm{A}\rangle, v : \mathrm{A}$ for some

$\delta \preceq \sigma$, in which case it is

$$a!\langle v\rangle \mid T_C(\Gamma, s', \sigma)$$

We leave the reader to check the following:

LEMMA 3.24. *Suppose* $\Gamma;\Delta\triangleright \quad \stackrel{s}{\Longrightarrow}_\sigma \Gamma';\Delta'\triangleright \quad '$, *where* $(\quad '\,|\,\overline{I})\not\Downarrow$, *for some* I *such that* $(\Gamma \text{ after}_\sigma\ s)$ $\text{allows}_\sigma\ I$. *Then*

- $T_C(\Gamma, s, I, \sigma)$ *is defined*
- $\Gamma \vdash_\sigma T_C(\Gamma, s, I, \sigma)$
- $\not\!\!\mathbf{must}\ T_C(\Gamma, s, I, \sigma)$.

Proof. By induction on s. □

COROLLARY 3.25. $\Gamma\triangleright_\sigma P \sqsubseteq_{must} \quad$ *and* $\Gamma;\Delta\triangleright P \Downarrow^I_\sigma s$ *implies* $\Gamma;\Delta\triangleright \quad \Downarrow^I_\sigma s$.

Proof. Suppose, on the contrary, that for some s, $\Gamma;\Delta \triangleright P \Downarrow^I_\sigma s$, while $\Gamma;\Delta\triangleright \quad \stackrel{s}{\Longrightarrow}_\sigma \Gamma';\Delta'\triangleright \quad$, for some $\quad '$ such that $(\quad '\,|\,\overline{I})\not\Downarrow$. By the previous Lemma it is sufficient to show P **must** $T_C(\Gamma, s, I, \sigma)$, which can easily be done by induction on s. □

Tests for Acceptance Sets. Let us first extend the predicate $allows_\sigma$ to apply to output acceptance sets, in addition to sets of input actions. We write Γ *allows*$_\sigma$ O if, for each $a! \in O$, $\mathsf{r}_\delta\langle \mathrm{A}\rangle \in \Gamma(a)$ for some $\delta \preceq \sigma$, and $\Gamma \vdash v : \mathrm{A}$ for some value v; note that this means $\Gamma \vdash_\sigma a!\langle v\rangle$.

We now define terms $T(\Gamma, s, O, I, \sigma)$, where O is an output acceptance set and I is a set of input actions, by induction on s. The inductive cases are very similar to the corresponding cases in the definition of the tests for convergence.

ε: Here $T(\Gamma, s, O, I, \sigma)$ is only defined if Γ *allows*$_\sigma$ O, I, in which case it is

$$\prod\{\, a!\langle v\rangle \mid a?v \in I \,\} \mid \prod\{\, a?(x : A_a)\,\omega!\langle\rangle \mid a! \in O \,\}.$$

Here the type A_a is determined by the fact that Γ *allows*$_\sigma$ O.

$a!v \cdot s'$: Here the test is given by

$$(\mathsf{n}\ \mathsf{w}\, n)\ \ n!\langle\rangle \mid n?()\,\omega!\langle\rangle \mid a?(x:\mathrm{A}) \quad \mathsf{if}\ x = v \ \mathsf{th}\ \mathsf{n}\ n?()\,T(\Gamma \sqcap v:\mathrm{A}, s', O, I, \sigma)\ \mathsf{ls}\ \ \mathbf{0}$$

where A is determined by $\mathsf{r}_\delta\langle \mathrm{A}\rangle \in \Gamma(a)$ for some $\delta \preceq \sigma$.

$(c)a!c \cdot s'$: Here it is defined by

$(\mathsf{n}\ \mathsf{w}\, n)\ n!$

4 Non-Interference Results

In this section we reconsider the approach taken to non-interference in Section 4 of [10]. The essential idea is that if a process is well-typed at a given level σ then its behaviour at that level is independent of processes "running at higher security levels"; or more generally "running at security levels independent to σ". A particular formulation of such a result was given in Theorem 5.3 of [10]:

THEOREM 4.1. *If* $\Gamma \vdash_\sigma P$, *and* $\Gamma \vdash_\top H, K$, *where* H, K *are* σ-free *processes, then* $\Gamma \rhd_\sigma P \simeq_{may}$ *implies* $\Gamma \rhd_\sigma P \mid H \simeq_{may} \mid K$.

Here, because of our more refined notions of well-typing, we can give offer a significant improvement on this Theorem, and moreover the formulation is actually easier.

