Types for Access Control in a Calculus of Mobile Resources

Hans Hüttel Morten Kühnrich

Department of Computer Science
Aalborg University
Denmark

SecCo'05, 22 August 2005

Process calculi with sites and mobility

A number of process calculi with explicit notions of sites and of mobility have emerged:

- Mobile Ambients [Gordon, Cardelli 1997]
- The Seal Calculus [Vitek, Castagna 1999]
- The Dπ-calculus [Hennessy, Riely 1998]
- The Agent Calculus [Pierce et al., 1998]
- The π^D-calculus [Hoshina et al. , 2001]
- The Calculus of Mobile Resources [Godskesen et al. 2002]

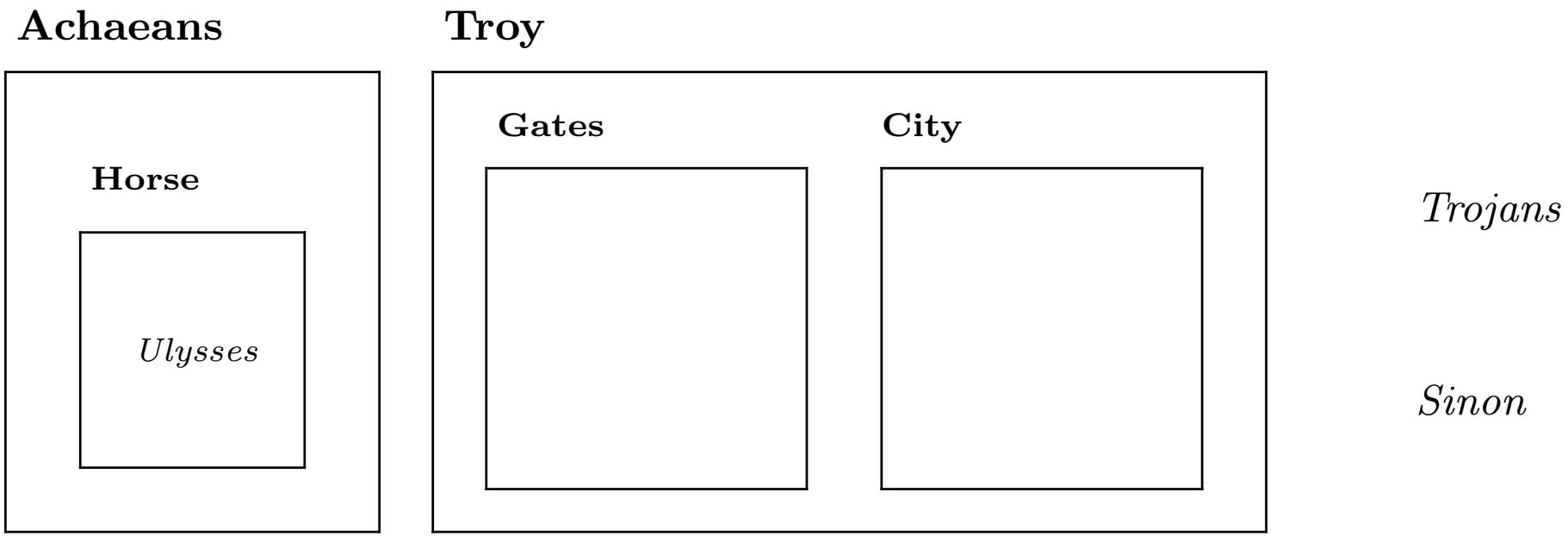
Access control seen through the eyes of process calculi ...

How can we reason about mobility?

