

FLOW LOGIC

Flemming Nielson & Hanne Riis Nielson

Informatics & Mathematical Modelling

Technical University of Denmark

GLOBALAN Summerschool August 2006

Flow Logic

Covered today:

- [Part 1](#): An Introduction based on Mobile Ambients
- [Part 2](#): Executive Summary ([see Part 6 for details](#))
- [Part 3](#): An Application to Firewalls in Mobile Ambients

Covered in detail in the papers:

- [Part 4](#): Discretionary Access Control in Mobile Ambients
- [Part 5](#): Mandatory Access Control in Mobile Ambients
- [Part 6](#): A Multi-Paradigmatic Approach to Static Analysis

Suggested reading:

- Part 1, 3, 4, 5: H. Riis Nielson, F. Nielson, and M. Buchholtz: **Security for Mobility**. In *Foundations of Security Analysis and Design II*, LNCS 2946, pages 207 – 266, Springer, 2004.
- Part 6: H. Riis Nielson and F. Nielson. **Flow Logic: a multi-paradigmatic approach to static analysis**. In *The Essence of Computation: Complexity, Analysis, Transformation*, LNCS 2566, pages 223 - 244, Springer, 2002.

You can find these papers on the GLOBAN CD.

FLOW LOGIC

Part 1:

An Introduction based on
Mobile Ambients

Flemming Nielson & Hanne Riis Nielson

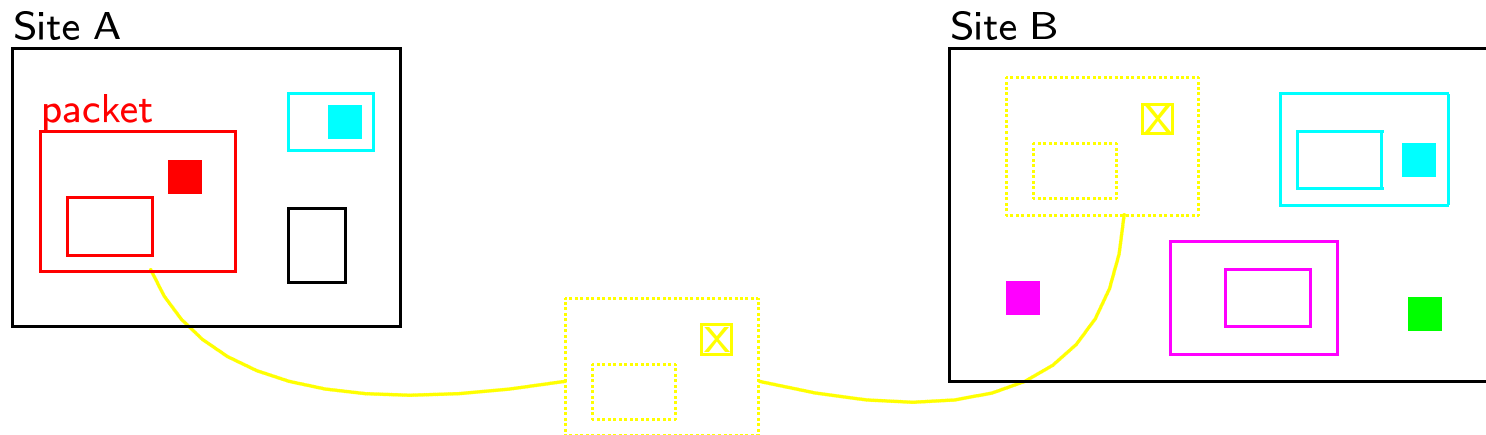
Mobile Ambients

- **Overview:** the ambient view of computation
- **Syntax:** processes and capabilities
- **Semantics:** structural congruence and transition relation

The ambient view of computation

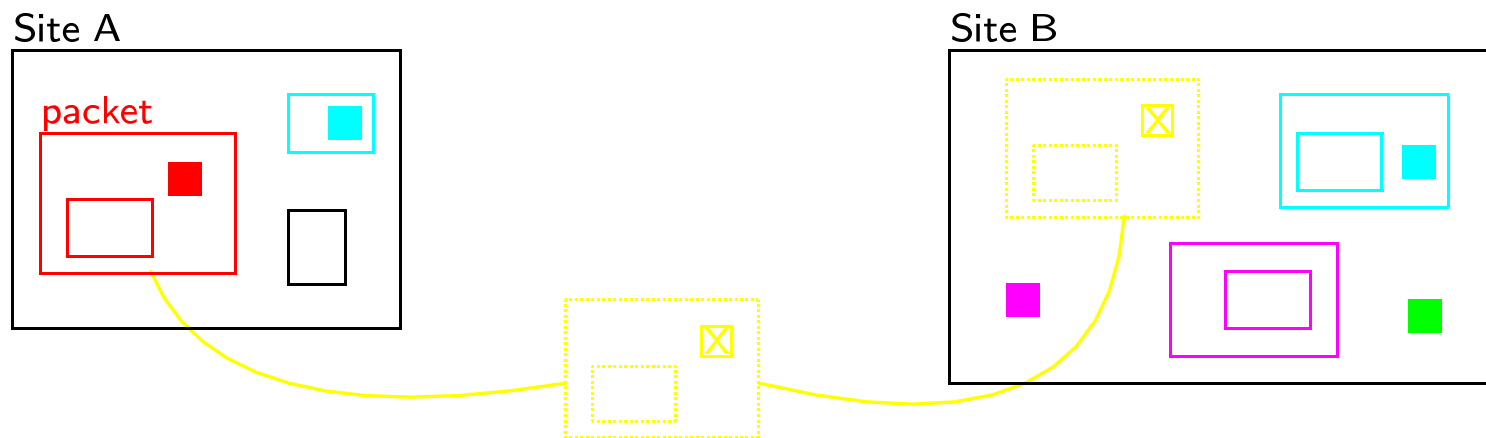
An ambient is a bounded place where computations take place

- the boundary determines what is inside and what is outside
 - as such it is a **high-level security abstraction**
- the ambient moves as a whole
- example ambients: applets, agents, laptops, ...



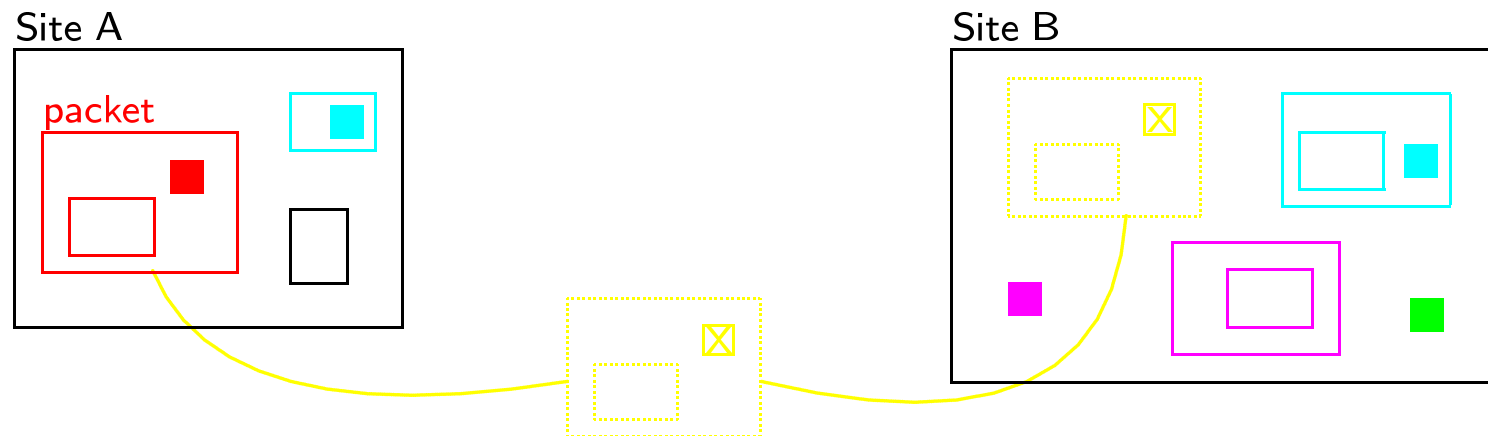
Ambients can be nested inside other ambients forming a tree structure

- mobility is represented as navigation within this hierarchy of ambients
- example: to move a packet from one site to another we must first remove it from the enclosing ambient and then insert it in the new enclosing ambient



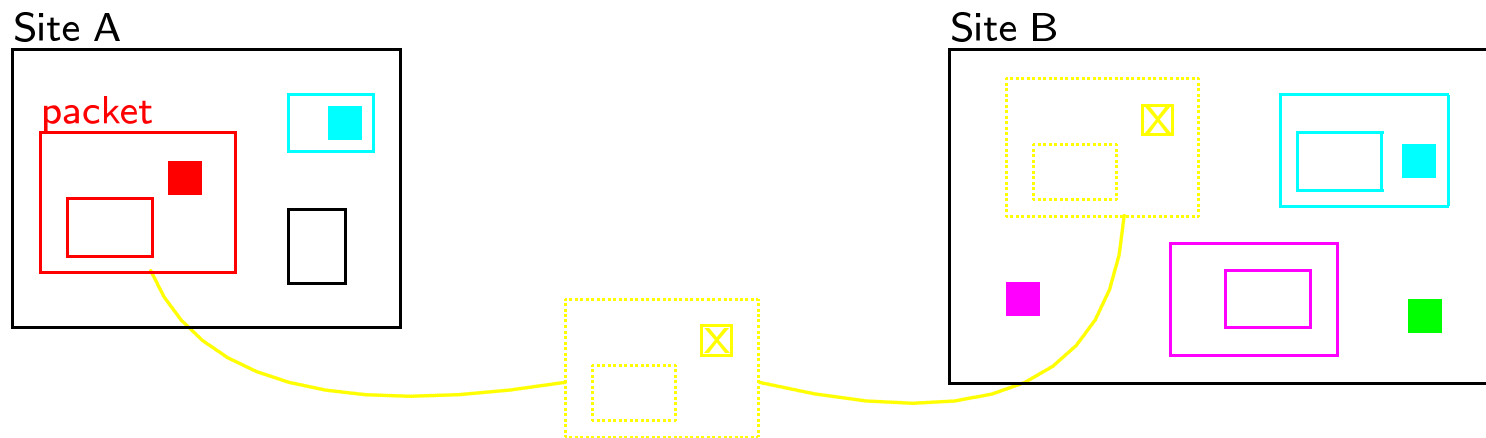
Each ambient contains a number of multi-threaded running processes

- the top-level processes of an ambient have direct control over it and can instruct it to move and thereby change the future behaviour of its processes and subambients
- the processes of subambients have no control over the parent
- processes continue running while being moved



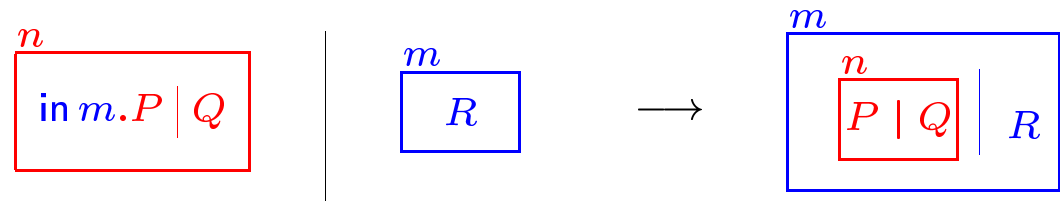
Each ambient has a name

- only the name can be used to control the access to the ambient: entry, exit, communication, etc.
- ambient names are unforgeable

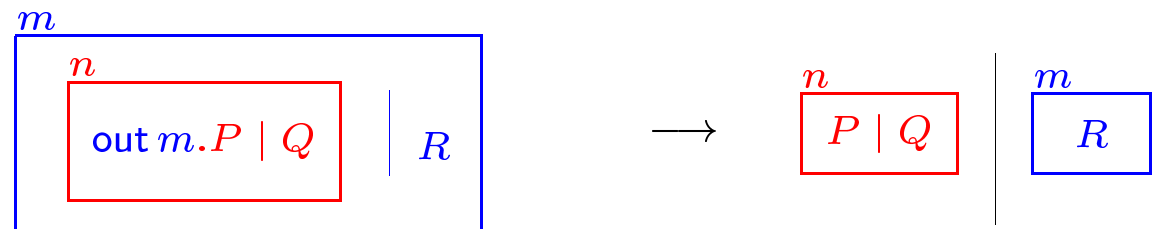


Mobility primitives

Move into an ambient:



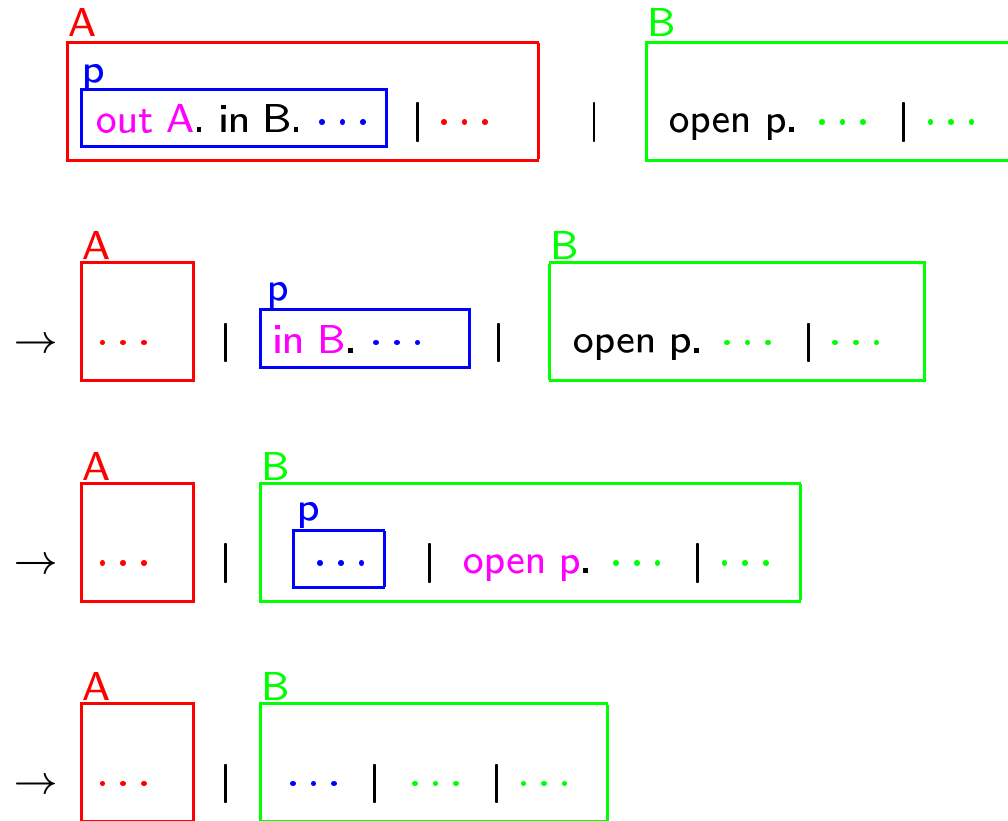
Move out of an ambient:



Dissolve an ambient:



Example: a packet on a network



the packet p moves out of site A

... and then into site B

... where it is dissolved

Example: kidnapping an ambient

$$(\nu k)(\boxed{}^k \mid \text{go (in } n\text{)}. \text{in } k) \mid \boxed{P}^n$$

the ambient $\text{in } k$ moves into n

$$\rightarrow (\nu k)(\boxed{}^k \mid \boxed{\text{in } k \mid P}^n)$$


the ambient n moves into k

$$\rightarrow (\nu k)(\boxed{\boxed{P}^n}^k)$$

the name k is private so nobody can interact with n

Syntax of Mobile Ambients

- Processes — based on the π -calculus

$P ::= (\nu n: \mu) P$	introduces a process with private name n in group μ
$(\nu \mu) P$	introduces a new group named μ with its scope
0	the inactive process
$P_1 \mid P_2$	two concurrent processes
$!P$	any number of occurrences of P
$n[P]$	an ambient named n 
$M.P$	a capability M followed by P

- Capabilities — of the core calculus

$M ::= \text{in } n$	move the enclosing ambient into a sibling named n
$\text{out } n$	move the enclosing ambient out of a parent named n
$\text{open } n$	dissolve a sibling ambient named n

Semantics of Mobile Ambients

- Structural congruence relation: $P \equiv Q$

Examples: $!P \equiv !P \mid P$

$$P \equiv Q \Rightarrow n[P] \equiv n[Q]$$

- Transition relation: $P \rightarrow Q$

Examples: $n[\text{in } m.P \mid Q] \mid m[R] \rightarrow m[n[P \mid Q] \mid R]$

$$P \equiv P' \wedge P' \rightarrow Q' \wedge Q' \equiv Q \Rightarrow P \rightarrow Q$$

