State-oriented noninterference for CCS

Ilaria Castellani

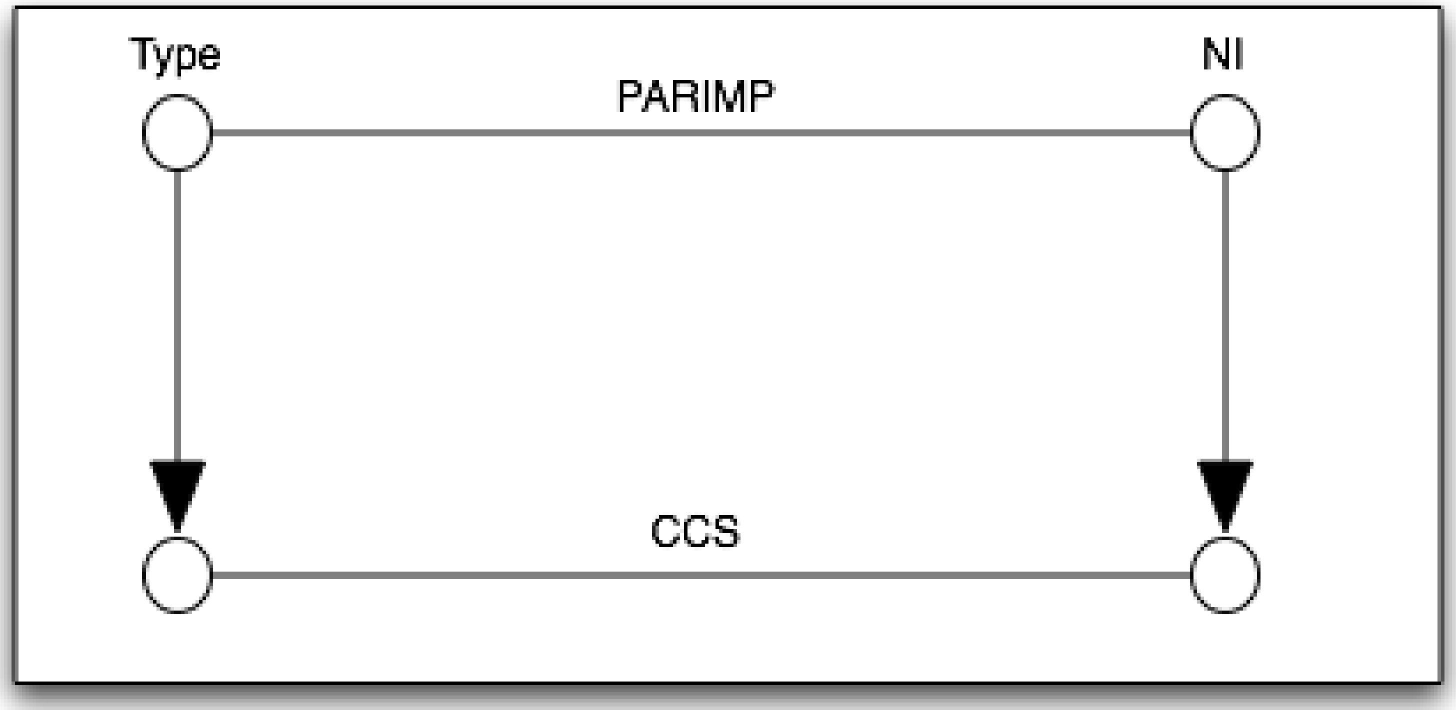
Inria Sophia Antipolis

2ème réunion PARSEC **Paris, 20 juin 2007**

Motivation

- Relate language-based security and process calculi security.
- First objective: relate the noninterference property (NI) for a parallel imperative language with security properties for CCS.
- Starting point:
  - [Focardi, Rossi & Sabelfeld '05]: translation of a sequential imperative language into CCS, preserving *time-sensitive* NI.
  - [Honda, Yoshida, Vasconcelos '01] and following papers: translation of more powerful languages into a variant of the π-calculus, preserving both NI and types.

