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
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 $D\pi$-calculus [Hennessy, Riely 1998]
- The Agent Calculus [Pierce et al., 1998]
- The $\pi^D$-calculus [Hoshina et al., 2001]
- The Calculus of Mobile Resources [Godskesen et al. 2002]
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

Achaeans

- Horse
  - Ulysses

Trojan

- Gates
- City

Trojans

- Sinon
The fall of Troy

Achaeans

Troy

Gates

Horse

Ulysses

City

Trojans

Sinon
The fall of Troy

Achaeans  

Troy

Gates

City

Horse

Ulysses

Trojans

Sinon
The fall of Troy

Achaeans

Troy

Gates

Horse

Boom!

Trojans

Sinon

Hans Hüttel, Morten Kühnrich

Types for Access Control in a Calculus of Mobile Resources
**The fall of Troy**

Sinon is a problem here. So is the security policy of Troy.
[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 $\mathcal{N}$.

\[
p ::= \begin{array}{ll}
0 & \text{nil process} \\
\lambda.p & \text{prefixed process} \\
p_1 \parallel p_2 & \text{parallel composition} \\
!p & \text{replication} \\
(n : s, b)p & \text{restriction} \\
\tilde{n} \lfloor \cdot \rfloor_m & \text{vacant slot with deletion name } m \\
\tilde{n} \lfloor \cdot \rfloor_m & \text{full slot with deletion name } m \\
\end{array}
\]

\[
\lambda ::= \begin{array}{ll}
\gamma \alpha & \text{direction path} \\
\delta \triangleright \delta' & \text{move resource from slot at } \delta \text{ to slot at } \delta' \\
\n & \text{delete slot with name } n
\end{array}
\]
The fall of Troy (MR calculus version)

\[
\begin{align*}
\text{Ulysses} & \quad \overset{\text{def}}{=} \quad \text{!kill.destroy} \\
\text{Horse} & \quad \overset{\text{def}}{=} \quad \text{horse [Ulysses]} \\
\text{Achaeans} & \quad \overset{\text{def}}{=} \quad \text{camp [Horse]} \\
\text{Troy} & \quad \overset{\text{def}}{=} \quad \text{gates [] || city []} \\
\text{Trojans} & \quad \overset{\text{def}}{=} \quad \text{camp \triangleright gates} \\
\text{Sinon} & \quad \overset{\text{def}}{=} \quad \text{gates horse \triangleright city} \\
\text{Invasion} & \quad \overset{\text{def}}{=} \quad \text{Achaeans || Trojans || Troy || Sinon}
\end{align*}
\]
In $C_\gamma(q)$ and $D_\gamma(q)$ a subprocess $q$ is found along path $\gamma$.

\[
\gamma\alpha.p \parallel C_\gamma(\alpha.q) \longrightarrow p \parallel C_\gamma(q)
\]

\[
\gamma\delta_1 > \gamma\delta_2.p \parallel C_\gamma(D_{\delta_1}(q) \parallel D_{\delta_2}(\cdot)) \longrightarrow p \parallel C_\gamma(D_{\delta_1}(\cdot) \parallel D_{\delta_2}(q))
\]

\[
\downarrow m.p \parallel \tilde{n}[r]_m \longrightarrow p, \quad r = \cdot \lor r = q
\]
Groups and security policies

Every slot is assigned a group from the set \( B \) and a security policy from the set \( S \). Types \( T \) range over \( B \) and \( S \). A security policy is a quadruple:

\[
\text{SEC}\left[\text{TG}, \text{TS}, \text{TR}, \text{TM}\right]
\]

- Trusted guests \( \text{TG} \)
- Trusted senders \( \text{TS} \)
- Trusted receivers \( \text{TR} \)
- Legal moves \( \text{TM} \)

If e.g. \( \text{TR} \) is known to be \( \{ T_1, \ldots, T_i \} \), we write \( \text{TR}[T_1, \ldots, T_i] \).
Groups and security policies