Structural congruence relation

$$P \equiv P$$

$$P \equiv Q \wedge Q \equiv R \Rightarrow P \equiv R$$

$$P \equiv Q \Rightarrow Q \equiv P$$

$$P \equiv Q \Rightarrow (\nu n: \mu) P \equiv (\nu n: \mu) Q$$

$$P \equiv Q \Rightarrow (\nu \mu) P \equiv (\nu \mu) Q$$

$$P \equiv Q \Rightarrow P \mid R \equiv Q \mid R$$

$$P \equiv Q \Rightarrow !P \equiv !Q$$

$$P \equiv Q \Rightarrow n[P] \equiv n[Q]$$

$$P \equiv Q \Rightarrow m.P \equiv m.Q$$

$$P \mid Q \equiv Q \mid P$$

$$(P \mid Q) \mid R \equiv P \mid (Q \mid R)$$

$$P \mid \mathbf{0} \equiv P$$

$$!P \equiv P \mid !P$$

$$!\mathbf{0} \equiv \mathbf{0}$$

$$(\nu n: \mu) \mathbf{0} \equiv \mathbf{0}$$

$$(\nu \mu) \mathbf{0} \equiv \mathbf{0}$$

$$(\nu n: \mu) (\nu n': \mu') P \equiv (\nu n': \mu') (\nu n: \mu) P \quad \text{if } n \neq n'$$

$$(\nu \mu) (\nu \mu') P \equiv (\nu \mu') (\nu \mu) P$$

$$(\nu n: \mu) (\nu \mu') P \equiv (\nu \mu') (\nu n: \mu) P \quad \text{if } \mu \neq \mu'$$

$$(\nu n: \mu) (P \mid Q) \equiv P \mid (\nu n: \mu) Q \quad \text{if } n \notin \underline{\text{fn}}(P)$$

$$(\nu \mu) (P \mid Q) \equiv P \mid (\nu \mu) Q \quad \text{if } \mu \notin \underline{\text{fg}}(P)$$

$$(\nu n': \mu) (n[P]) \equiv n[(\nu n': \mu) P] \quad \text{if } n \neq n'$$

$$(\nu \mu) (n[P]) \equiv n[(\nu \mu) P]$$

$$(\nu n: \mu) P \equiv (\nu n': \mu) (P\{n \leftarrow n'\}) \quad \text{if } n' \notin \underline{\text{fn}}(P)$$

$$\text{AVOID } (\nu \mu) P \equiv (\nu \mu') (P\{\mu \leftarrow \mu'\}) \quad \text{if } \mu' \notin \underline{\text{fg}}(P)$$

Transition relation

$$P \rightarrow Q \Rightarrow (\nu n: \mu) P \rightarrow (\nu n: \mu) Q$$

$$P \rightarrow Q \Rightarrow (\nu \mu) P \rightarrow (\nu \mu) Q$$

$$P \rightarrow Q \Rightarrow n[P] \rightarrow n[Q]$$

$$P \rightarrow Q \Rightarrow P \mid R \rightarrow Q \mid R$$

$$P \equiv P' \wedge P' \rightarrow Q' \wedge Q' \equiv Q \Rightarrow P \rightarrow Q$$

$$n[\text{in } m.P \mid Q] \mid m[R] \rightarrow m[n[P \mid Q] \mid R]$$

$$m[n[\text{out } m.P \mid Q] \mid R] \rightarrow n[P \mid Q] \mid m[R]$$

$$\text{open } n.P \mid n[Q] \rightarrow P \mid Q$$

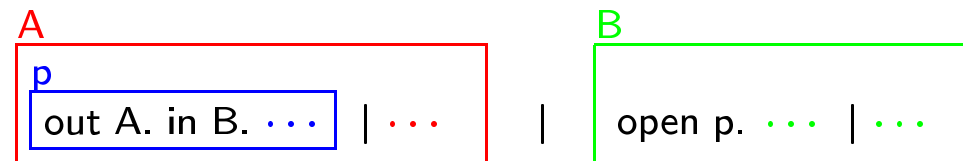
Static Analysis for Mobile Ambients

- The aims of the analysis
- The nature of approximation

The aims of the analysis

1. Which **ambients** may turn up inside other ambients during the execution?
2. Which **capabilities** might be possessed by an ambient during the execution?

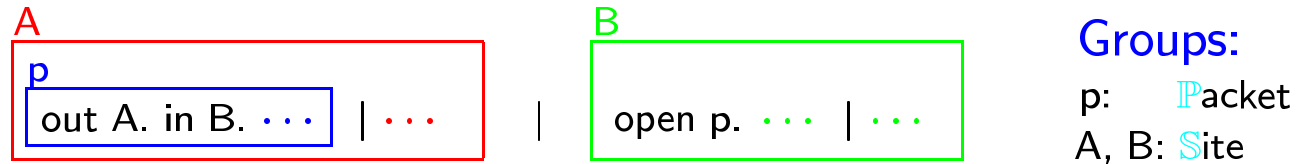
Example:



The exact answer (for 1):

- p will turn up inside A — holds initially
 - p will turn up inside B — holds after two steps
- but
- A and B never turns up inside p

The analysis in terms of groups



The exact answer:

- p may turn up inside A
- p may turn up inside B
- but
- A and B never turns up inside p

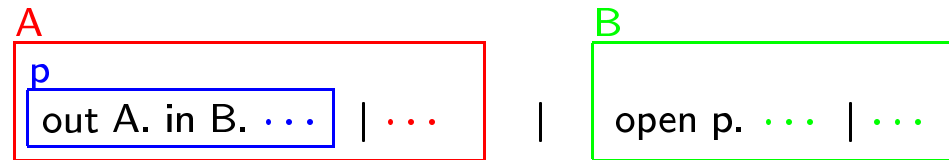
The analysis result with groups:

- \mathbb{P} may turn up inside \mathbb{S}
- but
- \mathbb{S} never turns up inside \mathbb{P}

Assumptions

- groups cannot be renamed — they carry the analysis information
- groups are only allowed at the top-level — they are global

The nature of approximation



An acceptable and precise analysis result

- \mathbb{P} may turn up inside \mathbb{S}
- but
- \mathbb{S} never turns up inside \mathbb{P}

An acceptable but imprecise analysis result

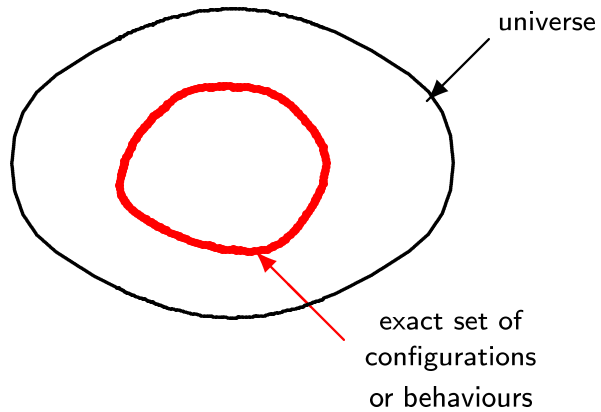
- \mathbb{P} may turn up inside \mathbb{S}
- \mathbb{S} may turn up inside \mathbb{P}

An unacceptable analysis result

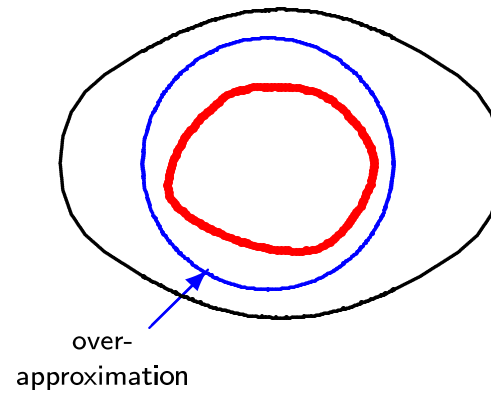
- \mathbb{S} may turn up inside \mathbb{P}
- but
- \mathbb{P} never turns up inside \mathbb{S}

The nature of approximation

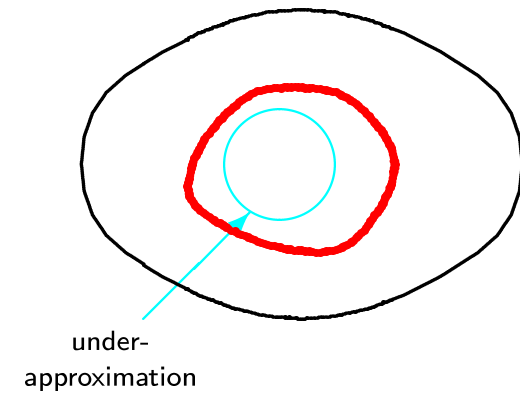
The exact world



Over-approximation



Under-approximation



Our analysis of mobile ambients: over-approximation

Flow Logic for Mobile Ambients

- Analysis estimates
- Analysis judgements
- The Flow Logic approach
- Syntax-directed definition
 - of the analysis of processes
 - of the analysis of capabilities: in, out, and open

Analysis estimates

The analysis estimate

$$\mathcal{I} : \mathbf{Group} \rightarrow \mathcal{P}(\mathbf{Group} \cup \mathbf{Cap})$$

tells us for each ambient group $\mu \in \mathbf{Group}$:

- which ambient groups may be inside an ambient in group μ
- which group capabilities may be possessed by an ambient in group μ

A group capability $M \in \mathbf{Cap}$ is given by

$$M ::= \text{in } \mu \mid \text{out } \mu \mid \text{open } \mu$$

Analysis judgements

$$\mathcal{I} \models_{\Gamma}^{\mu} P$$

means that

$$\mathcal{I} : \mathbf{Group} \rightarrow \mathcal{P}(\mathbf{Group} \cup \mathbf{Cap})$$

is an acceptable analysis estimate for the process P when it occurs inside an ambient from the group μ and when the ambients are in the groups specified by the group environment

$$\Gamma : \mathbf{Name} \rightarrow \mathbf{Group}$$

Hence $\mathcal{I} \models_{\Gamma}^{\mu} P$ is either **true** or **false**.

Example

$$\mathcal{I} \models_{\Gamma}^* A [p [\text{out } A. \text{ in } B]] \mid B [\text{open } p]$$

holds for

Γ	Name \rightarrow Group
A	\mathcal{S}
B	\mathcal{S}
p	\mathcal{P}

\mathcal{I}	Group $\rightarrow \mathcal{P}(\text{Group} \cup \text{Cap})$
\star	$\{\mathcal{S}, \mathcal{P}\}$
\mathcal{S}	$\{\mathcal{P}, \mathcal{S}, \text{in } \mathcal{S}, \text{out } \mathcal{S}, \text{open } \mathcal{P}\}$
\mathcal{P}	$\{\text{in } \mathcal{S}, \text{out } \mathcal{S}\}$

The analysis result shows that

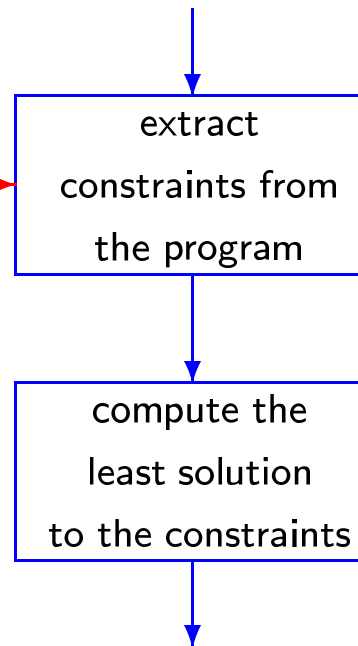
- \mathcal{P} may turn up inside \mathcal{S} — and so may \mathcal{S}
- \mathcal{S} will never turn up inside \mathcal{P} — and neither will \mathcal{P}

The Flow Logic approach

Specification

clauses specifying
acceptability of
analysis estimates

Implementation



Acceptable analysis results

- Each acceptable analysis estimate for a composite program must also be an acceptable analysis estimate for its sub-programs; perhaps more imprecise than need be.
- Each acceptable analysis estimate must mimick the semantics: if the semantics maps one configuration into another then it must be reflected in the analysis estimate.

syntax directed
analysis of processes

analysis of
capabilities

Analysis of processes

$\mathcal{I} \models_{\Gamma}^{\star} (\nu n: \mu) P$ iff $\mathcal{I} \models_{\Gamma[n \mapsto \mu]}^{\star} P$ update group environment; check process

$\mathcal{I} \models_{\Gamma}^{\star} (\nu \mu) P$ iff $\mathcal{I} \models_{\Gamma}^{\star} P$ check process

$\mathcal{I} \models_{\Gamma}^{\star} \mathbf{0}$ iff true nothing to check

$\mathcal{I} \models_{\Gamma}^{\star} P_1 \mid P_2$ iff $\mathcal{I} \models_{\Gamma}^{\star} P_1 \wedge \mathcal{I} \models_{\Gamma}^{\star} P_2$ check both branches

$\mathcal{I} \models_{\Gamma}^{\star} !P$ iff $\mathcal{I} \models_{\Gamma}^{\star} P$ check process; ignore multiplicity

$\mathcal{I} \models_{\Gamma}^{\star} n[P]$ iff $\mu \in \mathcal{I}(\star) \wedge \mathcal{I} \models_{\Gamma}^{\mu} P$ μ is inside \star ; check process
 where $\mu = \Gamma(n)$

Example: subambient

Checking

$$\mathcal{I} \models_{\Gamma}^{\star} A [p [\text{out } A. \text{ in } B]] \mid B [\text{open } p]$$

involves checking

$$\mathcal{S} \in \mathcal{I}(\star) \text{ and } \mathcal{I} \models_{\Gamma}^{\mathcal{S}} p [\text{out } A. \text{ in } B]$$

Γ	Name \rightarrow Group
A	\mathcal{S}
B	\mathcal{S}
p	\mathcal{P}

\mathcal{I}	Group $\rightarrow \mathcal{P}(\text{Group} \cup \text{Cap})$
\star	$\{\mathcal{S}, \mathcal{P}\}$
\mathcal{S}	$\{\mathcal{P}, \mathcal{S}, \text{in } \mathcal{S}, \text{out } \mathcal{S}, \text{open } \mathcal{P}\}$
\mathcal{P}	$\{\text{in } \mathcal{S}, \text{out } \mathcal{S}\}$

Analysis of in-capability

$\mathcal{I} \models_{\Gamma}^* \text{in } n. P$ iff $\text{in } \mu \in \mathcal{I}(\star) \wedge \mathcal{I} \models_{\Gamma}^* P \wedge$

$\forall \mu^a, \mu^p : \text{in } \mu \in \mathcal{I}(\mu^a) \wedge$

$\mu^a \in \mathcal{I}(\mu^p) \wedge$

$\mu \in \mathcal{I}(\mu^p)$

$\Rightarrow \mu^a \in \mathcal{I}(\mu)$

μ^a has the capability in μ

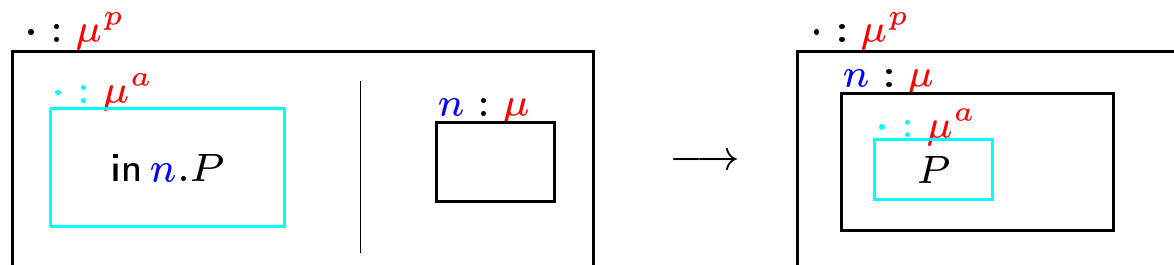
μ^p is parent of μ^a

μ^a has a sibling in group μ

μ^a may move into μ

where $\mu = \Gamma(n)$

Mimicking the semantics:



Example: in-capability

Checking

$$\mathcal{I} \models_{\Gamma}^{\star} A [p [\text{out } A. \text{ in } B]] \mid B [\text{open } p]$$

involves checking

$$\mathcal{I} \models_{\Gamma}^{\mathbb{P}} \text{in } B$$

Γ	Name \rightarrow Group
A	\mathbb{S}
B	\mathbb{S}
p	\mathbb{P}

\mathcal{I}	Group $\rightarrow \mathcal{P}(\text{Group} \cup \text{Cap})$
\star	$\{\mathbb{S}, \mathbb{P}\}$
\mathbb{S}	$\{\mathbb{P}, \mathbb{S}, \text{in } \mathbb{S}, \text{out } \mathbb{S}, \text{open } \mathbb{P}\}$
\mathbb{P}	$\{\text{in } \mathbb{S}, \text{out } \mathbb{S}\}$

which holds because $\text{in } \mathbb{S} \in \mathcal{I}(\mathbb{P})$ and

$$\text{in } \mathbb{S} \in \mathcal{I}(\mu^a) \wedge \mu^a \in \mathcal{I}(\mu^p) \wedge \mathbb{S} \in \mathcal{I}(\mu^p) \Rightarrow \mu^a \in \mathcal{I}(\mathbb{S})$$

holds for all $(\mu^a, \mu^p) \in \{(\mathbb{S}, \star), (\mathbb{S}, \mathbb{S}), (\mathbb{P}, \star), (\mathbb{P}, \mathbb{S})\}$

Analysis of out-capability

$\mathcal{I} \models_{\Gamma}^* \text{out } n. P \text{ iff } \text{out } \mu \in \mathcal{I}(\star) \wedge \mathcal{I} \models_{\Gamma}^* P \wedge$