Objective



Translate a parallel imperative language PARIMP into the process calculus CCS, preserving both noninterference (NI) and types.

Language-based security

- Information: contained in "objects", used by "subjects".
- Objects have security levels, eg: high = secret, low = public.
- Secure information flow: no flow from high to low objects.

$X_L := Y_H$	not secure
$Z_H := Y_H \ ; \ X_L := 0$	secure

- Imperative languages:
  - Subjects = programs. Objects = variables. Tools:
  - (self-)bisimulation to formalise the security property;
  - type systems to statically ensure it.

Process calculi security

- Subjects = processes. Objects = channels $a, b, c \ldots$

$$a_h(x).\,\overline{b}_\ell\langle x\rangle \qquad\qquad\qquad \text{not secure}$$

- Data flow and control flow are closely intertwined:

$$a_h(x).\,\overline{b}_\ell \qquad\qquad a_h(x).\,\overline{b}_\ell\langle v\rangle \qquad\qquad \text{secure?}$$

Warning ! Can be used to implement indirect insecure flows:

$$(\, a_h(x).\,\texttt{if}\ x\ \texttt{then}\ \overline{b}_\ell\ \texttt{else}\ \overline{c}_\ell \mid (\, b_\ell.\,\overline{d}_\ell\langle 0\rangle \;+\; c_\ell.\,\overline{d}_\ell\langle 1\rangle \,)\,) \setminus \{b_\ell, c_\ell\}$$

The imperative language PARIMP

Variables X, Y, Z, values V, V' and expressions E, E':

$$E ::= F(X_1, \ldots, X_n)$$

Syntax of programs (or commands) C, D:

$$\begin{aligned} C,\ D \quad ::= \quad & \texttt{nil} \ \mid\ X := E \ \mid\ C\,;\,D \ \mid\ (\texttt{if } E \texttt{ then } C \texttt{ else } D) \ \mid \\ & (\texttt{while } E \texttt{ do } C) \ \mid\ (C \parallel D) \end{aligned}$$

Semantics: transitions on configurations $\langle C, s\rangle \to \langle C', s'\rangle$ where s, s' are *states* (finite mappings from variables to values).

Operational semantics of PARIMP (1/3)

(Assign-Op) $$\dfrac{}{\langle X := E, s\rangle \rightarrow \langle \mathtt{nil}, s[s(E)/X]\rangle}$$

(Seq-Op1) $$\dfrac{\langle C, s\rangle \rightarrow \langle C', s'\rangle}{\langle C; D, s\rangle \rightarrow \langle C'; D, s'\rangle}$$

(Seq-Op2) $$\dfrac{}{\langle \mathtt{nil}; D, s\rangle \rightarrow \langle D, s\rangle}$$

Operational semantics of PARIMP (2/3)

(Cond-Op1) $$\frac{s(E) = tt}{\langle \texttt{if } E \texttt{ then } C \texttt{ else } D, s\rangle \longrightarrow \langle C, s\rangle}$$

(Cond-Op2) $$\frac{s(E) \neq tt}{\langle \texttt{if } E \texttt{ then } C \texttt{ else } D, s\rangle \longrightarrow \langle D, s\rangle}$$

(While-Op1) $$\frac{s(E) = tt}{\langle \texttt{while } E \texttt{ do } C, s\rangle \longrightarrow \langle C; \texttt{while } E \texttt{ do } C, s\rangle}$$

(While-Op2) $$\frac{s(E) \neq tt}{\langle \texttt{while } E \texttt{ do } C, s\rangle \longrightarrow \langle \texttt{nil}, s\rangle}$$

Operational semantics of PARIMP (3/3)

(ParL-Op1) $$\frac{\langle C, s\rangle \to \langle C', s'\rangle}{\langle C \parallel D, s\rangle \to \langle C' \parallel D, s'\rangle}$$

(ParL-Op2) $$\frac{}{\langle \texttt{nil} \parallel D, s\rangle \to \langle D, s\rangle}$$

(ParR-Op1) $$\frac{\langle D, s\rangle \to \langle D', s'\rangle}{\langle C \parallel D, s\rangle \to \langle C \parallel D', s'\rangle}$$

(ParR-Op2) $$\frac{}{\langle C \parallel \texttt{nil}, s\rangle \to \langle C, s\rangle}$$

Security property for PARIMP

Variables: partitioned into L (low variables) and H (high variables).