- \( \star \) is a wildcard type (‘any group’). \( W \) ranges over \( B \cup \{ \star \} \).
- A slot with \( TR[T_1, \ldots, T_i] \) allows moves to slots with types in \( \{T_1, \ldots, T_i\} \).
- A slot with \( TS[T_1, \ldots, T_i] \) allows moves from slots of types \( \{T_1, \ldots, T_i\} \).
- A slot with component \( TG[T_1, \ldots, T_i] \) may contain slots of types \( \{T_1, \ldots, T_i\} \).
- A slot with a legal move \( W_1 \rightarrow W_2 \) allows a move from a slot of type \( W_1 \) to a slot of type \( W_2 \).
Suppose a slot \( n \) has the security policy

\[
\text{Sec} \left[ \text{tr} \left[ \text{CIA, FBI} \right], \text{ts} \left[ \text{Capitol} \right], \text{tg} \left[ \text{Politician} \right], \text{tm} \left[ \text{Politician} \leftarrow \text{Politician} \right] \right]
\]

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.
Typing judgements have the form

\[ \Delta, \Gamma \vdash p \Rightarrow A \& e \]

where we assume

- A security policy type environment \( \Delta : \mathcal{N} \rightarrow S \).
- A group environment \( \Gamma : \mathcal{N} \rightarrow B \).

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

\[
(T\text{-}SLOT) \quad \frac{\Delta, \Gamma \vdash p \Rightarrow A \& e}{\Delta, \Gamma \vdash \tilde{n} [p]_m \Rightarrow A \cup \tilde{n} \& e} \left( \begin{array}{c} n \in \tilde{n} \\ A \sqsubseteq_{\Delta, \Gamma} \Delta(n) \\ e \sqsubseteq \Delta(n) \end{array} \right)
\]

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

\[(T\text{-Move})\frac{\Delta, \Gamma \vdash p \Rightarrow A \& e}{\Delta, \Gamma \vdash \delta \triangleright \delta'. p \Rightarrow A \& e \cup \{\Gamma(n) \triangleright \Gamma(n')\}}\]

\[
\begin{pmatrix}
\delta = \gamma_1 n, \\
\delta' = \gamma_2 n' \\
n \cong_{\Delta, \Gamma} n' \\
\text{consistent}_{\Delta, \Gamma}(\delta) \\
\text{consistent}_{\Delta, \Gamma}(\delta')
\end{pmatrix}
\]

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 \(\delta\) and \(\delta'\) is allowed as a guest.
Theorem (Subject reduction)

For any well-typed process $p : A \& e$ it holds that if $p \rightarrow p'$ then $p' : A' \& e'$ is also well-typed and $A' \subseteq A$ and $e' \subseteq e$. 
We call a process safe in \((\Gamma, \Delta)\) 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 \((\Gamma, \Delta)\).

**Corollary**

If \(p\) is well-typed under \(\Delta, \Gamma\) and \(p \rightarrow^{*} p'\), then \(p'\) is safe.
The goal of type inference

Given:

1. A process $p$
2. A group environment $\Gamma$

compute a security policy $\Delta^*$ that makes $p$ typable under $\Gamma$.

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

$\Delta^*$ must be minimal w.r.t. $\prec$. 
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 $\Gamma$.

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

The pair of constraint sets is then solved. The solution yields a security policy environment $\Delta$.

We write $(p, \Gamma) \leadsto \Delta$ if $\Delta$ is found from $p$ and $\Gamma$. 
Definition

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

Theorem

*Let \(\Delta^*\) be the minimal security policy environment for the consistent process \(p\) with respect to the group environment \(\Gamma\). It holds that*

1. \(\Delta^*, \Gamma \vdash p \Rightarrow A & e\) for some \(A\) and \(e\).

2. \(\nexists \Delta' : \Delta' < \Delta^* \land \Delta', \Gamma \vdash p \Rightarrow A & e\) 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.
J. Godskesen, T. Hildebrandt and V. Sassone.
A Calculus of Mobile Resources. CONCUR’02, 2002.

A. Gordon and L. Cardelli. Mobile Ambients

M. Hennessy and J. Riely

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