$\forall \mu^a, \mu^g : \text{out } \mu \in \mathcal{I}(\mu^a) \wedge$

μ^a has the capability out μ

$\mu^a \in \mathcal{I}(\mu) \wedge$

μ is parent of μ^a

$\mu \in \mathcal{I}(\mu^g)$

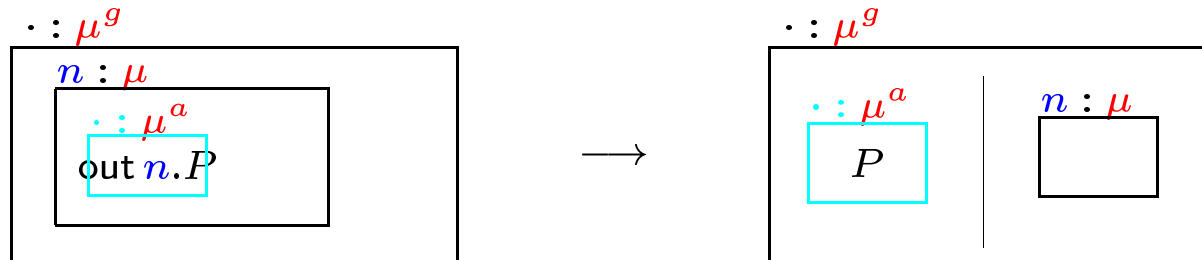
μ^g is grandparent of μ

$\Rightarrow \mu^a \in \mathcal{I}(\mu^g)$

μ^a may move out of μ

where $\mu = \Gamma(n)$

Mimicking the semantics:



Example: out-capability

Checking

$$\mathcal{I} \models_{\Gamma}^{\star} A [p [\text{out } A. \text{ in } B]] \mid B [\text{open } p]$$

involves checking

$$\mathcal{I} \models_{\Gamma}^{\mathbb{P}} \text{out } A. \text{ in } B$$

Γ	Name \rightarrow Group
A	\mathbb{S}
B	\mathbb{S}
p	\mathbb{P}

\mathcal{I}	Group $\rightarrow \mathcal{P}(\text{Group} \cup \text{Cap})$
\star	$\{\mathbb{S}, \mathbb{P}\}$
\mathbb{S}	$\{\mathbb{P}, \mathbb{S}, \text{in } \mathbb{S}, \text{out } \mathbb{S}, \text{open } \mathbb{P}\}$
\mathbb{P}	$\{\text{in } \mathbb{S}, \text{out } \mathbb{S}\}$

which holds because $\mathcal{I} \models_{\Gamma}^{\mathbb{P}} \text{in } B$ and $\text{out } \mathbb{S} \in \mathcal{I}(\mathbb{P})$ and

$$\text{out } \mathbb{S} \in \mathcal{I}(\mu^a) \wedge \mu^a \in \mathcal{I}(\mathbb{S}) \wedge \mathbb{S} \in \mathcal{I}(\mu^g) \Rightarrow \mu^a \in \mathcal{I}(\mu^g)$$

holds for all $(\mu^a, \mu^g) \in \{(\mathbb{S}, \star), (\mathbb{S}, \mathbb{S}), (\mathbb{P}, \star), (\mathbb{P}, \mathbb{S})\}$

Analysis of open-capability

$\mathcal{I} \models_{\Gamma}^* \text{open } n. P$ iff $\text{open } \mu \in \mathcal{I}(\star) \wedge \mathcal{I} \models_{\Gamma}^* P \wedge$

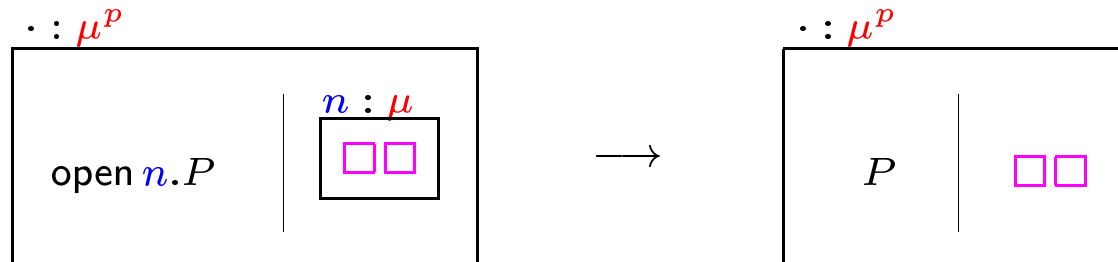
$\forall \mu^p : \text{open } \mu \in \mathcal{I}(\mu^p) \wedge$ μ^p has the capability open μ

$\mu \in \mathcal{I}(\mu^p)$ μ is sibling

$\Rightarrow \mathcal{I}(\mu) \subseteq \mathcal{I}(\mu^p)$ anything in μ may appear in μ^p

where $\mu = \Gamma(n)$

Mimicking the semantics:



Example: open-capability

Checking

$$\mathcal{I} \models_{\Gamma}^* A [p [\text{out } A. \text{ in } B]] \mid B [\text{open } p]$$

involves checking

$$\mathcal{I} \models_{\Gamma}^{\mathcal{S}} \text{open } p$$

Γ	Name \rightarrow Group
A	\mathcal{S}
B	\mathcal{S}
p	\mathcal{P}

\mathcal{I}	Group $\rightarrow \mathcal{P}(\text{Group} \cup \text{Cap})$
*	$\{\mathcal{S}, \mathcal{P}\}$
\mathcal{S}	$\{\mathcal{P}, \mathcal{S}, \text{in } \mathcal{S}, \text{out } \mathcal{S}, \text{open } \mathcal{P}\}$
\mathcal{P}	$\{\text{in } \mathcal{S}, \text{out } \mathcal{S}\}$

which holds because $\text{open } \mathcal{P} \in \mathcal{I}(\mathcal{S})$ and

$$\text{open } \mathcal{P} \in \mathcal{I}(\mu^{\mathcal{P}}) \wedge \mathcal{P} \in \mathcal{I}(\mathcal{S}) \Rightarrow \mathcal{I}(\mathcal{P}) \subseteq \mathcal{I}(\mu^{\mathcal{P}})$$

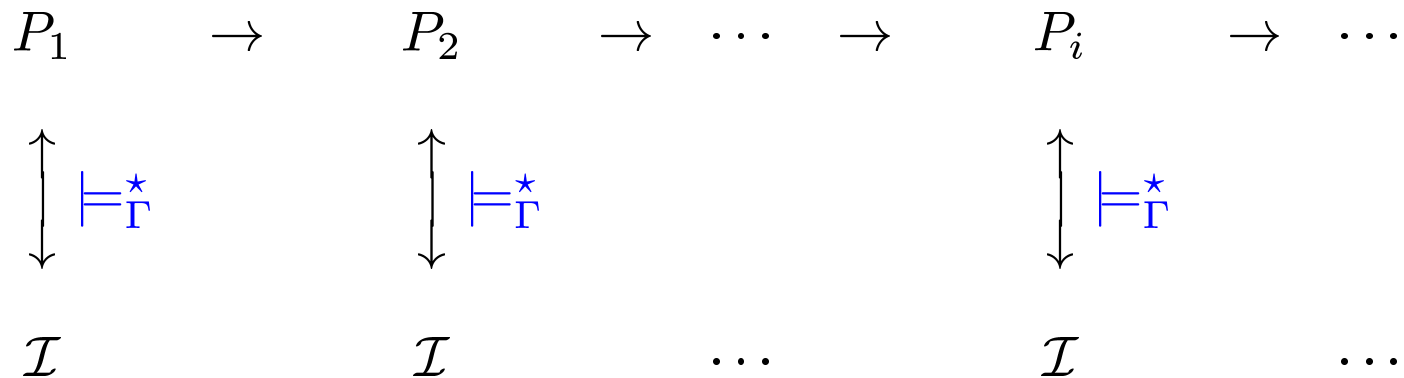
holds for $\mu^{\mathcal{P}} = \mathcal{S}$

Properties of the analysis

- Semantic correctness
 - We err on the safe side!
- Moore family property
 - We have best analysis results!

Semantic correctness

The analysis estimate is preserved during the execution:



Subject reduction result:

If $\mathcal{I} \models_{\Gamma}^* P$ and $P \rightarrow^* Q$ then $\mathcal{I} \models_{\Gamma}^* Q$

Familiar from type systems

Proof of subject reduction result

Lemma: The analysis is invariant under the structural congruence:

If $P \equiv Q$ then $\mathcal{I} \models_{\Gamma}^* P$ if and only if $\mathcal{I} \models_{\Gamma}^* Q$

The proof is by induction on the inference of $P \equiv Q$

Theorem: The analysis is preserved under the transition relation:

If $P \rightarrow Q$ and $\mathcal{I} \models_{\Gamma}^* P$ then $\mathcal{I} \models_{\Gamma}^* Q$

The proof is by induction on the inference of $P \rightarrow Q$

Moore family property

All processes can be analysed and has a least (best) analysis result:

The set $\{\mathcal{I} \mid \mathcal{I} \models_{\Gamma}^* P\}$ is a Moore family

If $\mathcal{Y} \subseteq \{\mathcal{I} \mid \mathcal{I} \models_{\Gamma}^* P\}$ then $\sqcap \mathcal{Y} \models_{\Gamma}^* P$ where $(\sqcap \mathcal{Y})(\mu) = \bigcap \{\mathcal{I}(\mu) \mid \mathcal{I} \in \mathcal{Y}\}$

The proof is by structural induction on P .

Corollaries:

- All processes can be analysed:
 - a Moore family cannot be empty
- All processes has a least (best) analysis result:
 - a Moore family has a least element

FLOW LOGIC

Part 2:

Executive Summary

Flemming Nielson & Hanne Riis Nielson

The motivation for Flow Logic

- There are many approaches to static analysis, e.g.
 - data flow analysis
 - abstract interpretation
 - constraint based analysis
 - type systems
- But unfortunately
 - the approaches are developed in different communities, and
 - insights obtained in one are seldom used by others.
- Flow Logic is an attempt to bridge the gap.

The setting for Flow Logic

- Programs P in some programming language or process algebra
- Semantics $P \rightarrow P'$ often in the form of a Structural Operational Semantics or Reaction Semantics
- Analysis estimates \mathcal{I} usually elements of some complete lattice (L, \sqsubseteq)
- A validity judgement $\mathcal{I} \models P$ defined by clauses
- A subject reduction result $\mathcal{I} \models P \wedge P \rightarrow P' \Rightarrow \mathcal{I} \models P'$ indicating that the validity is preserved under reduction
- A Moore Family result showing that $\mathcal{I}_* = \sqcap \{ \mathcal{I} \mid \mathcal{I} \models P_* \}$ defines a unique least solution

The validity judgement

- Usually there is one clause for each syntactic construct ϕ of the programming language to which P belongs

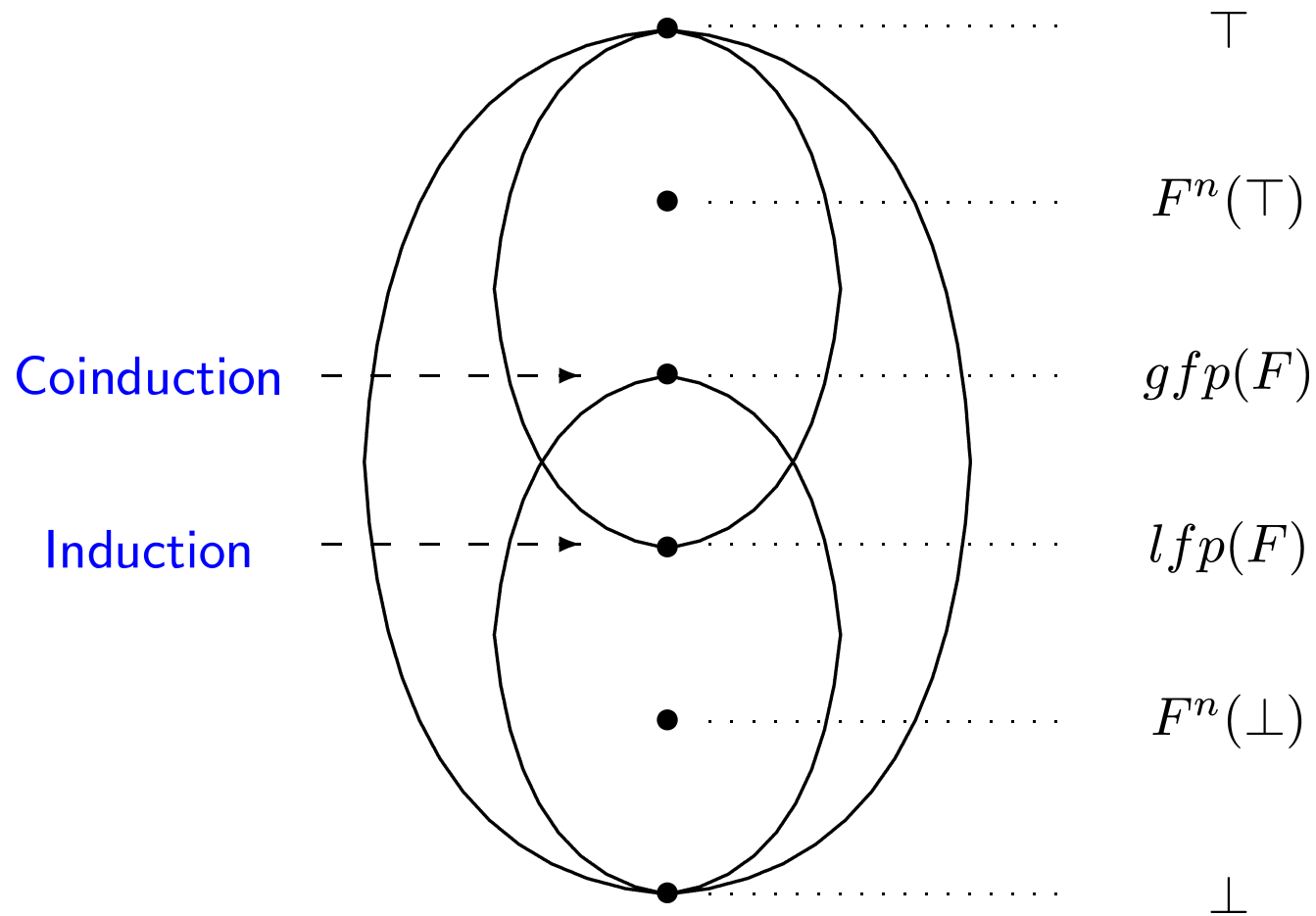
- Each clause takes the form

$$\mathcal{I} \models \phi(\dots P_i \dots) \text{ iff (some formula } \Psi \text{ with } \mathcal{I} \models P')$$

for various subprograms P'

- The first challenge is to ensure that this constitutes a well-defined definition.
 - Sometimes an inductive definition suffices; this is the case if each P' is some P_i (see above).
 - Otherwise a coinductive definition is called for; this works if the formula Ψ gives rise to a monotonic functional F .

The structure of fixed points (Tarski)



Induction versus Coinduction

- Sometimes induction and coinduction agrees. Then a clause

$$\mathcal{I} \models \phi(P_1, P_2) \text{ iff } (\Psi \wedge \mathcal{I} \models P_1 \wedge \mathcal{I} \models P_1)$$

may be written in a more familiar form as an inference rule

$$\frac{\Psi \quad \mathcal{I} \models P_1 \quad \mathcal{I} \models P_2}{\mathcal{I} \models \phi(P_1, P_2)}$$

- When induction and coinduction differs we always take the coinductive definition of the validity judgement.
 - This is not so often the case for process algebras.

Subject reduction result

Lemma: The analysis is invariant under the structural congruence:

If $P \equiv Q$ then $\mathcal{I} \models P$ if and only if $\mathcal{I} \models Q$

The proof is by induction on the inference of $P \equiv Q$:

- Using the iff form of the definition of \models to freely fold and unfold formulae (regardles of which fixpoint used).
- Using a notion of canonical names (invariant under α -conversion) if names are collected in the analysis (\mathcal{I}).
- Sometimes the unfolding of recursion allowed by $A(y) \equiv P[y/x]$ (if we have $A(x) \stackrel{\Delta}{=} P$) creates severe complications!

Theorem: The analysis is preserved under the transition relation:

$$\text{If } P \rightarrow Q \text{ and } \mathcal{I} \models P \text{ then } \mathcal{I} \models Q$$

The proof is by induction on the inference of $P \rightarrow Q$:

- Using the iff form of the definition of \models to freely fold and unfold formulae (regardless of which fixpoint used).
- Being careful with substitution to get placeholder labels to work correctly (if present): $x^l[y^m/x] = y^l$.

Moore family property

Theorem: The set $\{\mathcal{I} \mid \mathcal{I} \models_{\Gamma}^* P\}$ is a Moore family

This means that $\sqcap \mathcal{Y} \models_{\Gamma}^* P$ whenever $\mathcal{Y} \subseteq \{\mathcal{I} \mid \mathcal{I} \models_{\Gamma}^* P\}$ where
 $(\sqcap \mathcal{Y})(\mu) \stackrel{\Delta}{=} \bigcap \{\mathcal{I}(\mu) \mid \mathcal{I} \in \mathcal{Y}\}$

The proof is by structural induction on P .