L-equality on states:

$s =_L t$ if $\mathtt{dom}(s) = \mathtt{dom}(t)$ and $(X \in \mathtt{dom}(s) \cap L \;\Rightarrow\; s(X) = t(X))$

L-bisimulation on programs:

Symmetric relation $\mathcal{S} \subseteq (\mathcal{C} \times \mathcal{C})$ such that $C\,\mathcal{S}\,D$ implies, for any s and t such that $s =_L t$:

$$\text{if } \langle C, s\rangle \to \langle C', s'\rangle, \text{ then there exist } D', t' \text{ such that}$$

$$\langle D, t\rangle \mapsto \langle D', t'\rangle \quad \text{where} \quad s' =_L t' \quad \text{and} \quad C'\,\mathcal{S}\,D'$$

where $\mapsto$ is the reflexive closure of $\to$ (at most one step).

Security property for PARIMP (ctd)

L-bisimilarity: $C \simeq_L D$ if $C \mathcal{S} D$ for some L-bisimulation $\mathcal{S}$.

L-security: a program C is *L-secure* if $C \simeq_L C$.

Examples of insecure programs:

1. $(\texttt{while}\ x_H\ \texttt{do nil})\,;\, y_L := 0$

2. $\texttt{if}\ x_H = 0 \quad \texttt{then}\ \ \texttt{loop}\ (y_L := 0\,;\, y_L := 1)$
   $\qquad\qquad\qquad\ \ \texttt{else}\ \ \texttt{loop}\ (y_L := 1\,;\, y_L := 0)$

   where $\texttt{loop}\ C \stackrel{\text{def}}{=} (\texttt{while}\ tt\ \texttt{do}\ C)$.

The process calculus CCS (core)

Process prefixes: $\pi ::= a(x) \;\mid\; \overline{a}\langle e\rangle \;\mid\; a \;\mid\; \overline{a}$

Parametric processes: $T ::= A \;\mid\; (\texttt{rec}\, A(\tilde{x}) \,.\, P)$

Syntax of CCS processes:

$$P, Q \quad ::= \quad \textstyle\sum_{i \in I} \pi_i.P_i \;\mid\; (P \mid Q) \;\mid\; (\nu a)\, P \;\mid\; T(\tilde{e})$$

Abbreviations:

$$\mathbf{0} \stackrel{\text{def}}{=} \textstyle\sum_{i \in \emptyset} \pi_i.P_i \qquad \pi_1.P_1 + \pi_2.P_2 \stackrel{\text{def}}{=} \textstyle\sum_{i \in \{1,2\}} \pi_i.P_i$$

Semantics of CCS (1/3)

Actions α, β, γ:

$$Act \stackrel{\text{df}}{=} \{av : a \in \mathcal{N},\, v \in Val\} \cup \{\bar{a}v : a \in \mathcal{N},\, v \in Val\} \cup \{\tau\}$$

Operational rules for nondeterministic choice:

$$(\text{SUM-OP}_1) \quad \textstyle\sum_{i \in I} \pi_i.P_i \xrightarrow{av} P_i\{v/x\}, \quad \text{if } \pi_i = a(x) \text{ and } v \in Val$$

$$(\text{SUM-OP}_2) \quad \textstyle\sum_{i \in I} \pi_i.P_i \xrightarrow{\overline{a}v} P_i\,, \quad \text{if } \pi_i = \overline{a}\langle e \rangle \text{ and } val(e) = v$$

Semantics of CCS (2/3)