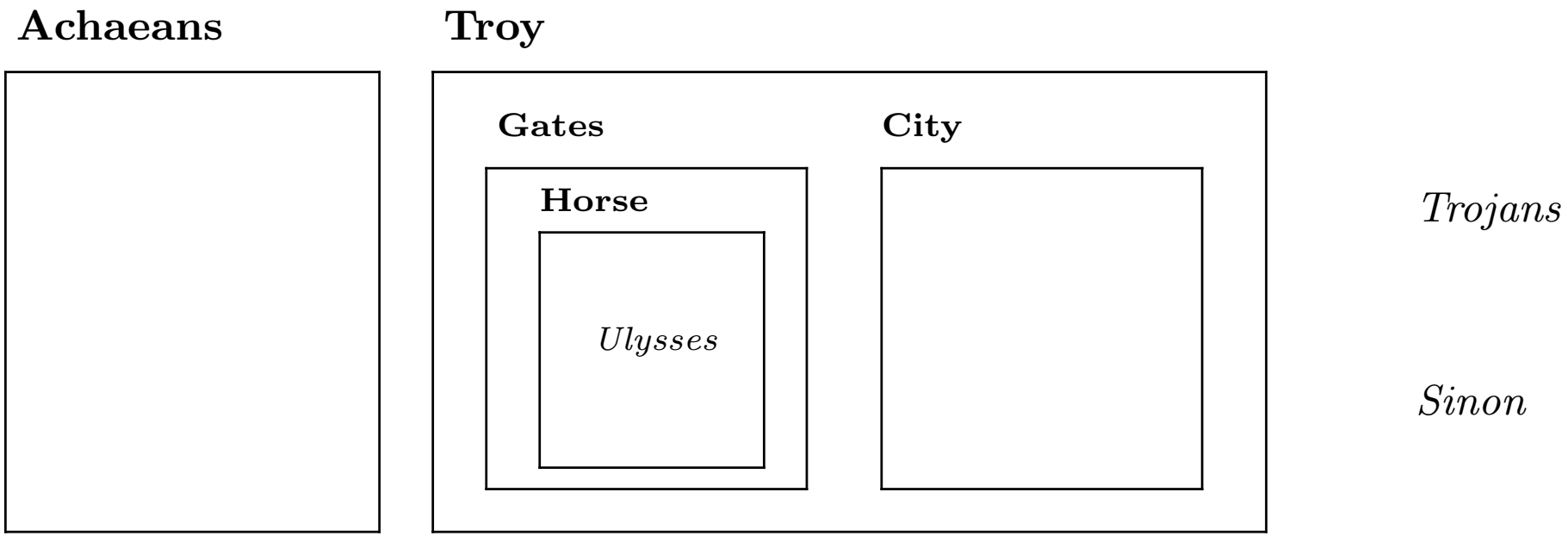
A common approach is to develop a type system that rules out dangerous mobility.

In this talk we describe a type system for controlling border crossing phenomena in the Calculus of Mobile Resources.