It follows that $\mathcal{I}_{\star} = \sqcap \{\mathcal{I} \mid \mathcal{I} \models P_{\star}\}$ defines a unique least solution:

- we have $\mathcal{I}_{\star} \models P_{\star}$
- if $\mathcal{I} \models P_{\star}$ then $\mathcal{I}_{\star} \sqsubseteq \mathcal{I}$

Calculating the least solution

Generate $\mathcal{I}_\star = \sqcap\{\mathcal{I} \mid \mathcal{I} \models P_\star\}$ as a formula in a suitable format:

- Conditional constraints — with links to bitvector frameworks.
- Horn Clauses
 - Datalog
 - Alternating Least Fixed Point Logic (*“The Succinct Solver”*)
The asymptotic time to check a solution equals the asymptotic time to calculate a solution.
 - H3, H1, ... solvers

See the talk by Helmut Seidl for detailed information.

Ingredients for more advanced analyses

- Adding context information (from 0-CFA to k-CFA):

$$\mathcal{I} \models_c P \text{ iff } \dots \mathcal{I} \models_{cc'} P' \dots \wedge \dots \{cc'\} \subseteq \mathcal{I}(c) \dots$$

Here c indicates the dynamic context in which the process P is encountered; this is useful for achieving more precise analyses of programming languages and process algebras.

- Making analyses data dependent.

This is mainly used for imperative and object-oriented languages.

Abstract versus Compositional

One of the methodological choices of Flow Logic is when to use abstract and when to use compositional specifications.

- **Abstract:**

- Abstract specifications place no demands on which processes may be used on the right-hand-sides of iff .
- Usually processes communicated (in a higher-order process algebra) and recursive processes are analysed at each invocation.
- This leads to treating open systems for free.
- The coinductive interpretation (desired) may differ from the inductive one.
- Often more pleasant to read and easier to understand.

- **Compositional:**

- Compositional specifications demand that only subprocesses of the process on the left-hand-side of an iff may be used on the right-hand-side.
- Therefore processes communicated (in a higher-order process algebra) and recursive processes must be analysed once and for all at their point of definition.
- Unless special care is taken this leads to analysing closed systems only.
- The coinductive interpretation (desired) agrees with the inductive one.
- Usually necessary in order to implement the analysis.

Verbose versus Succinct

Another methodological choice of Flow Logic is when to use verbose and when to use succinct specifications.

- **Verbose:**

- All the information of interest is collected in a global manner:

$$\mathcal{I}, \mathcal{I}_I, \mathcal{I}_O \models P$$

- Makes some forms of implementation easier to deal with.
- Often requires placeholder labels in the syntax of processes.

- Succinct:

- Only some of the information of interest is collected in a global manner; other pieces of information may be
 - * synthesized to describe the effect of local components:

$$\mathcal{I} \models P : \triangleright \hat{O}$$

which is useful for functional languages and process algebras.

- * record input- and output-dependencies

$$\mathcal{I} \models P : \hat{I} \triangleright \hat{O}$$

which is mainly useful for procedural languages.

- Generally leads to simpler analysis domains.
- Often more pleasant to read and easier to understand.

A word on notation

Some of the notation used has striking resemblance to (can be considered equivalent to) other well-known notations:

- Succinct Compositional Flow Logic

$$\mathcal{I} \models P : \hat{I} \triangleright \hat{O}$$

- Hoare Logic

$$\mathcal{I} \vdash \{\hat{I}\} P \{\hat{O}\}$$

- Extended Attribute Grammar

$$\mathcal{I} \vdash P : \langle \downarrow \hat{I} \uparrow \hat{O} \rangle$$

FLOW LOGIC

Part 3:

An Application to Firewalls
in Mobile Ambients

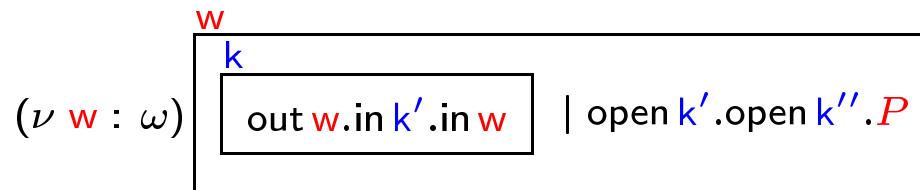
Flemming Nielson & Hanne Riis Nielson

An Application to Firewalls

- A model of a firewall
- Analysing software in an unknown environment
- The hardest attacker

A model of a firewall

To be useful the firewall should allow selected agents to enter the firewall while keeping others out.



Secret name of the firewall: w
Passwords: k, k', k''

- Agents **knowing** the passwords should be allowed to enter the firewall:
 - **easy**: check that agents of a specific form can enter
- Agents **not knowing** the passwords should **not** be allowed enter:
 - **hard**: we cannot inspect all agents

Agents knowing the passwords (I)

The firewall

$$(\nu w : \omega) \boxed{\begin{array}{l} \color{red}{w} \\ \color{blue}{k} \\ \text{out } \color{red}{w} . \text{in } \color{blue}{k}' . \text{in } \color{red}{w} \quad | \quad \text{open } \color{blue}{k}' . \text{open } \color{blue}{k}'' . \color{red}{P} \end{array}}$$

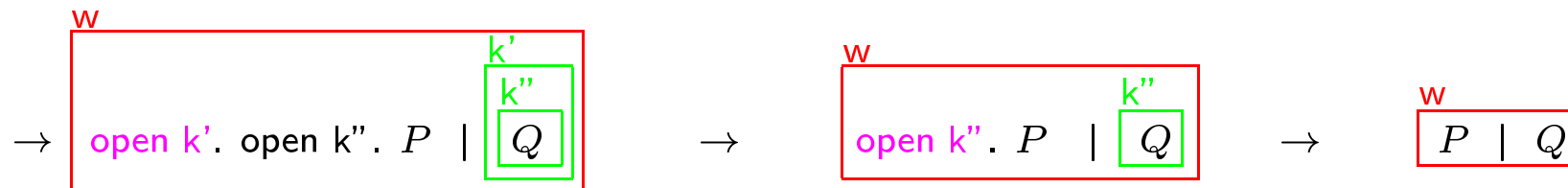
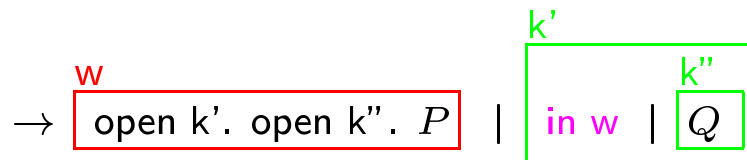
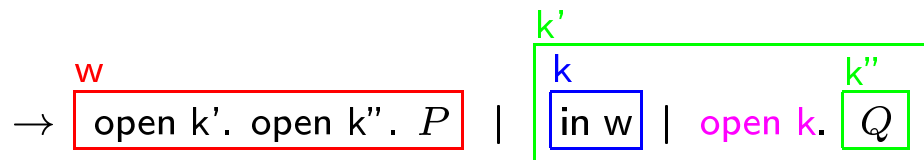
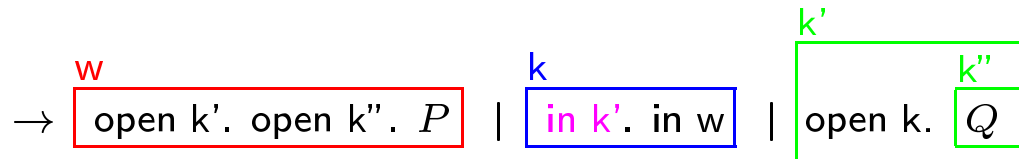
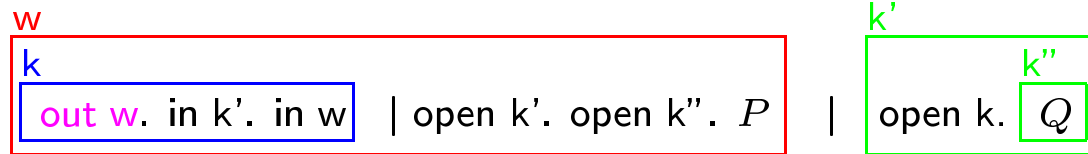
can be entered by agents of the form

$$\color{green}{k}' \boxed{\text{open } \color{green}{k} . \color{green}{k}'' \boxed{Q}}$$

and gives rise to

$$(\nu w : \omega) \boxed{\color{red}{w} \boxed{P} \mid Q}$$

Agents knowing the passwords (II)



The probe is sent out of the firewall to fetch the agent

Agents not knowing the passwords

- an agent that does not initially know the passwords **might learn** them subsequently
- the firewall might contain a **trap door** through which agents might enter

How can we ensure that agents not knowing the passwords cannot enter?

An agent learning the passwords

The agent $\boxed{\text{in } t. R}$

can enter the firewall $(\nu w : \omega) \boxed{\text{out } w. \text{in } k'. \text{in } w} \mid \text{open } k'. \text{open } k''. P$

when placed in a context with $\boxed{\text{open } t. \text{in } k'}$ \mid $\boxed{\text{open } t \mid \text{open } k. \boxed{\text{in } k''. Q}}$

giving rise to $(\nu w : \omega) \boxed{P \mid Q \mid R}$

A firewall with a trap door

The agent $\boxed{\text{in } t. R}$

can enter the firewall

$(\nu w : \omega)$
 $\boxed{\text{out } w. \text{in } k'. \text{in } w}$
 $|$
 $\text{open } k'. \text{open } k''. P$
 $|$
 $\text{open } t$
 $|$
 $\boxed{\text{out } w. \text{in } w. \text{open } e}$
 $|$
 $\boxed{\phantom{\text{out } w. \text{in } w. \text{open } e}}$
 $|$
 $\boxed{\phantom{\text{out } w. \text{in } w. \text{open } e}}$

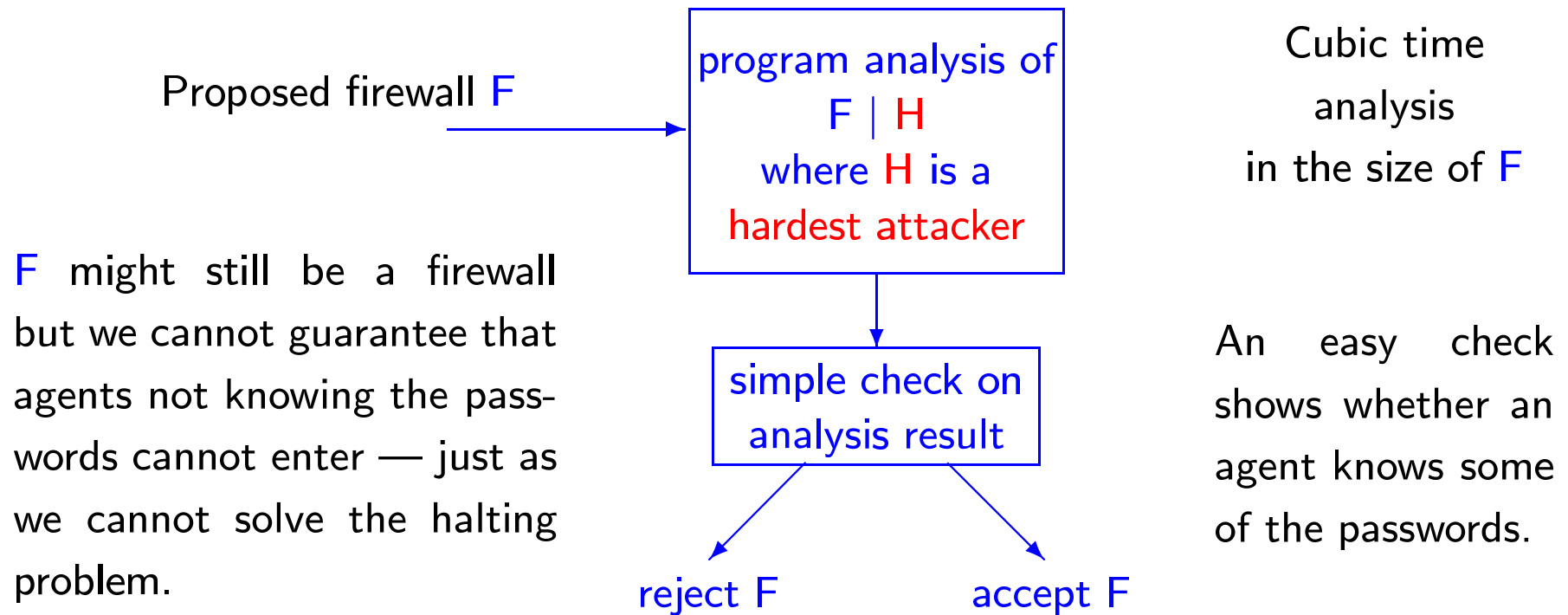
giving rise to

$(\nu w : \omega)$
 $\boxed{\text{out } w. \text{in } k'. \text{in } w}$
 $|$
 P
 $|$
 R

Analysing software in an unknown environment

- **Observation:** it is easy to analyse a **closed** system
- **Problem:** we want to analyse an **open** system where the attacker is unknown
- **Question:** can we identify an attacker that is as hard to protect the system against as any other attacker?
 - and thereby **turn the open system into a closed system**
- **Answer:** yes — **the hardest attacker**
 - if we can guarantee that the hardest attacker cannot enter the firewall then no attacker can ever enter the firewall

The overall idea



The firewall passes the test; the “firewall” with the trap door does not.

Terminology (I)

A **proposed firewall** is given by

$$(\Gamma, (\nu w: \omega) w [F], \kappa)$$

where Γ assigns groups to the finitely many names occurring in $(\nu w: \omega) w [F]$ and κ is the group associated with the set of passwords

An **unaware attacker** (relative to the proposed firewall) is a process U such that

- it neither mentions the name of the firewall nor any of its passwords:

$$\{\Gamma(n) \mid n \in \text{fn}(U)\} \cap \{\omega, \kappa\} = \emptyset$$

OBS: a more refined notion of a proposed firewall may specify the format of a successful agent as $(\Gamma, (\nu w: \omega) w [F], A)$ and then use A to obtain a more restricted definition of an unaware attacker – as an example, U may know the passwords but not how to use them to enter the firewall

Terminology (II)

- **Dynamic property:** A proposed firewall is **protective** if the semantics prevents **any** unaware attacker from entering
- **Static property:** A proposed firewall is **strongly protective** if the analysis can validate that **no** unaware attackers can enter the firewall
- **Corollary of the subject reduction result:**

strongly protective \Rightarrow **protective**

- **Consequence:** it is sufficient to check whether the proposed firewall is strongly protective

Obs: the analysis is approximative so there are protective firewalls that are not strongly protective — they will be rejected by the test!

Key observations

Let $(\Gamma, (\nu w: \omega) w [F], \kappa)$ be a proposed firewall

- an unaware attacker U may mention any of the (finitely many) groups mentioned in Γ except for ω and κ
- we can safely rename all groups in U that are not in Γ to be the distinguished group \top not used elsewhere (a kind of supergroup)
- the analysis estimate \mathcal{I} for $(\nu w: \omega) w [F] \mid U$ is an element of

$$\mathbf{Group} \rightarrow \mathcal{P}(\mathbf{Group} \cup \mathbf{Cap})$$

where $\mathbf{Group} = \{\text{groups mentioned in } \Gamma\} \cup \{\omega, \kappa, \top\}$ is finite!

The hardest attacker

Key observations:

- the set $\mathbf{Group} \rightarrow \mathcal{P}(\mathbf{Group} \cup \mathbf{Cap})$ is **finite**
- the set $\mathbf{Group} \rightarrow \mathcal{P}(\mathbf{Group} \cup \mathbf{Cap})$ contains the analysis estimates for $(\nu w: \omega) w [F] \mid U$ for all the **infinitely many** unaware attackers U

The hardest attacker H is an unaware attacker that can create **all** possible ambients and that possesses **all** possible capabilities **as far as** the analysis can observe within the finite domain $\mathbf{Group} \rightarrow \mathcal{P}(\mathbf{Group} \cup \mathbf{Cap})$

Details of the test

For the hardest attacker H :

- **calculate** the most precise (i.e. least) analysis result:

$$\mathcal{I}_H \models_{\Gamma}^* ((\nu w: \omega) w [F]) \mid H$$

This can be done in cubic time in the size of F

- **check** whether any of the ambients in the group \top (i.e. originating in H) may turn up inside the proposed firewall:

does $\top \in \mathcal{I}_H^+(\omega)$?

- **if yes** then reject the proposed firewall (i.e. it is not strongly protective)
- **if no** then accept the proposed firewall as it is guaranteed to be protective

What does H look like?

$$H = !(\nu n: \top) (P \mid n [P])$$

where

$$P = \text{in } n \mid \text{out } n \mid \text{open } n$$

OBS: if the analysis is changed we also need to change the hardest attacker!

Why does it work?

Suppose U is an unaware attacker and that \mathcal{I}_U is the least analysis result:

$$\mathcal{I}_U \models_{\Gamma}^* ((\nu w: \omega) w [F]) \mid U$$

Then $\mathcal{I}_U \subseteq \mathcal{I}_H$ by “definition” of the hardest attacker H .