Operational rules for parallelism, restriction and recursion:

$$(\text{PAR-OP}_1) \quad \frac{P \xrightarrow{\alpha} P'}{P \mid Q \xrightarrow{\alpha} P' \mid Q} \qquad (\text{PAR-OP}_2) \quad \frac{P \xrightarrow{\alpha} P'}{Q \mid P \xrightarrow{\alpha} Q \mid P'}$$

$$(\text{PAR-OP}_3) \quad \frac{P \xrightarrow{av} P' \quad Q \xrightarrow{\overline{a}v} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'} \qquad (\text{RES-OP}) \quad \frac{P \xrightarrow{\alpha} P' \quad b \neq subj(\alpha)}{(\nu b)P \xrightarrow{\alpha} (\nu b)P'}$$

$$(\text{REC-OP}) \quad \frac{P\{\tilde{v}/\tilde{x}\}\{\,(\texttt{rec}\ A(\tilde{x})\,.\,P)\,/\,A\,\} \xrightarrow{\alpha} P' \qquad \tilde{v} = val(\tilde{e})}{(\texttt{rec}\ A(\tilde{x})\,.\,P)(\tilde{e}) \xrightarrow{\alpha} P'}$$

Security properties for CCS

Weak transitions:

- $P \stackrel{\alpha}{\Longrightarrow} P' \quad \stackrel{\text{df}}{=} \quad P \xrightarrow{\tau}{}^{*} \xrightarrow{\alpha} \xrightarrow{\tau}{}^{*}$
- $P \stackrel{\hat{\alpha}}{\Longrightarrow} P' \quad \stackrel{\text{df}}{=} \quad \begin{cases} P \stackrel{\alpha}{\Longrightarrow} P' & \text{if } \alpha \neq \tau \\ P \xrightarrow{\tau}{}^{*} P' & \text{if } \alpha = \tau \end{cases}$

Weak bisimulation:

Symmetric relation $\mathcal{S} \subseteq (\mathcal{P}r \times \mathcal{P}r)$ such that $P \ \mathcal{S} \ Q$ implies:

if $P \xrightarrow{\alpha} P'$ then there exists Q' such that $Q \stackrel{\hat{\alpha}}{\Longrightarrow} Q'$ and $P' \ \mathcal{S} \ Q'$.

Weak bisimilarity: $P \approx Q$ if $P \ \mathcal{S} \ Q$ for some weak bisimulation $\mathcal{S}$.

Simple security (BNDC) [Focardi-Gorrieri '95]

Channels are partitioned into high channels $\mathcal{H}$ and low channels $\mathcal{L}$.

$\mathcal{P}r^{\mathcal{H}}_{\texttt{syn}}$: set of syntactically high processes, with no channels in $\mathcal{L}$.

Bisimulation-based Non Deducibility on Compositions (BNDC)

P is secure with respect to $\mathcal{H}$, $P \in \mathsf{BNDC}_{\mathcal{H}}$, if for every $\Pi \in \mathcal{P}r^{\mathcal{H}}_{\texttt{syn}}$:

$$(\nu\mathcal{H})(P \mid \Pi) \approx (\nu\mathcal{H})P$$

Examples.

$a_h \,.\, \overline{b}_\ell$	$a_h \;+\; \overline{b}_\ell$	not secure
$a_h \mid \overline{b}_\ell$	$a_h \,.\, \overline{b}_\ell \;+\; \overline{b}_\ell$	secure

Choosing $\Pi = \overline{a_h}$ for the first two, we get $(\nu\mathcal{H})(P \mid \Pi) \not\approx (\nu\mathcal{H})P$.

A more robust security property

Transitions $\stackrel{\widetilde{\alpha}}{\Longrightarrow}_{\mathcal{H}}$, allowing simulation of high actions by inaction:

$$P \stackrel{\widetilde{\alpha}}{\Longrightarrow}_{\mathcal{H}} P' \quad \stackrel{\text{df}}{=} \quad \begin{cases} P \stackrel{\hat{\alpha}}{\Longrightarrow} P' \text{ or } P \stackrel{\tau}{\longrightarrow}^{*} P' & \text{if} \quad subj(\alpha) \in \mathcal{H} \\ P \stackrel{\hat{\alpha}}{\Longrightarrow} P' & \text{otherwise} \end{cases}$$

Weak bisimulation up-to-high:

Symmetric relation $\mathcal{S} \subseteq (\mathcal{P}r \times \mathcal{P}r)$ such that $P \; \mathcal{S} \; Q$ implies:

if $P \stackrel{\alpha}{\longrightarrow} P'$ then there exists Q' such that $Q \stackrel{\widetilde{\alpha}}{\Longrightarrow}_{\mathcal{H}} Q'$ and $P' \; \mathcal{S} \; Q'$.