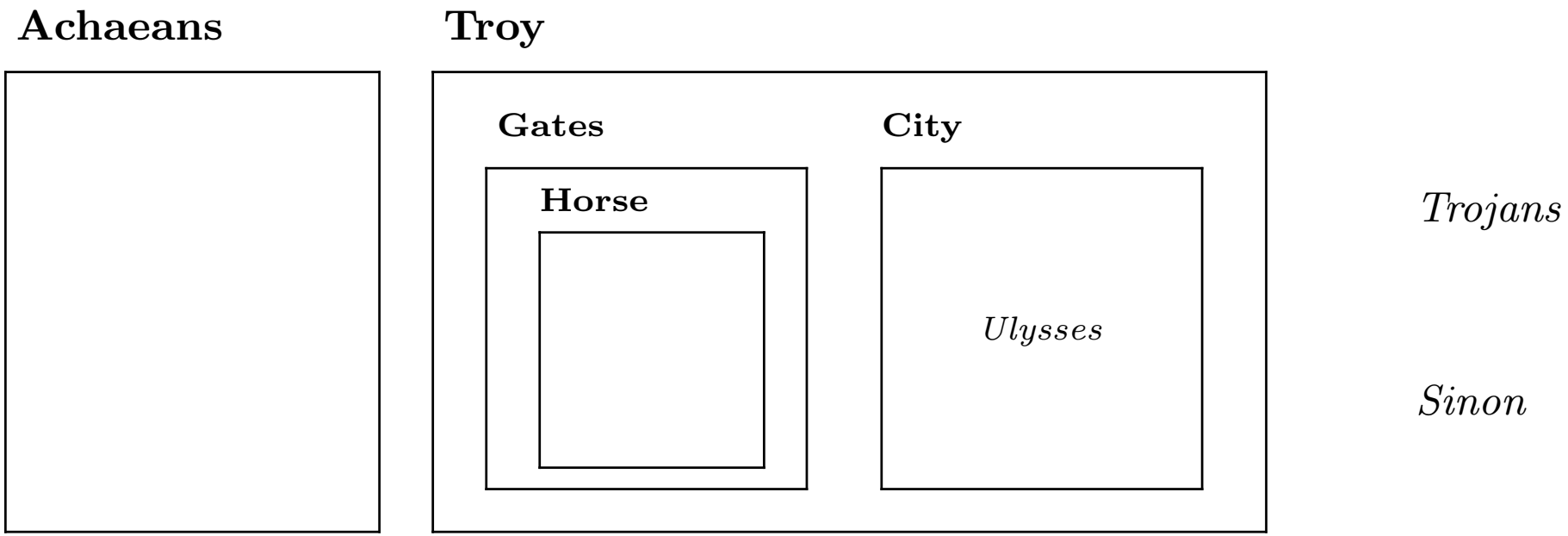
The fall of Troy



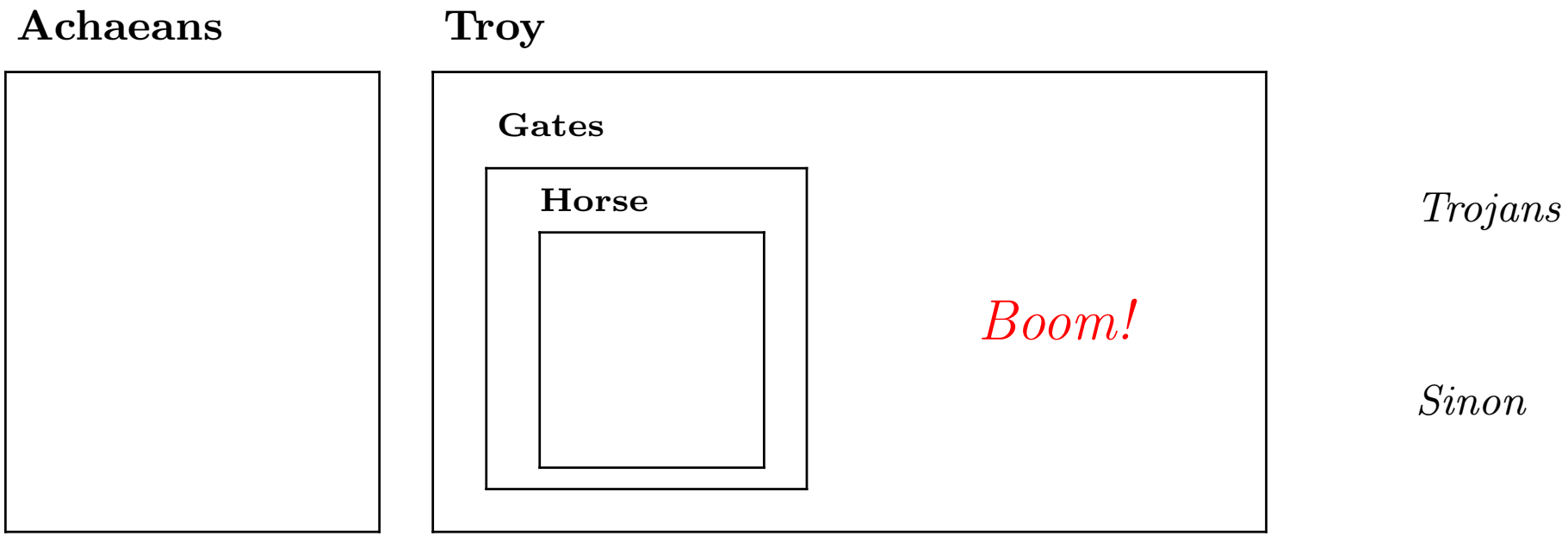
The fall of Troy



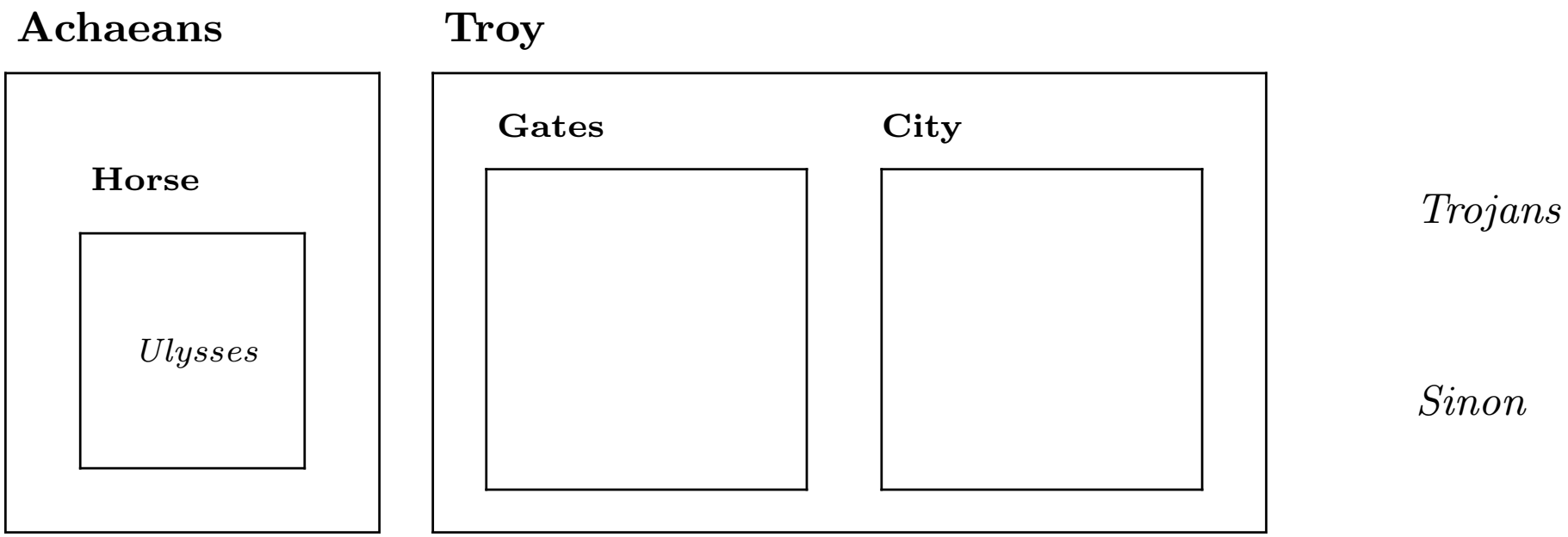
The fall of Troy



The fall of Troy



The fall of Troy



Sinon is a problem here. So is the security policy of Troy.

The MR Calculus

[Godskesen et al. 2002]

- Processes are located at slots
- A slot is either vacant or full (linearity).
- Slots may be nested.
- CCS-style synchronization between arbitrary subslots.
- Moves between arbitrary subslots are allowed.
- Mobility is objective – someone else instructs an object to move (cf. Mobile Ambients, where mobility is subjective)

Syntax of the MR Calculus

Slot and action names are taken from the set $\mathcal{N}$.

p	$::=$	$\mathbf{0}$	nil process
	$\mid$	$\lambda.p$	prefixed process
	$\mid$	$p_1 \parallel p_2$	parallel composition
	$\mid$	$!p$	replication
	$\mid$	$(n : s, b)p$	restriction
	$\mid$	$\tilde{n} \lfloor \cdot \rfloor_m$	vacant slot with deletion name m