So $\top \notin \mathcal{I}_H^+(\omega)$ yields $\top \notin \mathcal{I}_U^+(\omega)$.

(More precisely, the set of constraints needed for calculating \mathcal{I}_U is contained in the set of constraints needed for calculating \mathcal{I}_H .)

FLOW LOGIC

Part 4:

Discretionary Access Control
in Mobile Ambients

Flemming Nielson & Hanne Riis Nielson

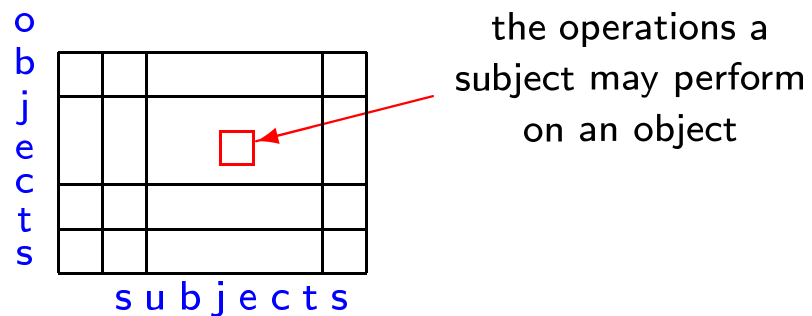
Discretionary Access Control

- Access Control
- Static Validation of Access Control
- Introducing Access Control Primitives
 - safe ambients
 - discretionary ambients
- Adapting the Static Analysis
 - taking co-capabilities into account
 - taking context into account
- Properties of the Analysis

Access Control

An active **subject** accesses a passive **object** with some specific **access operations**.

The **access control matrix** is the traditional way of defining what operations may be performed:



Beware: some presentations use a transposed matrix.

Access Control for Mobility

- **the subjects** are the **ambients** possessing the capabilities: the ambient may move into another ambient, may move out of another ambient or may dissolve another ambient
- **the objects** are the **ambients** that are entered, left or dissolved
- **the access operations** are the **capabilities**

Static Validation of Access Control

Validation of properties of the access operations with respect to an “access control policy”:

- **The program analysis** already approximates the set of access operations that an ambient may perform:
 - **crossing analysis**: may an ambient move into another ambient?
 - **opening analysis**: may an ambient dissolve another ambient?

(We shall see more interesting examples soon.)

Crossing analysis

Property:

The ambient n can cross the ambient n' during the execution of P , i.e.

- n executes the $\text{in } n'$ or the $\text{out } n'$ capability

In terms of groups:

Ambients of group μ can cross ambients in group μ' during the execution of P i.e.

- some n of group μ can cross some ambient n' of group μ'

In the analysis:

Ambients of group μ can cross ambients of group μ' during the execution of P , i.e.

- $\text{in } \mu' \in \mathcal{I}(\mu) \vee \text{out } \mu' \in \mathcal{I}(\mu)$

for the least \mathcal{I} such that $\mathcal{I} \models_{\Gamma}^* P$

Example: a packet on a network

The analysis can be used to validate:

- Ambients of group \mathbb{P} **may** cross ambients in group \mathbb{S}
 - check: $\text{in } \mathbb{S} \in \mathcal{I}(\mathbb{P}) \vee \text{out } \mathbb{S} \in \mathcal{I}(\mathbb{P})$
- Ambients in group \mathbb{S} **will never** cross ambients in group \mathbb{P}
 - check: $\text{in } \mathbb{P} \notin \mathcal{I}(\mathbb{S}) \wedge \text{out } \mathbb{P} \notin \mathcal{I}(\mathbb{S})$

A more precise analysis is needed to validate:

- Ambients of group \mathbb{S} **will never** cross ambients in group \mathbb{S}
 - we do not have: $\text{in } \mathbb{S} \notin \mathcal{I}(\mathbb{S}) \wedge \text{out } \mathbb{S} \notin \mathcal{I}(\mathbb{S})$

Analysis estimate:

Γ	Name \rightarrow Group
A	\mathbb{S}
B	\mathbb{S}
p	\mathbb{P}

\mathcal{I}	Group $\rightarrow \mathcal{P}(\text{Group} \cup \text{Cap})$
*	$\{\mathbb{S}, \mathbb{P}\}$
\mathbb{S}	$\{\mathbb{P}, \mathbb{S}, \text{in } \mathbb{S}, \text{out } \mathbb{S}, \text{open } \mathbb{P}\}$
\mathbb{P}	$\{\text{in } \mathbb{S}, \text{out } \mathbb{S}\}$

Opening analysis

Property:

The ambient n can open the ambient n' during the execution of P , i.e.

- n executes the $\text{open } n'$ capability

In terms of groups:

Ambients of group μ can open ambients in group μ' during the execution of P i.e.

- some n of group μ can open some n' of group μ'

In the analysis:

Ambients of group μ may open ambients in group μ' , i.e.

- $\text{open } \mu' \in \mathcal{I}(\mu)$

for the least \mathcal{I} such that $\mathcal{I} \models_{\Gamma}^* P$

Example: a packet on a network

The analysis can be used to validate:

- Ambients of group \mathcal{S} **may** open ambients in group \mathcal{P}
 - check: $\text{open } \mathcal{P} \in \mathcal{I}(\mathcal{S})$
- Ambients in group \mathcal{P} **will never** open any ambients
 - check: $\forall \mu : \text{open } \mu \notin \mathcal{I}(\mathcal{P})$

Analysis estimate:

Γ	Name \rightarrow Group
A	\mathcal{S}
B	\mathcal{S}
p	\mathcal{P}

\mathcal{I}	Group $\rightarrow \mathcal{P}(\text{Group} \cup \text{Cap})$
*	$\{\mathcal{S}, \mathcal{P}\}$
\mathcal{S}	$\{\mathcal{P}, \mathcal{S}, \text{in } \mathcal{S}, \text{out } \mathcal{S}, \text{open } \mathcal{P}\}$
\mathcal{P}	$\{\text{in } \mathcal{S}, \text{out } \mathcal{S}\}$

Access Control Primitives in Safe Ambients

Movement of ambients can only happen if both parties agree:

- if p wants to move out of A then A should be willing to let ambients leave
 - A must have the capability $\overline{\text{out}} A$
- if p wants to move into B then B should be willing to let ambients enter
 - B must have the capability $\overline{\text{in}} B$
- if B wants to dissolve p then p should be willing to be dissolved
 - p must have the capability $\overline{\text{open}} p$

Safe ambients were introduced by Levi and Sangiorgi at POPL 2000.

Syntax and semantics

$M ::= \text{in } n \mid \text{out } n \mid \text{open } n$ capabilities \approx access operations
 $\mid \overline{\text{in}} \ n \mid \overline{\text{out}} \ n \mid \overline{\text{open}} \ n$ co-capabilities \approx access rights

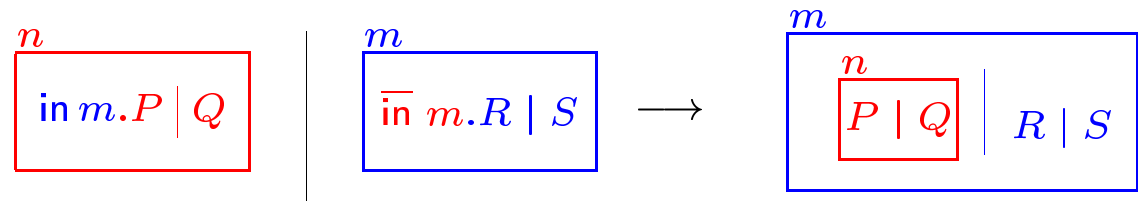
Extensions to the transition relation:

$$n[\text{in } m.P \mid Q] \mid m[\overline{\text{in}} \ m.R \mid S] \rightarrow m[n[P \mid Q] \mid R \mid S]$$
$$m[n[\text{out } m.P \mid Q] \mid \overline{\text{out}} \ m.R \mid S] \rightarrow n[P \mid Q] \mid m[R \mid S]$$
$$\text{open } n.P \mid n[\overline{\text{open}} \ n.Q \mid R] \rightarrow P \mid Q \mid R$$

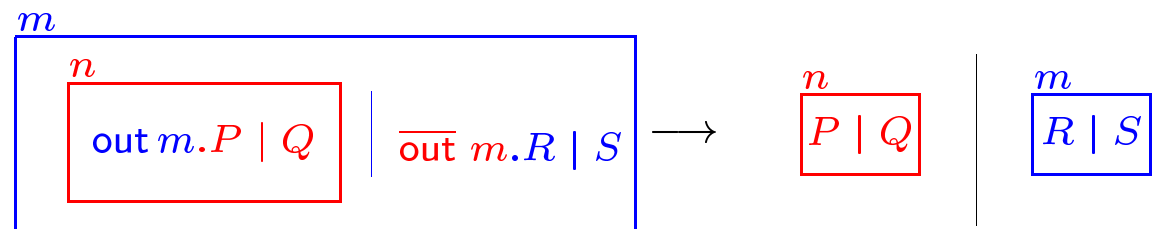
This amounts to integrating the reference monitor into the operational semantics.

In pictures:

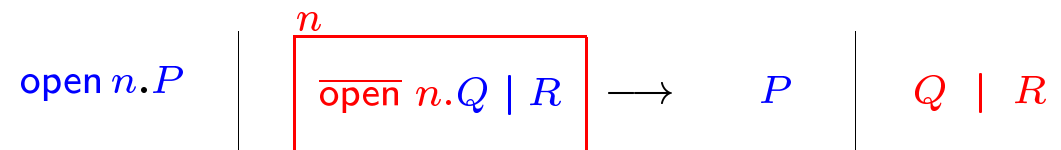
Move into an ambient:



Move out of an ambient:



Dissolve an ambient:



Example: a package on a network

$A [p [\text{out } A. \text{in } B. \overline{\text{open}} p] \mid \overline{\text{out}} A] \mid B [\overline{\text{in}} B. \text{open } p]$

$\rightarrow A [] \mid p [\text{in } B. \overline{\text{open}} p] \mid B [\overline{\text{in}} B. \text{open } p]$

$\rightarrow A [] \mid B [p [\overline{\text{open}} p] \mid \text{open } p]$

$\rightarrow A [] \mid B []$

Safe ambients and access control

A safe ambient process presents a system with a distributed access control matrix that dynamically evolves and that is concerned with multiplicities.

A system with a classical access control matrix uses co-capabilities as in

$$A [p [\text{out } A. \text{ in } B \mid ! \overline{\text{open}} p] \mid ! \overline{\text{out}} A] \mid B [! \overline{\text{in}} B \mid \text{open } p]$$

rather than as in

$$A [p [\text{out } A. \text{ in } B. \overline{\text{open}} p] \mid \overline{\text{out}} A] \mid B [\overline{\text{in}} B. \text{open } p]$$

Weakness of the model of access control

Safe ambients model a very rudimentary kind of access control:

- if n has the co-capability $\overline{\text{in}} n$ then any ambient may enter n

Discretionary ambients models a more general form of access control:

- if n has the co-capability $\overline{\text{in}}_{\mu} n$ then any ambient in group μ may enter n

Example: Safe ambients

subject that may have
the corresponding capability

	A	B	p
object A	$\overline{\text{out}} A$	$\overline{\text{out}} A$	$\overline{\text{out}} A$
object B	$\overline{\text{in}} B$	$\overline{\text{in}} B$	$\overline{\text{in}} B$
object p	$\overline{\text{open}} p$	$\overline{\text{open}} p$	$\overline{\text{open}} p$

rows must be equal

Example: Discretionary ambients

subject that may have
the corresponding capability

	A : \mathcal{S}	B : \mathcal{S}	p : \mathcal{P}
object A	—	—	$\overline{\text{out}}_{\mathcal{P}} A$
object B	—	—	$\overline{\text{in}}_{\mathcal{P}} B$
object p	$\overline{\text{open}}_{\mathcal{S}} p$	$\overline{\text{open}}_{\mathcal{S}} p$	—

Access Control Primitives in Discretionary Ambients

$$\begin{array}{ll}
 M ::= & \text{in } n \mid \text{out } n \mid \text{open } n & \text{access operations} \\
 & \mid \overline{\text{in}}_{\mu} n \mid \overline{\text{out}}_{\mu} n \mid \overline{\text{open}}_{\mu} n & \text{access rights}
 \end{array}$$

Extensions to the transition relation:

$$\Gamma \vdash n[\text{in } m.P \mid Q] \mid m[\overline{\text{in}}_{\mu} m.R \mid S] \rightarrow m[n[P \mid Q] \mid R \mid S] \quad \text{if } \Gamma(n) = \mu$$

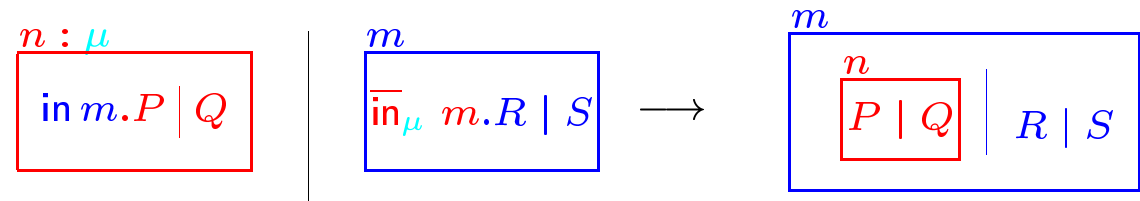
$$\Gamma \vdash m[n[\text{out } m.P \mid Q] \mid \overline{\text{out}}_{\mu} m.R \mid S] \rightarrow n[P \mid Q] \mid m[R \mid S] \quad \text{if } \Gamma(n) = \mu$$

$$\Gamma \vdash n[\text{open } m.P \mid m[\overline{\text{open}}_{\mu} m.Q \mid R]] \rightarrow n[P \mid Q \mid R] \quad \text{if } \Gamma(n) = \mu$$

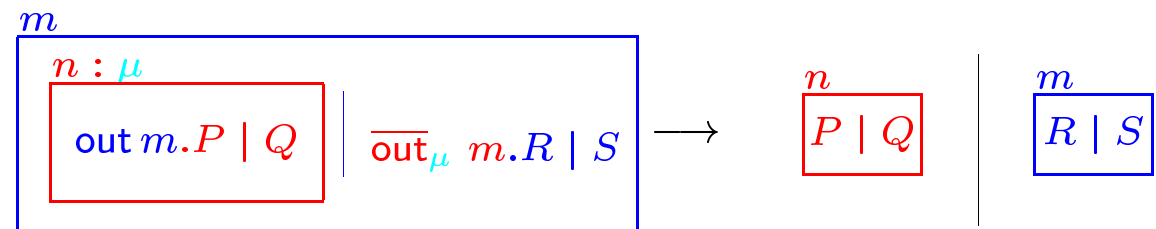
where $\Gamma : \text{Name} \rightarrow \text{Group}$ assigns groups to the names

In pictures:

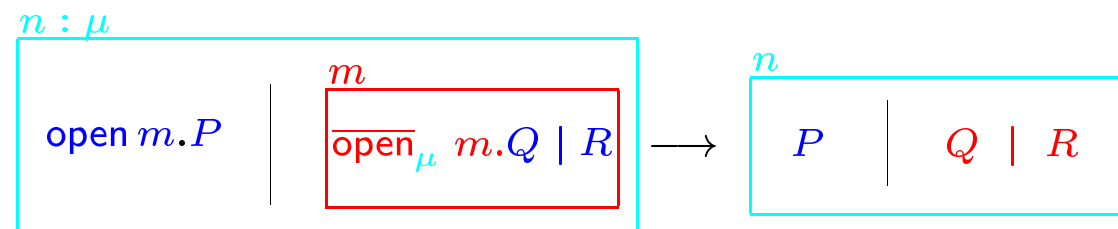
Move into an ambient:



Move out of an ambient:



Dissolve an ambient:



Example: a package on a network

$$\Gamma \vdash A [p [\text{out } A. \text{in } B. \overline{\text{open}}_S p] \mid \overline{\text{out}}_P A] \mid B [\overline{\text{in}}_P B. \text{open } p]$$

$$\rightarrow A [] \mid p [\text{in } B. \overline{\text{open}}_S p] \mid B [\overline{\text{in}}_P B. \text{open } p] \quad \text{because } \Gamma(p) = P$$

$$\rightarrow A [] \mid B [p [\overline{\text{open}}_S p] \mid \text{open } p] \quad \text{because } \Gamma(p) = P$$

$$\rightarrow A [] \mid B [] \quad \text{because } \Gamma(B) = S$$

Adapting the program analysis

For each ambient group $\mu \in \mathbf{Group}$ the analysis estimates

$$\mathcal{I} : \mathbf{Group} \rightarrow \mathcal{P}(\mathbf{Group} \cup \mathbf{Cap} \cup \overline{\mathbf{Cap}})$$

will tell us

- which ambient groups may be inside an ambient in group μ
- which access operations may an ambient in group μ possess (as subject)
- which access rights may an ambient in group μ provide (as object)