Let us say that the security level δ is independent of σ if $\delta \not\leq \sigma$. We can ensure that a process H is "running at a security level independent to σ" by demanding that $\Delta \vDash^{\delta} H$, for some δ independent of σ. In fact we will only require the weaker typing relation $\Delta \vDash^{w\,\delta} H$. This ensures that all the output actions of H are at a level independent of σ, as can be deduced from the following property:

LEMMA 4.2. *Suppose* $\Delta \vDash^{w\,\delta} H$. *Then* $\Gamma; \Delta \rhd H \xrightarrow{\mu}$

The consistency requirement on types implies $\delta' \preceq \sigma'$, which contradicts $\delta \not\preceq \sigma$.

Output from P to H: Here the derivation takes the form

$$\Gamma; \Delta \triangleright P \mid H \xrightarrow{\tau}_{\sigma} \Gamma; \Delta \triangleright (\tilde{c}:\tilde{C})(P' \mid H') \stackrel{s}{\Longrightarrow}_{\sigma}$$

where $P \xrightarrow{(\tilde{c})a!v} P'$ and $H \xrightarrow{a?v} H'$. So there exists a sequence s_C, associated with s, such that

$$\Gamma; \Delta\,; \tilde{c}:\tilde{C} \triangleright P' \mid H' \stackrel{s_C}{\Longrightarrow}_{\sigma} \qquad (*)$$

with the property that for for any R such that $\Gamma; \Delta\,; \tilde{c}:\tilde{C} \triangleright R \stackrel{s_C}{\Longrightarrow}_{\sigma}$ it follows that $\Gamma; \Delta \triangleright (\tilde{c}:\tilde{C})R \stackrel{s}{\Longrightarrow}_{\sigma}$.

Applying induction to $(*)$ we obtain

$$\Gamma; \Delta\,; \tilde{c}:\tilde{C} \triangleright P' \stackrel{s_C}{\Longrightarrow}_{\sigma}$$

Note that this is possible since Subject Reduction gives

$$\Delta, \tilde{c}:\tilde{C} \vdash_{r\sigma} P', \quad \Delta \sqcap v : \mathrm{A} \vdash^{w\delta} H'$$

where A is a type such that $\Delta, \tilde{c}:\tilde{C} <: \Delta \sqcap v : \mathrm{A}$. (In fact A is the type at which v is sent by P.)

It follows that $\Gamma; \Delta, \tilde{c}:\tilde{C} \triangleright P' \mid a!\langle v\rangle \stackrel{s_C}{\Longrightarrow}_{\sigma}$ and therefore

$$\Gamma; \Delta \triangleright (\mathsf{n}\ \mathsf{w}\,\tilde{c}:\tilde{C})\ (P' \mid a!\langle v\rangle) \stackrel{s}{\Longrightarrow}_{\sigma} .$$

But by Lemma 2.5 we know

$$P \equiv_{st} (\mathsf{n}\ \mathsf{w}\,\tilde{c}:\tilde{C})\ (P' \mid a!\langle v\rangle)$$

and the result follows.

□

We end the paper with a non-interference result with respect to *must* testing. Note that Theorem 4.3 is no longer true when $\sqsubseteq_{may}$ is replaced by

The presence or absence of H determines whether or not there is read contention on the channel n, which in turn influences the deadlock capabilities of P with respect to the channel a.

Here the problem is the type of the channel n; it may be read at both level bot and top. A not unreasonable restriction would be to require that the read capability of channels be confined to a particular security level, using single-level types. This would not rule out inter-level communication, but simply control it more tightly.

THEOREM 4.5 (NON-INTERFERENCE 2). *Let* Γ *and* Δ *be compatible single-level environments and suppose* $\Delta \vdash_{r\sigma} P$, . *Then*

$$\Gamma \rhd_\sigma P \sqsubseteq_{must} \quad \textit{implies } \Gamma \rhd_\sigma P \mid H \sqsubseteq_{must} \quad \mid K$$

for all finite processes H, K *such that* $\Delta \vdash^{w\,\delta} H, K$ *for some* δ *independent of* σ.

Note that we must restrict our attention to finite H and K since *must* testing is sensitive to divergence; if H is a divergent term then we could not expect $\Gamma \rhd^\sigma P \mid \mathbf{0} \simeq_{must} P \mid H$ to hold when P is a convergent term. This problem is avoided by restricting attention to finite terms,which can never diverge.

The remainder of the section is devoted to the proof of this final result of the paper. Throughout we will assume Γ and Δ are compatible single-level environments, $\Delta \vdash_{r\sigma} P$, $\Delta \vdash^{w\,\delta} H$ for some δ independent of σ, and moreover that H is a finite process.