	$\mid$	$\tilde{n} \lfloor p \rfloor_m$	full slot with deletion name m
λ	$::=$	$\gamma\alpha$	direction path
	$\mid$	$\delta \rhd \overline{\delta'}$	move resource from slot at δ to slot at δ'
	$\mid$	$\natural\, n$	delete slot with name n

The fall of Troy (MR calculus version)

$$
\begin{aligned}
\text{Ulysses} &\stackrel{\text{def}}{=} !\text{kill.destroy} \\
\text{Horse} &\stackrel{\text{def}}{=} \mathit{horse}\ \lfloor \text{Ulysses} \rfloor \\
\text{Achaeans} &\stackrel{\text{def}}{=} \mathit{camp}\ \lfloor \text{Horse} \rfloor \\
\text{Troy} &\stackrel{\text{def}}{=} \mathit{gates}\ \lfloor \cdot \rfloor \parallel \mathit{city}\ \lfloor \cdot \rfloor \\
\text{Trojans} &\stackrel{\text{def}}{=} \mathit{camp} \rhd \overline{\mathit{gates}} \\
\text{Sinon} &\stackrel{\text{def}}{=} \mathit{gates}\ \mathit{horse} \rhd \overline{\mathit{city}}
\end{aligned}
$$

$$
\text{Invasion} \stackrel{\text{def}}{=} \text{Achaeans} \parallel \text{Trojans} \parallel \text{Troy} \parallel \text{Sinon}
$$