Weak bisimilarity up to high: $P \approx_{\mathcal{H}} Q$ if $P \; \mathcal{S} \; Q$ for some weak bisimulation up to high $\mathcal{S}$.

Persistent security (PBNDC)[Focardi-Rossi '02]

Persistent BNDC (PBNDC)

P is persistently secure wrt $\mathcal{H}$, $P \in \mathsf{PBNDC}_{\mathcal{H}}$, if $P \approx_{\mathcal{H}} (\nu\mathcal{H})P$.

Theorem [Focardi-Rossi '02].

$P \in \mathsf{PBNDC}_{\mathcal{H}}$ iff $P' \in \mathsf{BNDC}$ for any reachable state P' of P.

Example.

$P = P_1 + P_2 = a_\ell.\,b_h.\,\overline{c_\ell} \;+\; a_\ell.\,(\nu d_\ell)(d_\ell \mid \overline{d_\ell} \mid d_\ell.\,\overline{c_\ell})$ is secure but not persistently secure.

Secure: show that $(\nu\mathcal{H})(P \mid \overline{b_h}) \approx (\nu\mathcal{H})P$.

Not persistently secure: the reachable state $b_h.\,\overline{c_\ell}$ is not secure.

A security type system for PBNDC

Inspired from Pottier's type system for the π-calculus (Pottier '02).

Security levels σ, δ, θ form a lattice $(\mathcal{T}, \leq)$, where $\leq$ stands for "less secret than". Here we assume $\mathcal{T} = \{\ell, h\}$, with $\ell \leq h$.

Type environment Γ: mapping from channels to security levels.

Type judgements: $\Gamma \vdash_{\sigma} P$.

Intuition: σ is a lower bound on the security level of channels in P.

Typing rules

(Sum)

$$\frac{\forall i \in I : \quad \Gamma(\pi_i) = \sigma \qquad \Gamma \vdash_\sigma P_i}{\Gamma \vdash_\sigma \sum_{i \in I} \pi_i . P_i}$$

(Par)

$$\frac{\Gamma \vdash_\sigma P \qquad \Gamma \vdash_\sigma Q}{\Gamma \vdash_\sigma P \mid Q}$$

(Res)

$$\frac{\Gamma, b : \theta \vdash_\sigma P}{\Gamma \vdash_\sigma (\nu b) P}$$

(Sub)

$$\frac{\Gamma \vdash_\sigma P \qquad \sigma' \leq \sigma}{\Gamma \vdash_{\sigma'} P}$$

(Rec$_1$)

$$\frac{\Gamma(A) = \sigma}{\Gamma \vdash_\sigma A(\tilde{e})}$$

(Rec$_2$)

$$\frac{\Gamma, A : \sigma \vdash_\sigma P}{\Gamma \vdash_\sigma (\mathtt{rec}\, A(\tilde{x}) \,.\, P)(\tilde{e})}$$

Soundness of the type system for PBNDC

Lemma [$\approx_{\mathcal{H}}-$ invariance under high actions]

If $\Gamma \vdash_\sigma P$ and $\mathcal{H} = \{\, a \in \mathcal{N} : \Gamma(a) = h \,\}$. If $P \xrightarrow{\alpha} P'$ and $\Gamma(\alpha) = h$ then $P \approx_{\mathcal{H}} P'$.

Main result: typability $\Rightarrow$ persistent security (PBNDC):

Theorem [Soundness]

If $\Gamma \vdash_\sigma P$ then $P \approx_{\mathcal{H}} (\nu\mathcal{H})P$, where $\mathcal{H} = \{\, a \in \mathcal{N} : \Gamma(a) = h \,\}$.