Access operations and access rights are given by

access operations $M \in \mathbf{Cap}$ $M ::= \text{in } \mu \mid \text{out } \mu \mid \text{open } \mu$

access rights $\overline{M} \in \overline{\mathbf{Cap}}$ $\overline{M} ::= \overline{\text{in}}_{\mu'} \mu \mid \overline{\text{out}}_{\mu'} \mu \mid \overline{\text{open}}_{\mu'} \mu$

Example: a packet on a network

$$\mathcal{I} \models_{\Gamma}^* A [p [\text{out } A. \text{in } B. \overline{\text{open}}_S p] \mid \overline{\text{out}}_P A] \mid B [\overline{\text{in}}_P B. \text{open } p]$$

will hold for

Γ	Name \rightarrow Group
A	S
B	S
p	P

\mathcal{I}	Group $\rightarrow \mathcal{P}(\text{Group} \cup \text{Cap} \cup \overline{\text{Cap}})$
*	{S, P}
S	{P, open P, $\overline{\text{in}}_P S$, $\overline{\text{out}}_P S$, in S, out S, $\overline{\text{open}}_S P$ }
P	{in S, out S, $\overline{\text{open}}_S P$ }

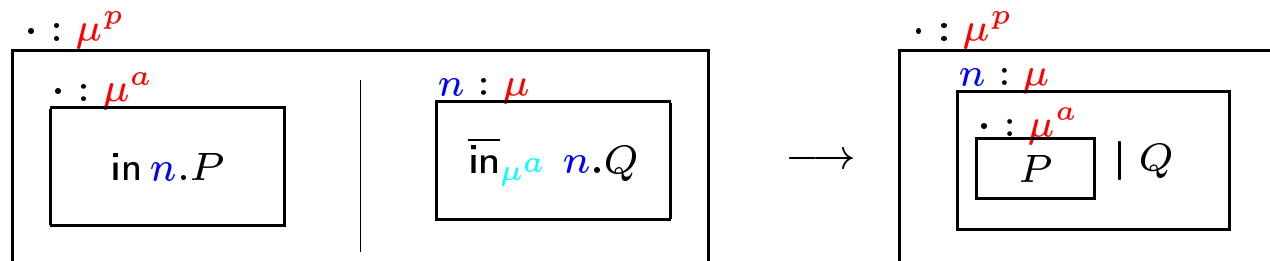
The analysis can be used to validate that ambients in group S never cross ambients in group S

- this property could **not** be validated for ambients without using the more powerful analysis based on sets of configurations
- we cannot validate it for safe ambients either (with the simple analysis)

Analysis of in-capability

$$\begin{aligned}
 \mathcal{I} \models_{\Gamma}^* \text{in } n. P \text{ iff } & \text{in } \mu \in \mathcal{I}(\star) \wedge \mathcal{I} \models_{\Gamma}^* P \wedge \\
 & \forall \mu^a, \mu^p : \text{in } \mu \in \mathcal{I}(\mu^a) \wedge \mu^a \in \mathcal{I}(\mu^p) \wedge \mu \in \mathcal{I}(\mu^p) \wedge \\
 & \quad \bar{\text{in}}_{\mu^a} \mu \in \mathcal{I}(\mu) \quad \mu \text{ provides the access right to } \mu^a \\
 & \Rightarrow \mu^a \in \mathcal{I}(\mu) \\
 & \text{where } \mu = \Gamma(n)
 \end{aligned}$$

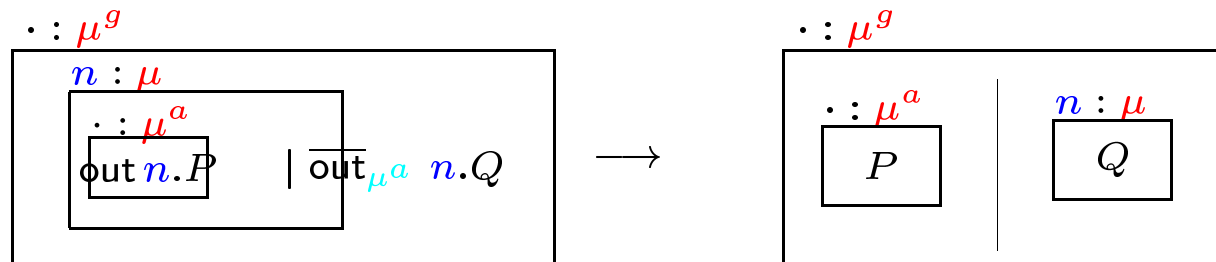
Mimicking the semantics:



Analysis of out-capability

$$\begin{aligned}
 \mathcal{I} \models_{\Gamma}^* \text{out } n. P \text{ iff } & \text{out } \mu \in \mathcal{I}(\star) \wedge \mathcal{I} \models_{\Gamma}^* P \wedge \\
 & \forall \mu^a, \mu^g : \text{out } \mu \in \mathcal{I}(\mu^a) \wedge \mu^a \in \mathcal{I}(\mu) \wedge \mu \in \mathcal{I}(\mu^g) \wedge \\
 & \quad \overline{\text{out}}_{\mu^a} \mu \in \mathcal{I}(\mu) \quad \mu \text{ provides the access right to } \mu^a \\
 & \Rightarrow \mu^a \in \mathcal{I}(\mu^g) \\
 & \text{where } \mu = \Gamma(n)
 \end{aligned}$$

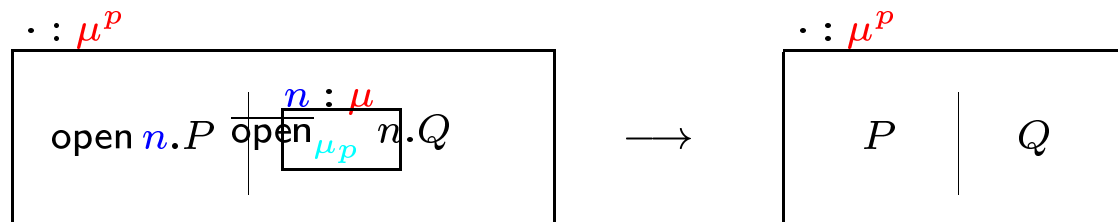
Mimicking the semantics:



Analysis of open-capability

$$\begin{aligned}
 \mathcal{I} \models_{\Gamma}^* \text{open } n. P \text{ iff } & \text{open } \mu \in \mathcal{I}(l) \wedge \mathcal{I} \models_{\Gamma}^* P \wedge \\
 & \forall \mu^p : \text{open } \mu \in \mathcal{I}(\mu^p) \wedge \mu \in \mathcal{I}(\mu^p) \wedge \\
 & \quad \overline{\text{open}}_{\mu^p} \mu \in \mathcal{I}(\mu) \quad \mu \text{ provides the access right to } \mu^p \\
 & \Rightarrow \mathcal{I}(\mu) \subseteq \mathcal{I}(\mu^p) \\
 & \text{where } \mu = \Gamma(n)
 \end{aligned}$$

Mimicking the semantics:



Analysis of co-capabilities

$$\mathcal{I} \models_{\Gamma}^{\star} \overline{\text{in}}_{\mu^a} n. P \text{ iff } \overline{\text{in}}_{\mu^a} \mu \in \mathcal{I}(\star) \wedge \mathcal{I} \models_{\Gamma}^{\star} P$$

where $\mu = \Gamma(n)$

$$\mathcal{I} \models_{\Gamma}^{\star} \overline{\text{out}}_{\mu^a} n. P \text{ iff } \overline{\text{out}}_{\mu^a} \mu \in \mathcal{I}(\star) \wedge \mathcal{I} \models_{\Gamma}^{\star} P$$

where $\mu = \Gamma(n)$

$$\mathcal{I} \models_{\Gamma}^{\star} \overline{\text{open}}_{\mu^P} n. P \text{ iff } \overline{\text{open}}_{\mu^P} \mu \in \mathcal{I}(\star) \wedge \mathcal{I} \models_{\Gamma}^{\star} P$$

where $\mu = \Gamma(n)$

Properties of the analysis

As for the simple analysis we can prove

- **Semantic correctness** (subject reduction result)
the analysis estimate is preserved during the execution:

$$\text{if } \mathcal{I} \models_{\Gamma}^* P \text{ and } P \rightarrow^* Q \text{ then } \mathcal{I} \models_{\Gamma}^* Q$$

- **Moore family property**
all processes can be analysed and has a least (best) analysis result:
the set $\{\mathcal{I} \mid \mathcal{I} \models_{\Gamma}^* P\}$ is a Moore family
- **Efficient implementation**
the cubic time implementation techniques developed for the simple analysis can be carried over to the analysis of discretionary ambients

FLOW LOGIC

Part 5:

Mandatory Access Control
in Mobile Ambients

Flemming Nielson & Hanne Riis Nielson

Mandatory Access Control

- Confidentiality
 - the Bell-LaPadula model
- Integrity
 - the Biba model

Confidentiality

The Bell-LaPadula security model is expressed using

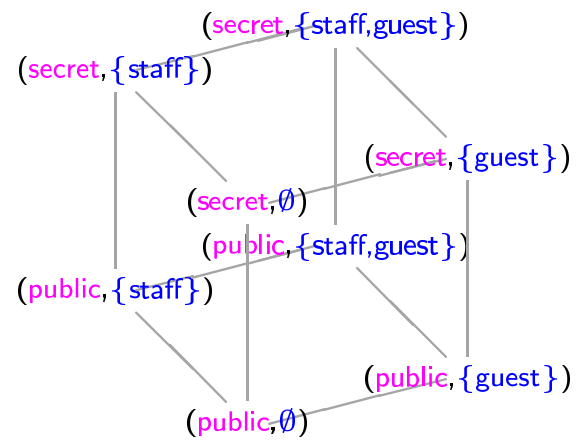
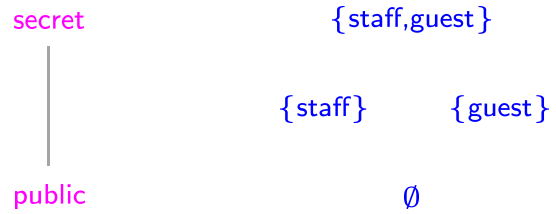
- an access control matrix
- an assignment of security levels to objects and subjects

The security levels are arranged in a lattice (L, \leq) :

$l_1 \leq l_2$ means that l_1 has a lower security level than l_2 .

The overall aim is to prevent information from flowing downwards from a high security level to a low security level — information may only flow upwards.

Example security lattices



Bell-LaPadula for mobile ambients

Assignment of **security levels** to ambient groups:

$$\mathcal{L} : \mathbf{Group} \rightarrow \{\mathbf{public}, \mathbf{secret}\}$$

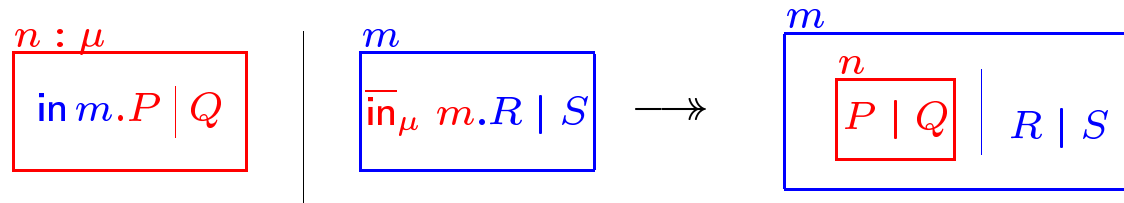
Recall: The overall aim is to prevent information from flowing downwards from a high security level to a low security level

Interpretation for mobile ambients:

- a secret ambient can enter any ambient
- an ambient can only leave a secret ambient when it is in a secret context
- a secret ambient can only be dissolved when it is in a secret context

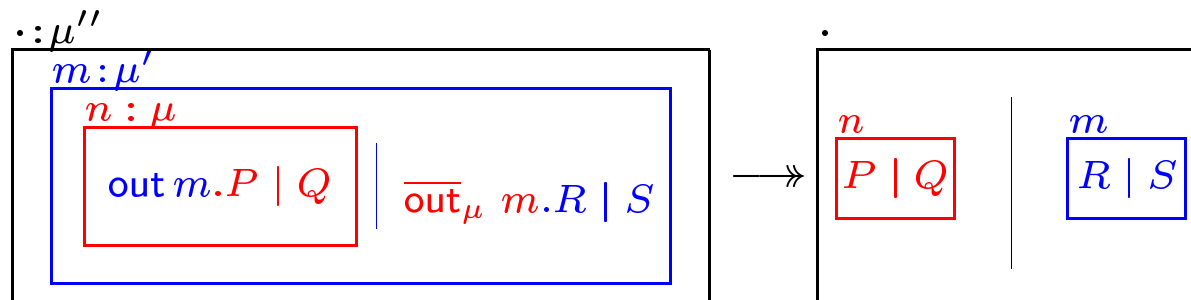
The overall system is assumed to be public.

The reference monitor (from \rightarrow to \twoheadrightarrow)



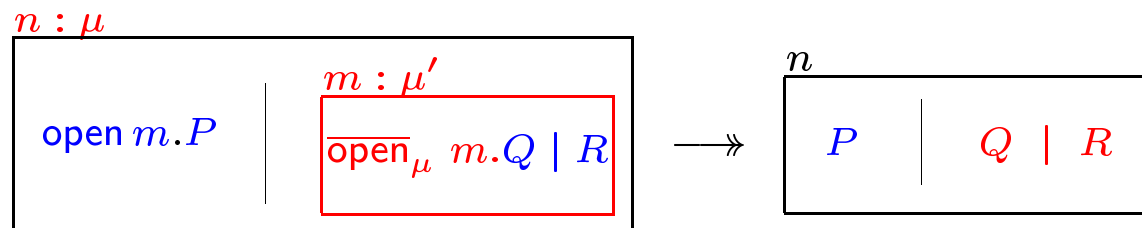
a secret ambient can enter any ambient:

$$\mathcal{L}(\mu) = \text{secret} \Rightarrow \text{true}$$



a secret ambient can leave an ambient in a secret context:

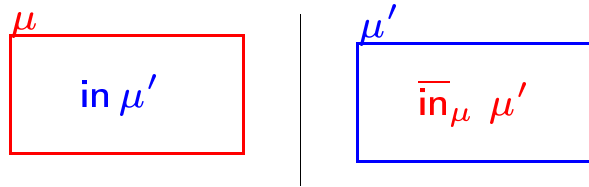
$$\mathcal{L}(\mu) = \text{secret} \Rightarrow \mathcal{L}(\mu'') = \text{secret}$$



a secret ambient can be dissolved in a secret context:

$$\mathcal{L}(\mu') = \text{secret} \Rightarrow \mathcal{L}(\mu) = \text{secret}$$

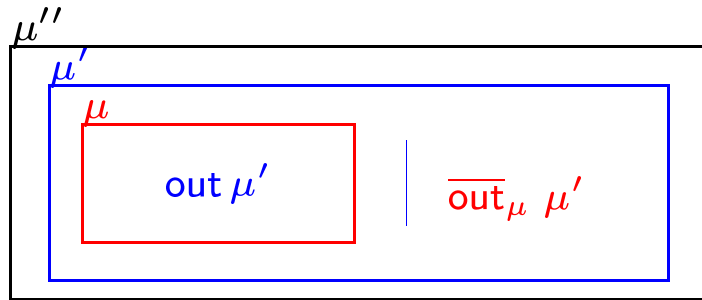
Bell-LaPadula checks on analysis results



a secret ambient can enter any ambient:

Ref. monitor: $\mathcal{L}(\mu) = \text{secret} \Rightarrow \text{true}$

Analysis check: **true**



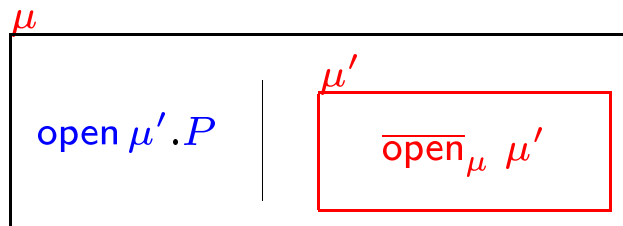
a secret ambient can leave an ambient in a secret context:

Ref. monitor: $\mathcal{L}(\mu) = \text{secret} \Rightarrow \mathcal{L}(\mu'') = \text{secret}$

Analysis check: $\text{out } \mu' \in \mathcal{I}(\mu) \wedge \mu \in \mathcal{I}(\mu') \wedge$

$\overline{\text{out}}_{\mu} \mu' \in \mathcal{I}(\mu') \wedge \mu' \in \mathcal{I}(\mu'') \Rightarrow \mathcal{L}(\mu) \leq \mathcal{L}(\mu'')$

Recall that $\text{public} \leq \text{secret}$



a secret ambient can be dissolved in a secret context:

Ref. monitor: $\mathcal{L}(\mu') = \text{secret} \Rightarrow \mathcal{L}(\mu) = \text{secret}$

Analysis check: $\text{open } \mu' \in \mathcal{I}(\mu) \wedge \overline{\text{open}}_{\mu} \mu' \in \mathcal{I}(\mu') \wedge$

$\mu' \in \mathcal{I}(\mu) \Rightarrow \mathcal{L}(\mu') \leq \mathcal{L}(\mu)$

Correctness result

Theorem: Assume that $\mathcal{I} \models_{\Gamma}^* P$ and that \mathcal{I} satisfies the Bell-LaPadula checks.
Then

$$\Gamma \vdash P \rightarrow^* Q \quad \text{implies} \quad \Gamma \vdash P \twoheadrightarrow^* Q$$

Thus the Bell-LaPadula reference monitor can be dispensed with when the analysis result satisfies the Bell-LaPadula checks.