Reduction semantics

In $\mathcal{C}_\gamma(q)$ and $\mathcal{D}_\gamma(q)$ a subprocess q is found along path γ.

$$\gamma\alpha.p \parallel \mathcal{C}_\gamma(\overline{\alpha}.q) \longrightarrow p \parallel \mathcal{C}_\gamma(q)$$

$$\gamma\delta_1 \rhd \overline{\gamma\delta_2}.p \parallel \mathcal{C}_\gamma\Big(\mathcal{D}_{\delta_1}(q) \parallel \mathcal{D}_{\delta_2}(\cdot)\Big) \longrightarrow p \parallel \mathcal{C}_\gamma\Big(\mathcal{D}_{\delta_1}(\cdot) \parallel \mathcal{D}_{\delta_2}(q)\Big)$$

$$\natural\, m.p \parallel \tilde{n}\,\lfloor r \rfloor_m \longrightarrow p, \quad r = \cdot \vee r = q$$

Groups and security policies

Every slot is assigned a group from the set B and a security policy from the set $\mathcal{S}$. Types T range over B and $\mathcal{S}$.
A security policy is a quadruple:

$$\textsc{Sec}[\underbrace{\mathrm{TG}}_{\text{Trusted guests}}, \underbrace{\mathrm{TS}}_{\text{Trusted senders}}, \underbrace{\mathrm{TR}}_{\text{Trusted receivers}}, \underbrace{\mathrm{TM}}_{\text{Legal moves}}]$$

If e.g TR is known to be $\{T_1, \ldots, T_i\}$, we write $\mathrm{TR}[T_1, \ldots, T_i]$.

Groups and security policies

- $\star$ is a wildcard type ('any group'). W ranges over $B \cup \{\star\}$.
- A slot with $\mathrm{TR}[T_1, \ldots, T_i]$ allows moves to slots with types in $\{T_1, \ldots, T_i\}$.
- A slot with $\mathrm{TS}[T_1, \ldots, T_i]$ allows moves from slots of types $\{T_1, \ldots, T_i\}$.
- A slot with component $\mathrm{TG}[T_1, \ldots, T_i]$ may contain slots of types $\{T_1, \ldots, T_i\}$.
- A slot with a legal move $W_1 \rhd W_2$ allows a move from a slot of type W_1 to a slot of type W_2.

Camp David

Suppose a slot n has the security policy

$$\textsc{Sec}[\textsc{tr}[\mathrm{CIA}, \mathrm{FBI}]\ \textsc{ts}[\mathrm{Capitol}]\ \textsc{tg}[\mathrm{Politician}]\ \textsc{tm}[\mathrm{Politician} \triangleright \mathrm{Politician}]]$$

1. Moves from n to CIA and FBI are allowed (trusted receivers).
2. Moves from the Capitol to n are allowed (trusted senders).
3. Politicians are welcome to guest n
4. Moves from any politician to another politician are allowed.

A type system for MR

Typing judgements have the form

$$\Delta, \Gamma \vdash p \Rightarrow A \;\&\; e$$

where we assume

- A security policy type environment $\Delta : \mathcal{N} \to \mathcal{S}$.
- A group environment $\Gamma : \mathcal{N} \to B$.

A (the interface) is the set of free slot names. e (the effect) is an abstract description of moves.

Typing slots

$$(\text{T-SLOT})\frac{\Delta, \Gamma \vdash p \Rightarrow A \;\&\; e}{\Delta, \Gamma \vdash \tilde{n}\,\lfloor p \rfloor_m \Rightarrow A \cup \tilde{n} \;\&\; e}\left(\begin{array}{lll} n \in \tilde{n} & & \\ A & \sqsubseteq_{\Delta,\Gamma} & \Delta(n) \\ e & \sqsubseteq & \Delta(n) \end{array}\right)$$

The slots with names in A are allowed to guest a slot named n. Names in $\tilde{n}$ are added to the interface.