LEMMA 4.6. *For every* s, $\Gamma; \Delta \rhd P \Downarrow \sqcap \nexists \in \{$

- The empty derivation.
  Here $A = \mathcal{R}^\sigma(\Gamma;\Delta \triangleright P)$. This means that $P \not\xrightarrow{\tau}$ but we may have $P \mid H \xrightarrow{\tau}$ either because $H \xrightarrow{\tau}$ or there may be a write up from P to H. But because H is syntactically finite and $P \Downarrow$ we know there is some $P'|H'$ such that $P|H \xrightarrow{\tau}{}^* P'|H' \not\xrightarrow{\tau}$. Let O be $\mathcal{O}^\sigma(\Gamma;\Delta \triangleright P'|H')$.
  Since $c(I) \cap A = \emptyset$ we know that $O \in \mathcal{O}^\sigma_I(\Gamma;\Delta \triangleright P|H,s)$ and because P' is obtained from P by write-ups it follows that $O \subseteq A$.
- The derivation has the form $\Gamma;\Delta \triangleright P \xrightarrow{(c)a!v}_\sigma \Gamma';\Delta' \triangleright P' \stackrel{s}{\Longrightarrow}_\sigma \mathcal{D}$.
  By Subject Reduction we know $\Delta' \vdash_{r\sigma} P'$ and therefore we may apply induction to obtain $O \in \mathcal{O}^\sigma_I(\Gamma;\Delta \triangleright P' \mid H,s)$ with the required properties. The result now follows since $\mathcal{O}^\sigma_I(\Gamma;\Delta \triangleright P' \mid H,s) \subseteq \mathcal{O}^\sigma_I(\Gamma;\Delta \triangleright P \mid H,(c)a!v \cdot s)$
- The remaining cases are similar.

□

We also have the converse.

Proposition 4.8. *Suppose $A \in \mathcal{A}^\sigma(\Gamma;\Delta \triangleright P|H,s)$ and, as in the previous Proposition, I is a set of inputs such that $c(I) \cap A = \emptyset$ and $(\Gamma \text{ after}_\sigma s)$ allows$_\sigma$ I. Then there exists some $O \in \mathcal{O}^\sigma_I(\Gamma;\Delta \triangleright P,s)$ such that $O - \overline{c}(I) \subseteq A$.*

Proof. Again by induction on the derivation

$$\Gamma;\Delta \triangleright P \mid H \stackrel{s}{\Longrightarrow}_\sigma \mathcal{D}, \quad \text{where } A = \mathcal{R}^\sigma(\mathcal{D})$$

As an example we examine the case

$$\Gamma;\Delta \triangleright P \mid H \xrightarrow{\tau} \mathcal{D}' \stackrel{s}{\Longrightarrow}_\sigma \mathcal{D},$$

where the initial τ consists of a communication between P and H. This must be a write-up from P to H; so $\mathcal{D}'$ has the form $\Gamma;\Delta \triangleright (\tilde{c}:\tilde{C})P' \mid H'$, where $P \xrightarrow{(\tilde{c}:\tilde{C})a!v} P'$ and $H \xrightarrow{a?v} H'$. We know P has the form $(\tilde{c}:\tilde{C})(a!\langle v\rangle \mid P')$, but more importantly that $\mathsf{r}_\delta\langle \mathrm{A}\rangle \in \Delta(a)$ for some δ independent from σ (†). What this means is there can can be no communication between $a!\langle v\rangle$ and any such that $\Delta \vdash_{r\sigma}$.

Now the derivation $\Gamma;\Delta \triangleright (\tilde{c}:\tilde{C})(P' \mid H') \stackrel{s}{\Longrightarrow}_\sigma \mathcal{D}$ can be transformed into $\Gamma;\Delta,\tilde{c}:\tilde{C} \triangleright P' \mid H' \stackrel{s_C}{\Longrightarrow}_\sigma \mathcal{E}$, where $\mathcal{R}^\sigma(\mathcal{E}) = \mathcal{R}^\sigma(\mathcal{D})$. Moreover we can apply induction to this derivation, to obtain $O \in \mathcal{O}^\sigma_I(\Gamma;\Delta,\tilde{c}:\tilde{C} \triangleright P', s_C)$ such that $O - \overline{c}(I) \subseteq A$.