Example: a packet on a network

$$\mathcal{I} \models_{\Gamma}^{\star} A [p [\text{out } A. \text{in } B. \overline{\text{open}}_S p] \mid \overline{\text{out}}_P A] \mid B [\overline{\text{in}}_P B. \text{open } p]$$

Assume that the sites are secret, the packets are public and the overall system is public.

Γ		\mathcal{I}		\mathcal{L}	
A	S	\star	{S, P}	\star	public
B	S	S	{P, open P, $\overline{\text{in}}_P S$, $\overline{\text{out}}_P S$, in S, out S, $\overline{\text{open}}_S P$ }	S	secret
p	P	P	{in S, out S, $\overline{\text{open}}_S P$ }	P	public

The Bell-LaPadula reference monitor can be dispensed with because:

- $\text{out } S \in \mathcal{I}(P)$, $P \in \mathcal{I}(S)$, $\overline{\text{out}}_P S \in \mathcal{I}(P)$ and $S \in \mathcal{I}(\star)$ have $\mathcal{L}(P) \leq \mathcal{L}(\star)$
- $\text{open } P \in \mathcal{I}(S)$, $\overline{\text{open}}_S P \in \mathcal{I}(P)$ and $P \in \mathcal{I}(S)$ have $\mathcal{L}(P) \leq \mathcal{L}(S)$

Recall that $\text{public} \leq \text{secret}$.

Integrity

The **Biba model** is expressed using

- an access control matrix
- an assignment of **integrity levels** to objects and subjects

The integrity levels are arranged in a lattice (L, \leq) :

$l_1 \leq l_2$ means that l_1 has a lower integrity level than l_2 .

The overall aim is to prevent the corruption of ‘trusted’ high level entities by ‘dubious’ low level entities – information may only flow downwards.

Example lattice: $(\{\text{dubious}, \text{trusted}\}, \leq)$ where **dubious** \leq **trusted**

Biba for mobile ambients

Assignment of **integrity levels** to ambient groups:

$$\mathcal{L} : \mathbf{Group} \rightarrow \{\mathbf{dubious}, \mathbf{trusted}\}$$

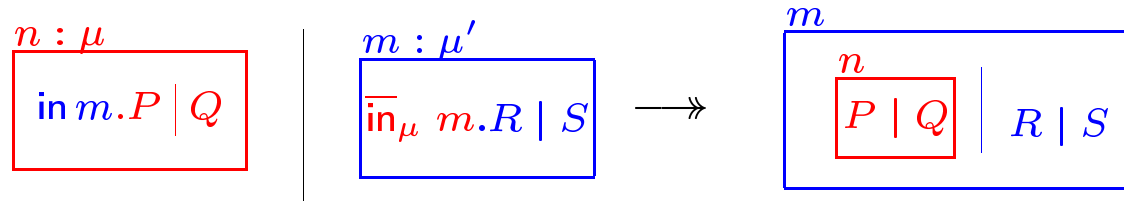
Recall: The overall aim is to prevent the corruption of ‘trusted’ high level entities by ‘dubious’ low level entities

Interpretation for mobile ambients:

- only trusted ambients can enter a trusted ambient
- any ambient can leave a trusted ambient
- only ambients with trusted subambients can be dissolved inside a trusted ambient

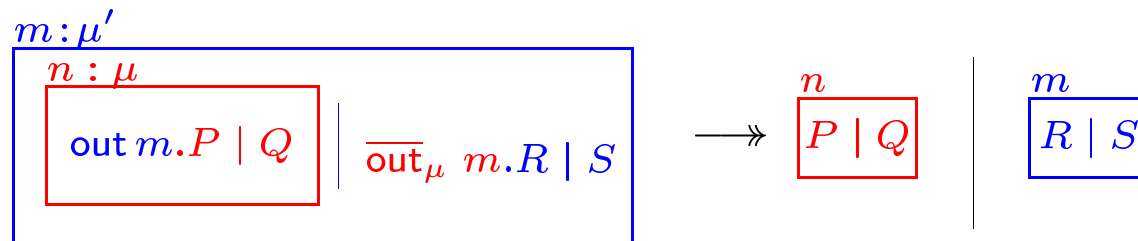
The overall system is assumed to be dubious.

The reference monitor (from \rightarrow to \twoheadrightarrow)



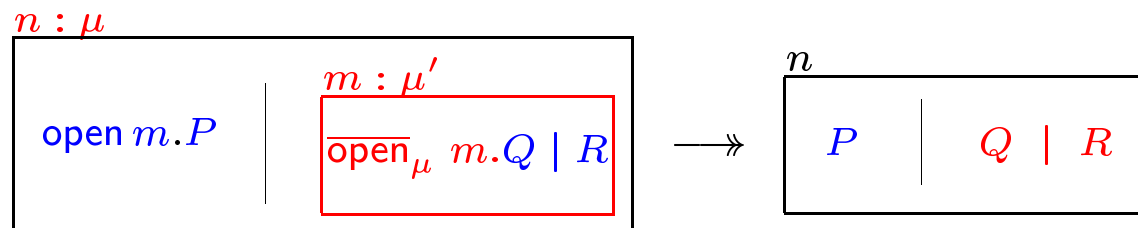
only trusted ambients can enter a trusted ambient:

$$\mathcal{L}(\mu') = \text{trusted} \Rightarrow \mathcal{L}(\mu) = \text{trusted}$$



any ambient can leave a trusted ambient:

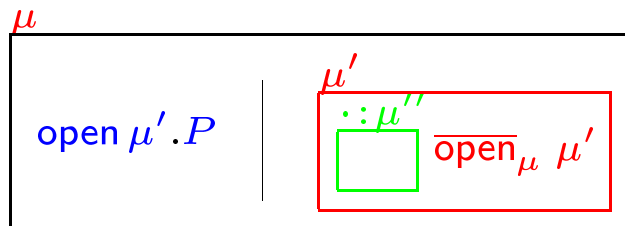
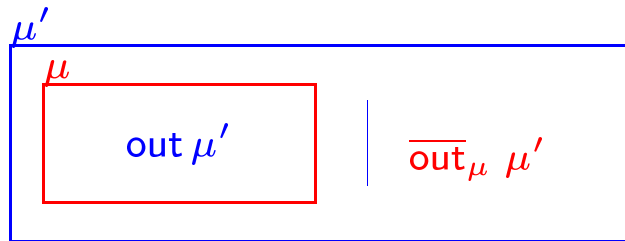
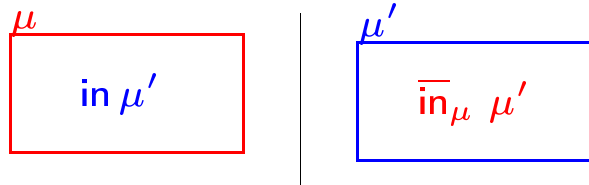
$$\mathcal{L}(\mu') = \text{trusted} \Rightarrow \text{true}$$



only ambients with trusted sub-ambients can be dissolved inside a trusted ambient:

$$\mathcal{L}(\mu) = \text{trusted} \Rightarrow \forall p[\cdot] \in \text{toplevel}(Q \mid R) : \mathcal{L}(\Gamma(p)) = \text{trusted}$$

Biba checks on analysis results



only trusted ambients can enter a trusted ambient:

Ref. monitor: $\mathcal{L}(\mu') = \text{trusted} \Rightarrow \mathcal{L}(\mu) = \text{trusted}$

Analysis check: $\text{in } \mu' \in \mathcal{I}(\mu) \wedge \overline{\text{in}}_{\mu} \mu' \in \mathcal{I}(\mu') \Rightarrow \mathcal{L}(\mu') \leq \mathcal{L}(\mu)$

Recall that dubious \leq trusted

any ambient can leave a trusted ambient:

Ref. monitor: $\mathcal{L}(\mu') = \text{trusted} \Rightarrow \text{true}$

Analysis check: true

only ambients with trusted subambients can be dissolved inside a trusted ambient:

Ref. monitor: $\mathcal{L}(\mu) = \text{trusted} \Rightarrow$

$\forall p[\cdot] \in \text{toplevel}(Q \mid R) : \mathcal{L}(\Gamma(p)) = \text{trusted}$

Analysis check: $\text{open } \mu' \in \mathcal{I}(\mu) \wedge \overline{\text{open}}_{\mu} \mu' \in \mathcal{I}(\mu') \wedge \mu' \in \mathcal{I}(\mu) \Rightarrow \forall \mu'' \in \mathcal{I}(\mu') : \mathcal{L}(\mu) \leq \mathcal{L}(\mu'')$

Correctness result

Theorem: Assume that $\mathcal{I} \models_{\Gamma}^* P$ and that \mathcal{I} satisfies the Biba checks. Then

$$\Gamma \vdash P \rightarrow^* Q \quad \text{implies} \quad \Gamma \vdash P \twoheadrightarrow^* Q$$

Thus the Biba reference monitor can be dispensed with when the analysis result satisfies the Biba checks.

Example: a packet on a network

$$\mathcal{I} \models_{\Gamma}^* A [p [\text{out } A. \text{in } B. \overline{\text{open}}_S p] \mid \overline{\text{out}}_P A] \mid B [\overline{\text{in}}_P B. \text{open } p]$$

Assume that the sites are dubious, the packets are trusted and the overall system is dubious.

Γ	
A	S
B	S
p	P

\mathcal{I}	
*	{S, P}
S	{P, open P, $\overline{\text{in}}_P S$, $\overline{\text{out}}_P S$, in S, out S, $\overline{\text{open}}_S P$ }
P	{in S, out S, $\overline{\text{open}}_S P$ }

\mathcal{L}	
*	dubious
S	dubious
P	trusted

The Biba reference monitor can be dispensed with because:

- $\text{in } S \in \mathcal{I}(P)$, $\overline{\text{in}}_P S \in \mathcal{I}(S)$ have $\mathcal{L}(S) \leq \mathcal{L}(P)$
- $\text{open } P \in \mathcal{I}(S)$, $\overline{\text{open}}_S P \in \mathcal{I}(P)$ and $P \in \mathcal{I}(S)$ have $\forall \mu \in \mathcal{I}(P) : \mathcal{L}(P) \leq \mathcal{L}(\mu)$

Recall that dubious \leq trusted.

Example: a packet on a network

$$\mathcal{I} \models_{\Gamma}^* A [p [\text{out } A. \text{in } B. \overline{\text{open}}_S p] \mid \overline{\text{out}}_P A] \mid B [\overline{\text{in}}_P B. \text{open } p]$$

Assume that the **sites are trusted**, the **packets are dubious** and the overall system is dubious.

Γ		\mathcal{I}		\mathcal{L}	
A	S	*	{S, P}	*	dubious
B	S	S	{P, open P, $\overline{\text{in}}_P S$, $\overline{\text{out}}_P S$, in S, out S, $\overline{\text{open}}_S P$ }	S	trusted
p	P	P	{in S, out S, $\overline{\text{open}}_S P$ }	P	dubious

The Biba reference monitor **cannot** be dispensed with because:

- $\text{in } S \in \mathcal{I}(P)$, $\overline{\text{in}}_P S \in \mathcal{I}(S)$ do **not** have $\mathcal{L}(S) \leq \mathcal{L}(P)$

The trusted sites may be corrupted by the dubious packet.

Recall that dubious \leq trusted.

FLOW LOGIC

Part 6:

A Multi-Paradigmatic Approach
to Static Analysis

Flemming Nielson & Hanne Riis Nielson

A Tutorial based on the λ -calculus

The λ -calculus: $e ::= c \mid x \mid \lambda x_0.e_0 \mid e_1 e_2$

Environment-based call-by-value big-step operational semantics:

$$\rho \vdash c \rightarrow c$$

$$\rho \vdash x \rightarrow \rho(x)$$

$$\rho \vdash \lambda x_0.e_0 \rightarrow \langle \lambda x_0.e_0, \rho \rangle$$

$$\frac{\rho \vdash e_1 \rightarrow \langle \lambda x_0.e_0, \rho_0 \rangle \quad \rho \vdash e_2 \rightarrow v_2 \quad \rho_0[x_0 \mapsto v_2] \vdash e_0 \rightarrow v_0}{\rho \vdash e_1 e_2 \rightarrow v_0}$$

Example:

$$\frac{[] \vdash \lambda x.x \mathfrak{Z} \rightarrow C_x \quad [] \vdash \lambda y.\lambda z.y \rightarrow C_y \quad \frac{\rho_x \vdash x \rightarrow C_y \quad \rho_x \vdash \mathfrak{Z} \rightarrow \mathfrak{Z} \quad \rho_y \vdash \lambda z.y \rightarrow C_z}{\rho_x \vdash x \mathfrak{Z} \rightarrow C_z}}{[] \vdash (\lambda x.x \mathfrak{Z})(\lambda y.\lambda z.y) \rightarrow C_z}$$

Abbreviations:

$$\begin{array}{ll} C_x : \langle \lambda x.x \mathfrak{Z}, [] \rangle & \rho_x : [x \mapsto C_y] \\ C_y : \langle \lambda y.\lambda z.y, [] \rangle & \rho_y : [y \mapsto \mathfrak{Z}] \\ C_z : \langle \lambda z.y, \rho_y \rangle & \end{array}$$

Example analysis

- Aim: to predict which values an expression may evaluate to
- Abstract representations of the semantic values:
 - a **constant** c is represented by \diamond
(so the analysis will record the presence of a constant but not its value)
 - a **closure** $\langle \lambda x_0. e_0, \rho_0 \rangle$ is represented by an abstract closure $\{\lambda x_0. e_0\}$
(so the analysis forgets about the environment)
 - an **environment** ρ is represented by a single global abstract environment $\hat{\rho} \in \widehat{\mathbf{Env}} = \mathbf{Var} \rightarrow \widehat{\mathbf{Val}}$ (where $\widehat{\mathbf{Val}}$ is the set of abstract values)
(so the analysis does not distinguish between different environments)

Abstract succinct specification

$\hat{\rho} \models_{\text{as}} e : \hat{v}$ The set \hat{v} of abstract values is an acceptable analysis estimate for the expression e in the context specified by the abstract environment $\hat{\rho}$.

$$\hat{\rho} \models_{\text{as}} c : \hat{v} \quad \underline{\text{iff}} \quad \diamond \in \hat{v}$$

$$\hat{\rho} \models_{\text{as}} x : \hat{v} \quad \underline{\text{iff}} \quad \hat{\rho}(x) \subseteq \hat{v}$$

$$\hat{\rho} \models_{\text{as}} \lambda x_0. e_0 : \hat{v} \quad \underline{\text{iff}} \quad \{\lambda x_0. e_0\} \in \hat{v}$$

$$\hat{\rho} \models_{\text{as}} e_1 e_2 : \hat{v} \quad \underline{\text{iff}} \quad \hat{\rho} \models_{\text{as}} e_1 : \hat{v}_1 \wedge \hat{\rho} \models_{\text{as}} e_2 : \hat{v}_2 \wedge$$

$$\forall \{\lambda x_0. e_0\} \in \hat{v}_1 : \hat{v}_2 \neq \emptyset \Rightarrow$$

$$[\hat{v}_2 \subseteq \hat{\rho}(x_0) \wedge \hat{\rho} \models_{\text{as}} e_0 : \hat{v}_0 \wedge \hat{v}_0 \subseteq \hat{v}]$$

Example:

$$\hat{\rho} \models_{\text{as}} (\lambda x.x\ 3)(\lambda y.\lambda z.y) : \{\{\lambda z.y\}\}$$

is an acceptable analysis estimate in the context given by:

$$\hat{\rho} = \begin{array}{|c|c|c|} \hline & x & y & z \\ \hline & \{\{\lambda y.\lambda z.y\}\} & \{\diamond\} & \emptyset \\ \hline \end{array}$$

Notes

- The specification is *abstract*; this follows the tradition of data flow analysis and abstract interpretation and is in contrast to the more type theoretic approaches.
- In the terminology of constraint-based analysis the analysis is *polyvariant*.
- The specification is *succinct* because the occurrence of \hat{v} in $\hat{\rho} \models_{as} e : \hat{v}$ expresses the overall analysis estimate for e ; if we want details about the analysis information for subexpressions of e then we have to inspect the reasoning leading to the judgement.
- The specification applies to *open systems* as well as closed systems since λ -abstractions are analysed when they are applied; hence they are not required to be part of the program of interest but may for example be part of library routines.

Towards a compositional specification

- a compositional specification will analyse the body of λ -abstractions at their definition point rather than at their application point
- we need a way of linking information available at these points
- we introduce a **global cache** \widehat{C} for recording analysis estimates for the bodies of the λ -abstractions.