Milner's translation of PARIMP into CCS (1/4)

A variable X is modelled by a *register*:

$$Reg_X(v) \stackrel{\text{def}}{=} put_X(x).Reg_X(x) + \overline{get_X}\langle v\rangle.Reg_X(v)$$

A state s is mapped to a *pool of registers*:

$$[\![s]\!] \quad = \quad Reg_{X_1}(s(X_1)) \mid \cdots \mid Reg_{X_n}(s(X_n)) \qquad \text{if } \mathtt{dom}(s) = \{X_1, \ldots, X_n\}$$

An expression $E = F(X_1, \ldots, X_n)$ is mapped to:

$$[\![F(X_1, \ldots, X_n)]\!] \quad = \quad get_{X_1}(x_1). \cdots .get_{X_n}(x_n).\overline{\mathtt{res}}\langle f(x_1, \ldots, x_n)\rangle.\mathbf{0}$$

Auxiliary operator *Into*, for transmission of values:

$$P\ Into(x)\ Q \stackrel{\text{def}}{=} (P \mid \mathtt{res(x)}.Q)\backslash\mathtt{res}$$

Translation of PARIMP into CCS (2/4)

A special channel `done`, on which processes signal termination.

Auxiliary operators *Done*, *Before* and *Par*:

$$Done \stackrel{\text{def}}{=} \overline{\texttt{done}}.\,\mathbf{0}$$

$$C\ Before\ D \stackrel{\text{def}}{=} (C[d/\texttt{done}] \mid d.\,D)\backslash d$$

$$C_1\ Par\ C_2 \stackrel{\text{def}}{=} ((C_1[d_1/\texttt{done}] \mid C_2[d_2/\texttt{done}]) \mid (d_1.\,d_2.Done + d_2.\,d_1.Done))\backslash\{d_1, d_2\}$$

Translation of PARIMP into CCS (3/4)

Translation of commands:

$$
\begin{array}{rcl}
[\![\texttt{nil}]\!] & = & \mathit{Done} \\
[\![X := E]\!] & = & [\![E]\!]\ \mathit{Into}\,(x)\ (\overline{\mathit{put}_X}\langle x\rangle .\ \mathit{Done}) \\
[\![C\ ;\ D]\!] & = & [\![C]\!]\ \mathit{Before}\,[\![D]\!] \\
[\![(\texttt{if}\ E\ \texttt{then}\ C_1\ \texttt{else}\ C_2)]\!] & = & [\![E]\!]\ \mathit{Into}\,(x)\ \ (\texttt{if}\ x\ \texttt{then}\ [\![C_1]\!]\ \texttt{else}\ [\![C_2]\!]) \\
[\![(\texttt{while}\ E\ \texttt{do}\ C)]\!] & = & W,\ \text{ where } W \stackrel{\text{def}}{=} [\![E]\!]\ \mathit{Into}\,(x) \\
 & & (\texttt{if}\ x\ \texttt{then}\ [\![C]\!]\ \mathit{Before}\ W\ \texttt{else}\ \mathit{Done}) \\
[\![(C_1 \parallel C_2)]\!] & = & [\![C_1]\!]\ \mathit{Par}\,[\![C_2]\!]
\end{array}
$$

Translation of PARIMP into CCS (4/4)

Translation of configurations $\langle C, s\rangle$:

$$[\![\langle C, s\rangle]\!] = (\, [\![C]\!] \mid [\![s]\!] \,) \setminus Acc_s \cup \{\texttt{done}\}$$

where Acc_s is the *access sort* of state s:

$$Acc_s \stackrel{\text{def}}{=} \{\, get_X \,,\, put_X \;\mid\; X \in \texttt{dom}(s) \,\}$$

Problem: atomicity of assignments is not preserved !

$$C \;=\; (X := X + 1 \parallel X := X + 1)$$

Problem with atomicity (1/2)