We can use (†) to prove O is also in $\mathcal{O}^\sigma_I(\Gamma;\Delta,\tilde{c}:\tilde{C} \triangleright a!\langle a\rangle \mid P', s_C)$. The result now follows since

$$\mathcal{O}^\sigma_I(\Gamma;\Delta,\tilde{c}:\tilde{C} \triangleright a!\langle a\rangle \mid P', s_C) \subseteq \mathcal{O}^\sigma_I(\Gamma;\Delta,\tilde{c}:\tilde{C} \triangleright (\tilde{c}:\tilde{C})(a!\langle a\rangle \mid P'), s).$$

□

COROLLARY 4.9. *(Theorem 4.5) suppose* $\Delta \vdash_{r\sigma} P, \quad .$ *Then*

$$\Gamma \rhd_\sigma P \sqsubseteq_{must} \quad \textit{implies } \Gamma \rhd_\sigma P \,|\, H \sqsubseteq_{must} \quad |\, K$$

for all finite processes H, K *such that* $\Delta \vdash^{w\,\delta} H, K$ *for some* δ *independent of* σ.

Proof. It is sufficient to prove

$$(\Gamma, \Gamma'); \Delta \rhd P \ll^\sigma (\Gamma, \Gamma'); \Delta \rhd P \,|\, H \quad \text{and} \quad (\Gamma, \Gamma'); \Delta \rhd P \,|\, H \ll^\sigma (\Gamma, \Gamma'); \Delta \rhd P.$$

These follow from the two previous Propositions and Lemma 4.6. □

5 Conclusions and Related Work

This paper is a direct continuation of the research reported in [10]. There we focused on the general topic of security types, showing that *resource access control* could be enforced using a typing system and *information flow control* could be obtained by a restriction to the set of types employed. The import of Subject Reduction was emphasised by developing a Type Safety Theorem, which in turn required a version of the language in which processes were tagged with their security levels. Here we concentrated on types for *information flow*, calling the resulting language the *security* π-calculus. The first main result consists of alternative characterisations of *may* and *must* testing for this language. These uses a novel labelled transition system, which records the ve8

properties have been shown to be expressible in terms of non-interference and it would be interesting to see whether these can be enforced by typing constraints using a type system such as ours. This would involve extending our

27th Internationa Conference on Automata, Languages and Programming, LNCS 1853, pages 354–371. Springer-Verlag, July 2000.

[8] J. A. Goguen and J. Meseguer. Security policies and security models. In *IEEE S mposium on Securit and privac* , 1992.

[9] M. Hennessy. *An Algebraic Theor of Processes.* MIT Press, 1988.

[10] Matthew Hennessy and James iely. Information flow vs resource access in the asynchronous pi-calculus (extended abstract). In U. Montanari, J. olim, and E. Welzl, editors, *Automata, Languages and Programming, 27th International Colloquium*, volume 1853 of *Lecture Notes in Computer Science*, pages 415–427, Geneva, Switzerland, 9–15 July 2000. Springer-Verlag. Full version available as CS Technical eport 200:3, University of Sussex.

[11] Kohei Honda and Mario Tokoro. On asynchronous communication semantics. In P. Wegner M. Tokoro, O. Nierstrasz, editor, *Proceedings of the ECOOP '91 Workshop on Object-Based Concurrent Computing*, volume 612 of *LNCS 612.* Springer-Verlag, 1992.

[12] Kohei Honda, Vasco Vasconcelos, and Nobuko Yoshida Honda. Secure information flow as typed process behaviour. In *Proceedings of European S mposium on Programming (ESOP) 2000.* Springer-Verlag, 2000.

[13] . Milner, J. Parrow, and D. Walker. Mobile logics for mobile processes. *Theoretical Computer Science*, 114:149–171, 1993.

[14] . De Nicola and M. Hennessy. Testing equivalences for processes. *Theoretical Computer Science*, 24:83–113, 1984.

[15] B. Pierce and D. Sangiorgi. Behavioral equivalence in the polymorphic pi-calculus. In *Proc. 24th POPL.* ACM Press, 1997. Full paper to appear in Journal of the ACM.

[16] Benjamin Pierce and Davide Sangiorgi. Typing and subtyping for mobile processes. *Mathematical Structures in Computer Science*, 6(5):409–454, 1996. Extended abstract in LICS '93.

[17] James iely and Matthew Hennessy. esource access control in systems of mobile agents (extended abstract). In *Proceedings of 3rd International Workshop on High-Level Concurrent Languages*, Nice, France, September 1998. Full version available as Computer Science Technical eport 2/98, University of Sussex, 1997. Available from `http://www.cogs.susx.ac.uk/`.

[18] A.W. oscoe, J.C.P. Woodcock, and L. Wulf. Non-interference through determinism. In *e,*