The effects in e are legal moves allowed by the security policy..

Typing moves

$$(\text{T-Move})\frac{\Delta, \Gamma \vdash p \Rightarrow A \;\&\; e}{\Delta, \Gamma \vdash \delta \rhd \overline{\delta'}.p \Rightarrow A \;\&\; e \cup \{\Gamma(n) \rhd \Gamma(n')\}} \begin{pmatrix} \delta = \gamma_1 n, \\ \delta' = \gamma_2 n' \\ n \bowtie_{\Delta,\Gamma} n' \\ \text{consistent}_{\Delta,\Gamma}(\delta) \\ \text{consistent}_{\Delta,\Gamma}(\delta') \end{pmatrix}$$

We add an abstract move to the effect.

The receiving slot must allow at least the guests allowed at the sending slot.

Any subslot along paths δ and δ' is allowed as a guest.

Subject reduction

Theorem (Subject reduction)

For any well-typed process $p : A \,\&\, e$ it holds that if $p \longrightarrow p'$ then $p' : A' \,\&\, e'$ is also well-typed and $A' \subseteq A$ and $e' \subseteq e$.

Type safety

We call a process *safe* in (Γ, Δ) if every visible subslot is allowed by the guesthood relation $\sqsubseteq_{\Delta,\Gamma}$ and all effects are allowed by the effect relation $\sqsubseteq$.

Theorem (Safety)

Any well-typed process $\Delta, \Gamma \vdash p \Rightarrow A \,\&\, e$ is safe in (Γ, Δ).

Corollary

If p is well-typed under Δ, Γ and $p \rightarrow^* p'$, then p' is safe.

The goal of type inference

Given:

1. A process p
2. A group environment Γ

compute a security policy Δ^* that makes p typable under Γ.

Say that $\Delta' < \Delta$ if their components are related under pointwise inclusion.

Δ^* must be minimal w.r.t. $<$.

The type inference algorithm

The algorithm extracts a pair of constraint sets $\langle C_1 | C_2 \rangle$ from a given process p under a given group environment Γ.

- C_1 contains constraints on the form $D \subseteq (n, t)$ where D is a set of types, and n is a name and $t \in \{\mathrm{TR}, \mathrm{TS}, \mathrm{TG}, \mathrm{TM}\}$.
- C_2 contains constraints on the form $(m, t) \subseteq (n, t)$, $t \in \{\mathrm{TR}, \mathrm{TS}, \mathrm{TG}, \mathrm{TM}\}$.

The pair of constraint sets is then solved. The solution yields a security policy environment Δ.
We write $(p, \Gamma) \rightsquigarrow \Delta$ if Δ is found from p and Γ.

The type inference algorithm is sound

Definition

Suppose $(p, \Gamma) \rightsquigarrow \Delta$. p is *consistent* if every subprocess $(n : s, b)p'$ has $\Delta(n) = s$.

Theorem

Let Δ^* be the minimal security policy environment for the consistent process p with respect to the group environment Γ. It holds that

1. $\Delta^*, \Gamma \vdash p \Rightarrow A \,\&\, e$ *for some A and e.*
2. $\nexists \Delta' : \Delta' < \Delta^* \wedge \Delta', \Gamma \vdash p \Rightarrow A \,\&\, e$ *for some A and e.*

Conclusion

- A type system for control of mobility in the MR calculus (the first such)
- Subject reduction
- Type safety (Trojan horses cannot appear)
- A sound type inference algorithm

Ideas for further work

- A complexity analysis of the type inference algorithm.
- Completeness of type inference
- Type systems that take causality into account.
- Type systems that control the deletion of slots.

Related Work

- J. Godskesen, T. Hildebrandt and V. Sassone.
  A Calculus of Mobile Resources. *CONCUR'02*, 2002.
- A. Gordon and L. Cardelli. Mobile Ambients
  Theoretical Computer Science, 2002.
- M. Hennessy and J. Riely
  Proceedings of POPL, 1998.
- M. Merro and V. Sassone
  Typing and Subtyping Mobility in Boxed Ambients.
  CONCUR'02, 2002.
- J. Vitek and G. Castagna
  Seal: A framework for secure mobile computations. *LNCS 1686*