Program $C = \ (X := X + 1 \parallel X := X + 1)$

The translation of C is:

$$
\begin{array}{rcl}
[\![C]\!] & = & (\,(get_X(x).\,\overline{\texttt{res}}\langle x+1\rangle \mid \texttt{res}(y).\,\overline{put_X}\langle y\rangle.\,\overline{d_1})\,\backslash\texttt{res} \\
 & \mid & (get_X(x).\,\overline{\texttt{res}}\langle x+1\rangle \mid \texttt{res}(y).\,\overline{put_X}\langle y\rangle.\,\overline{d_2})\,\backslash\texttt{res} \\
 & \mid & (d_1.\,d_2.Done + d_2.\,d_1.Done)\,)\,\backslash\,\{d_1, d_2\}
\end{array}
$$

The second get_X action may be executed before the first $\overline{put_X}$

$\Rightarrow$ the same value $v_1 = v_0 + 1$ may be assigned twice to X.

Problem with atomicity (2/2)

Suppose X has low level:

$$C_L = (X_L := X_L + 1 \parallel X_L := X_L + 1)$$

Consider the interleaving of the assignments in C_L:

$$D_L = (X_L := X_L + 1\,;\, X_L := X_L + 1)$$

Security is not preserved:

$\widehat{C} = (\texttt{if } z_H = 0 \texttt{ then } C_L \texttt{ else } D_L)$ is secure, but $[\![\widehat{C}]\!]$ is not secure.

Adapting the translation (1/2)

A global semaphore to ensure atomicity:

$$Sem \stackrel{\text{def}}{=} \mathtt{lock}.\,\mathtt{unlock}.\,Sem$$

Adapted translation of assignments and configurations:

$$[\![X := E]\!] \quad = \quad \overline{\mathtt{lock}}.\,[\![E]\!]\ Into\,(x)\ (\overline{put_X}\langle x\rangle.\,\overline{\mathtt{unlock}}.\,Done)$$

$$[\![\langle C, s\rangle]\!] \quad = \quad (\,[\![C]\!] \mid [\![s]\!] \mid Sem\,) \setminus Acc_s \cup \{\mathtt{done}, \mathtt{lock}, \mathtt{unlock}\}$$

Atomic translation of expression E:

$$[\![F(X_1, \ldots, X_n)]\!]_{at} \quad = \quad \overline{\mathtt{lock}}.\,getseq_{\tilde{X}}(\tilde{x}).\,\overline{\mathtt{res}}\langle f(\tilde{x})\rangle.\,\overline{\mathtt{unlock}}.\,\mathbf{0}$$

Adapting the translation (2/2)

Adapted translation of conditionals and loops:

$$\begin{array}{lcl}
[\![(\texttt{if}\ E\ \texttt{then}\ C_1\ \texttt{else}\ C_2)]\!] & = & [\![E]\!]_{at}\ \mathit{Into}\,(x)\ (\texttt{if}\ x\ \texttt{then}\ [\![C_1]\!]\ \texttt{else}\ [\![C_2]\!]) \\
[\![(\texttt{while}\ E\ \texttt{do}\ C)]\!] & = & W,\ \text{where}\ W \stackrel{\text{def}}{=} [\![E]\!]_{at}\ \mathit{Into}\,(x) \\
 & & (\texttt{if}\ x\ \texttt{then}\ [\![C]\!]\ \mathit{Before}\ W\ \texttt{else}\ \mathit{Done})
\end{array}$$

Security is preserved by the translation

To set an operational correspondence between $\langle C, s \rangle$ and its image:

$$[\![\langle C, s \rangle]\!] = (\, [\![C]\!] \mid [\![s]\!] \mid \mathit{Sem}\,) \setminus \mathit{Acc}_s \cup \{\mathtt{done}, \mathtt{lock}, \mathtt{unlock}\}$$

one needs a means to observe changes performed by $[\![C]\!]$ on $[\![s]\!]$.

Observable register OReg_X:

$$\begin{aligned}
\mathit{OReg}_X(v) \stackrel{\text{def}}{=}\quad & \mathit{put}_X(x).\,\mathit{OReg}_X(x) + \overline{\mathit{get}_X}\langle v\rangle.\,\mathit{OReg}_X(v) + \\
& \overline{\mathtt{lock}}.\,(\,\mathit{in}_X(x)\,.\,\overline{\mathtt{unlock}}.\,\mathit{OReg}_X(x) + \overline{\mathtt{unlock}}.\,\mathit{OReg}_X(x)) + \\
& \overline{\mathtt{lock}}.\,(\,\overline{\mathit{out}_X}\langle v\rangle.\,\overline{\mathtt{unlock}}.\,\mathit{OReg}_X(v) + \overline{\mathtt{unlock}}.\,\mathit{OReg}_X(v))
\end{aligned}$$

Operational correspondence

Labelled transitions $\xrightarrow{in_X\, v}$ and $\xrightarrow{\overline{out}_X}$ (and $\xrightarrow{\tau} \stackrel{\text{def}}{=} \longrightarrow$) for configurations:

$$(\textsc{In-Op}) \quad \frac{X \in \mathtt{dom}(s)}{\langle C, s\rangle \xrightarrow{in_X\, v} \langle C, s[v/X]\rangle} \qquad\qquad (\textsc{Out-Op}) \quad \frac{s(X) = v}{\langle C, s\rangle \xrightarrow{\overline{out}_X\, v} \langle C, s\rangle}$$

Transitions are preserved and reflected by the translation:

1. $\langle C, s\rangle \xrightarrow{\alpha} \langle C', s'\rangle$ implies $\exists\, P\,.\ [\![\langle C, s\rangle]\!] \stackrel{\alpha}{\Longrightarrow} P \ \approx\ [\![\langle C', s'\rangle]\!]$
2. $[\![\langle C, s\rangle]\!] \stackrel{\alpha}{\Longrightarrow} P$ implies either $P \approx [\![\langle C, s\rangle]\!]$ or
   $\exists\, C', s'\,.\ P \approx [\![\langle C', s'\rangle]\!] \ \wedge\ \langle C, s\rangle \stackrel{\hat{\alpha}}{\Longrightarrow} \langle C', s'\rangle$.

Security is preserved: C secure $\Rightarrow [\![\langle C, s\rangle]\!]$ satisfies PBNDC.

Types are not preserved by the translation

Consider the program, typable in PARIMP:

$$C = (X_H := X_H + 1\,;\, Y_L := Y_L + 1)$$

Translation of C:

$$(\nu d) \;\; (\,\overline{\texttt{lock}}.\,(\nu\,\texttt{res}_1)\,(get_{X_H}(x).\,\overline{\texttt{res}_1}\langle x+1\rangle \mid \texttt{res}_1(z_1).\,\overline{put_{X_H}}\langle z_1\rangle.\,\overline{\texttt{unlock}}.\,\overline{d}) \;\; \mid$$

$$d.\,\overline{\texttt{lock}}.\,(\nu\,\texttt{res}_2)\,(get_{Y_L}(y).\,\overline{\texttt{res}_2}\langle y+1\rangle \mid \texttt{res}_2(z_2).\,\overline{put_{Y_L}}\langle z_2\rangle.\,\overline{\texttt{unlock}}.\,\overline{\texttt{done}})\,)$$

Which choice of security levels for channels `lock`, `unlock` and d?

Adapting the type system

Idea: restricted high actions without parameters do not leak information if they are granted to be enabled uniformly in all low-equivalent states.

True for actions `lock`, `unlock`: the semaphore is always released after a finite number of steps.

Instead, action `done` may be prevented by deadlock or divergence:

$\Rightarrow$ by restricting the use of loops in the source program one may obtain an ad hoc solution.

Conclusion

1. Security preserving translation into CCS (variant of Milner's), extending work by Focardi, Rossi and Sabelfeld '05 in two ways:

- parallel imperative language
- time insensitive security property

2. Equivalence preserving translation (as a by-product).

3. Security type system for CCS (PBNDC), inspired by Pottier '02, which needs to be tuned to reflect a type system on PARIMP.

Future work:

- more general security type system for CCS
- move to more complex languages