Extend the syntax with labels $\ell_0 \in \mathbf{Lab}$

$$e ::= c \mid x \mid \lambda x_0. e_0^{\ell_0} \mid e_1 e_2$$

and take $\widehat{C} \in \widehat{\mathbf{Cache}} = \mathbf{Lab} \rightarrow \widehat{\mathbf{Val}}$

Compositional succinct specification

$(\widehat{C}, \widehat{\rho}) \models_{cs} e : \widehat{v}$ The set \widehat{v} is an acceptable analysis estimate for e in the context specified by \widehat{C} and $\widehat{\rho}$.

$$(\widehat{C}, \widehat{\rho}) \models_{cs} c : \widehat{v} \quad \underline{\text{iff}} \quad \diamond \in \widehat{v}$$

$$(\widehat{C}, \widehat{\rho}) \models_{cs} x : \widehat{v} \quad \underline{\text{iff}} \quad \widehat{\rho}(x) \subseteq \widehat{v}$$

$$\begin{aligned} (\widehat{C}, \widehat{\rho}) \models_{cs} \lambda x_0. e_0^{\ell_0} : \widehat{v} \quad \underline{\text{iff}} \quad & \{\lambda x_0. e_0^{\ell_0}\} \in \widehat{v} \wedge \\ & \widehat{\rho}(x_0) \neq \emptyset \Rightarrow [(\widehat{C}, \widehat{\rho}) \models_{cs} e_0^{\ell_0} : \widehat{v}_0 \wedge \widehat{v}_0 \subseteq \widehat{C}(\ell_0)] \end{aligned}$$

$$\begin{aligned} (\widehat{C}, \widehat{\rho}) \models_{cs} e_1 e_2 : \widehat{v} \quad \underline{\text{iff}} \quad & (\widehat{C}, \widehat{\rho}) \models_{cs} e_1 : \widehat{v}_1 \wedge (\widehat{C}, \widehat{\rho}) \models_{cs} e_2 : \widehat{v}_2 \wedge \\ & \forall \{\lambda x_0. e_0^{\ell_0}\} \in \widehat{v}_1 : \widehat{v}_2 \subseteq \widehat{\rho}(x_0) \wedge \widehat{C}(\ell_0) \subseteq \widehat{v} \end{aligned}$$

Example:

$$(\hat{C}, \hat{\rho}) \models_{cs} (\lambda x.(x\ 3)^1)(\lambda y.(\lambda z.y^3)^2) : \{\{\lambda z.y^3\}\}$$

is an acceptable analysis estimate in the context where:

$$\hat{C} =$$

1	2	3
$\{\{\lambda z.y^3\}\}$	$\{\{\lambda z.y^3\}\}$	\emptyset

$$\hat{\rho} =$$

x	y	z
$\{\{\lambda y.(\lambda z.y^3)^2\}\}$	$\{\diamond\}$	\emptyset

Notes

- The specification is *compositional*; it still has *succinct* components since the \hat{v} of $(\hat{C}, \hat{\rho}) \models_{cs} e : \hat{v}$ gives the analysis information for the overall expression e ; to get hold of the analysis information of subexpressions we have to inspect the reasoning leading to the judgement $(\hat{C}, \hat{\rho}) \models_{cs} e : \hat{v}$.
- In contrast to the abstract specification, we only have to find one acceptable analysis estimate for each subexpression; in the terminology of constraint-based analysis the analysis is said to be *monovariant*
- The specification is restricted to *closed* systems since the bodies of λ -abstractions only are analysed at their definition points.

Towards a verbose specification

- a more implementation oriented specification will record the analysis estimates of *all* program points — there will be no need to inspect the reasoning leading to the analysis judgements
- the cache can be extended to contain this information for all program points

Extend the syntax with labels for all subexpressions:

$$e^l ::= c^l \mid x^l \mid (\lambda x_0. e_0^{l_0})^l \mid (e_1^{l_1} e_2^{l_2})^l$$

Compositional verbose specification

$(\widehat{C}, \widehat{\rho}) \models_{cv} e^\ell$ with the idea that $\widehat{C}(\ell)$ is the analysis estimate for e

$$(\widehat{C}, \widehat{\rho}) \models_{cv} c^\ell \quad \underline{\text{iff}} \quad \diamond \in \widehat{C}(\ell)$$

$$(\widehat{C}, \widehat{\rho}) \models_{cv} x^\ell \quad \underline{\text{iff}} \quad \widehat{\rho}(x) \subseteq \widehat{C}(\ell)$$

$$\begin{aligned} (\widehat{C}, \widehat{\rho}) \models_{cv} (\lambda x_0. e_0^{\ell_0})^\ell \quad \underline{\text{iff}} \quad & \{\lambda x_0. e_0^{\ell_0}\} \in \widehat{C}(\ell) \wedge \\ & \widehat{\rho}(x_0) \neq \emptyset \Rightarrow (\widehat{C}, \widehat{\rho}) \models_{cv} e_0^{\ell_0} \end{aligned}$$

$$\begin{aligned} (\widehat{C}, \widehat{\rho}) \models_{cv} (e_1^{\ell_1} e_2^{\ell_2})^\ell \quad \underline{\text{iff}} \quad & (\widehat{C}, \widehat{\rho}) \models_{cv} e_1^{\ell_1} \wedge (\widehat{C}, \widehat{\rho}) \models_{cv} e_2^{\ell_2} \wedge \\ & \forall \{\lambda x_0. e_0^{\ell_0}\} \in \widehat{C}(\ell_1) : \widehat{C}(\ell_2) \subseteq \widehat{\rho}(x_0) \wedge \widehat{C}(\ell_0) \subseteq \widehat{C}(\ell) \end{aligned}$$

Example:

$$(\widehat{C}, \widehat{\rho}) \models_{cv} (\lambda x.(x^4 3^5)^1)^6 (\lambda y.(\lambda z.y^3)^2)^7)^8$$

where

$$\widehat{C} =$$

1	2	3	4
$\{\{\lambda z.y^3\}\}$	$\{\{\lambda z.y^3\}\}$	\emptyset	$\{\{\lambda y.(\lambda z.y^3)^2\}\}$
5	6	7	8
$\{\diamond\}$	$\{\{\lambda x.(x^4 3^5)^1\}\}$	$\{\{\lambda y.(\lambda z.y^3)^2\}\}$	$\{\{\lambda z.y^3\}\}$

$$\widehat{\rho} =$$

x	y	z
$\{\{\lambda y.(\lambda z.y^3)^2\}\}$	$\{\diamond\}$	\emptyset

Abstract verbose specification

$(\widehat{C}, \widehat{\rho}) \models_{\text{av}} e^\ell$ extends the compositional verbose specification to open systems

$$(\widehat{C}, \widehat{\rho}) \models_{\text{av}} c^\ell \quad \underline{\text{iff}} \quad \diamond \in \widehat{C}(\ell)$$

$$(\widehat{C}, \widehat{\rho}) \models_{\text{av}} x^\ell \quad \underline{\text{iff}} \quad \widehat{\rho}(x) \subseteq \widehat{C}(\ell)$$

$$(\widehat{C}, \widehat{\rho}) \models_{\text{av}} (\lambda x_0. e_0^{\ell_0})^\ell \quad \underline{\text{iff}} \quad \{\lambda x_0. e_0^{\ell_0}\} \in \widehat{C}(\ell)$$

$$(\widehat{C}, \widehat{\rho}) \models_{\text{av}} (e_1^{\ell_1} e_2^{\ell_2})^\ell \quad \underline{\text{iff}} \quad (\widehat{C}, \widehat{\rho}) \models_{\text{av}} e_1^{\ell_1} \wedge (\widehat{C}, \widehat{\rho}) \models_{\text{av}} e_2^{\ell_2} \wedge$$

$$\forall \{\lambda x_0. e_0^{\ell_0}\} \in \widehat{C}(\ell_1) : \widehat{C}(\ell_2) \neq \emptyset \Rightarrow$$

$$[\widehat{C}(\ell_2) \subseteq \widehat{\rho}(x_0) \wedge (\widehat{C}, \widehat{\rho}) \models_{\text{av}} e_0^{\ell_0} \wedge \widehat{C}(\ell_0) \subseteq \widehat{C}(\ell)]$$

Example:

$$(\hat{C}, \hat{\rho}) \models_{\text{av}} (\lambda x.(x^4 3^5)^1)^6 (\lambda y.(\lambda z.y^3)^2)^7)^8$$

where \hat{C} and $\hat{\rho}$ are as in the previous example.

Relationship between the specifications

$$\begin{array}{ccc} (\hat{C}, \hat{\rho}) \models_{\text{cs}} e : \hat{C}(\ell) & \Leftrightarrow & (\hat{C}, \hat{\rho}) \models_{\text{cv}} e^\ell \\ \Downarrow & & \Downarrow \\ \hat{\rho} \models_{\text{as}} e : \hat{C}(\ell) & \Leftrightarrow & (\hat{C}, \hat{\rho}) \models_{\text{av}} e^\ell \end{array}$$

Flow Logic Specifications

- Describe **the universe of discourse for the analysis estimates**
 - usually given by complete lattices and hence follows the approaches of data flow analysis, constraint-based analysis and abstract interpretation.
- Describe **the format of the judgements and their defining clauses**
 - focuses on *what* the analysis does and not *how* it does it
 - * it is not necessary to think about the design and the implementation of the analysis at the same time
 - * one can concentrate on specifying the analysis, i.e. on how to collect analysis information and link it together
 - * the problem of trading efficiency for precision can be studied at the specification level and hence independently of implementation details

Desirable properties of a flow logic specification

- the judgements are well-defined;
- the judgements are semantically correct;
- the judgements have a Moore family (or model intersection) property
- the judgements have efficient implementations.

Well-definedness of the analysis

Compositional specifications: It is straightforward to see that they are well-defined; a simple induction on the syntax of the programs will do.

Abstract specifications: It is not so obvious; for the abstract succinct analysis it is the following clause that is problematic:

$$\begin{aligned} \hat{\rho} \models_{\text{as}} e_1 e_2 : \hat{v} \quad \underline{\text{iff}} \quad & \hat{\rho} \models_{\text{as}} e_1 : \hat{v}_1 \wedge \hat{\rho} \models_{\text{as}} e_2 : \hat{v}_2 \wedge \\ & \forall \{\lambda x_0. e_0\} \in \hat{v}_1 : \hat{v}_2 \neq \emptyset \Rightarrow \\ & [\hat{v}_2 \subseteq \hat{\rho}(x_0) \wedge \hat{\rho} \models_{\text{as}} e_0 : \hat{v}_0 \wedge \hat{v}_0 \subseteq \hat{v}] \end{aligned}$$

Solution

- turn the defining clauses into an appropriate **functional over some complete lattice** — in doing so we shall impose some demands on the form of the clauses such that monotonicity of the functional is enforced
- **Tarski's fixed point theorem** ensures that the functional has fixed points
- since we are giving meaning to a specification, the analysis has to be defined as the *greatest* fixed point, i.e. **co-inductively**.

Example: The abstract succinct specification

Complete lattice $(\widehat{\mathbf{Env}} \times \mathbf{Exp} \times \widehat{\mathbf{Val}} \rightarrow \{\text{tt}, \text{ff}\}, \sqsubseteq)$:

$$Q_1 \sqsubseteq Q_2 \text{ iff } \forall(\hat{\rho}, e, \hat{v}) : (Q_1(\hat{\rho}, e, \hat{v}) = \text{tt}) \Rightarrow (Q_2(\hat{\rho}, e, \hat{v}) = \text{tt})$$

The clauses for $\hat{\rho} \models_{\text{as}} e : \hat{v}$ define the functional

$$Q_{\text{as}} : (\widehat{\mathbf{Env}} \times \mathbf{Exp} \times \widehat{\mathbf{Val}} \rightarrow \{\text{tt}, \text{ff}\}) \rightarrow (\widehat{\mathbf{Env}} \times \mathbf{Exp} \times \widehat{\mathbf{Val}} \rightarrow \{\text{tt}, \text{ff}\})$$

by

$$Q_{\text{as}}(Q)(\hat{\rho}, c, \hat{v}) = (\diamond \in \hat{v})$$

$$Q_{\text{as}}(Q)(\hat{\rho}, x, \hat{v}) = (\hat{\rho}(x) \subseteq \hat{v})$$

$$Q_{\text{as}}(Q)(\hat{\rho}, \lambda x_0. e_0, \hat{v}) = (\{\lambda x_0. e_0\} \in \hat{v})$$

$$Q_{\text{as}}(Q)(\hat{\rho}, e_1 e_2, \hat{v}) = \exists \hat{v}_1, \hat{v}_2 : Q(\hat{\rho}, e_1, \hat{v}_1) \wedge Q(\hat{\rho}, e_2, \hat{v}_2) \wedge$$

$$\forall \{\lambda x_0. e_0\} \in \hat{v}_1 : \hat{v}_2 \neq \emptyset \Rightarrow$$

$$[\exists \hat{v}_0 : \hat{v}_2 \subseteq \hat{\rho}(x_0) \wedge Q(\hat{\rho}, e_0, \hat{v}_0) \wedge \hat{v}_0 \subseteq \hat{v}]$$

Semantic correctness

Classical distinction between

- **semantic based:** the analysis information can be proved correct with respect to a semantics
- **semantic directed:** the analysis specification is calculated from a semantic specification

Flow logic is a semantics based approach to static analysis

The correctness can be established with respect to many different kinds of semantics (e.g. big-step or small-step operational semantics or denotational semantics) — the actual choice of semantics may significantly influence the style of the proof.

Correctness relation

Specifies the relationship between the entities of the semantics and the analysis estimates.

Example:

$$\begin{aligned}c \mathcal{R}_{\text{Val}} (\hat{v}, \hat{\rho}) & \quad \underline{\text{iff}} \quad \diamond \in \hat{v} \\ \langle \lambda x_0. e_0, \rho_0 \rangle \mathcal{R}_{\text{Val}} (\hat{v}, \hat{\rho}) & \quad \underline{\text{iff}} \quad \{ \lambda x_0. e_0 \} \in \hat{v} \wedge \rho_0 \mathcal{R}_{\text{Env}} \hat{\rho} \\ \rho \mathcal{R}_{\text{Env}} \hat{\rho} & \quad \underline{\text{iff}} \quad \forall x \in \text{dom}(\rho) : \rho(x) \mathcal{R}_{\text{Val}} (\hat{\rho}(x), \hat{\rho})\end{aligned}$$

Subject reduction result

For a big-step operational semantics: $\rho \vdash e \rightarrow v$

Theorem: If $\rho \vdash e \rightarrow v$, $\rho \mathcal{R}_{\text{Env}} \hat{\rho}$ and $\hat{\rho} \models_{\text{as}} e : \hat{v}$ then $v \mathcal{R}_{\text{Val}} (\hat{v}, \hat{\rho})$.

For a small-step operational semantics: $\rho \Vdash e \rightarrow e'$

Theorem: If $\rho \Vdash e \rightarrow e'$, $\rho \mathcal{R}_{\text{Env}} \hat{\rho}$ and $\hat{\rho} \models_{\text{as}} e : \hat{v}$ then $\hat{\rho} \models_{\text{as}} e' : \hat{v}$.

Small-step operational semantics

An environment-based small-step operational semantics for the λ -calculus uses values $v \in \mathbf{Val}$ together with intermediate expressions containing closures $\langle \lambda x_0.e_0, \rho_0 \rangle$ and special constructs of the form $\mathbf{bind} \rho_0 \text{ in } e_0$; the role of the latter is to stack the environments arising in applications. The transitions are written as $\rho \Vdash e \rightarrow e'$ and are defined by:

$$\rho \Vdash x \rightarrow \rho(x)$$

$$\rho \Vdash \lambda x_0.e_0 \rightarrow \langle \lambda x_0.e_0, \rho \rangle$$

$$\rho \Vdash (\langle \lambda x_0.e_0, \rho_0 \rangle) v \rightarrow \mathbf{bind} \rho_0[x_0 \mapsto v] \text{ in } e_0$$

$$\frac{\rho_0 \Vdash e \rightarrow e'}{\rho \Vdash \mathbf{bind} \rho_0 \text{ in } e \rightarrow \mathbf{bind} \rho_0 \text{ in } e'}$$

$$\frac{\rho \Vdash e_1 \rightarrow e'_1}{\rho \Vdash e_1 e_2 \rightarrow e'_1 e_2}$$

$$\frac{\rho \Vdash e_2 \rightarrow e'_2}{\rho \Vdash e_1 e_2 \rightarrow e_1 e'_2}$$

$$\frac{\rho_0 \Vdash e \rightarrow v}{\rho \Vdash \mathbf{bind} \rho_0 \text{ in } e \rightarrow v}$$

Note: the analysis is extended to operate on the intermediate terms:

$$\hat{\rho} \models_{\text{as}} \langle \lambda x_0. e_0, \rho_0 \rangle : \hat{v} \quad \underline{\text{iff}} \quad \langle \lambda x_0. e_0, \rho_0 \rangle \mathcal{R}_{\text{Val}} (\hat{v}, \hat{\rho})$$

$$\hat{\rho} \models_{\text{as}} \text{bind } \rho_0 \text{ in } e_0 : \hat{v} \quad \underline{\text{iff}} \quad \hat{\rho} \models_{\text{as}} e_0 : \hat{v} \wedge \rho_0 \mathcal{R}_{\text{Env}} \hat{\rho}$$

Moore family result

Having specified an analysis it is natural to ask

- whether every expression admits an acceptable analysis estimate and
- whether every expression has a best or most informative analysis estimate.

It is sufficient to prove that the set of acceptable analysis estimates enjoys a Moore family (or model intersection) property: A *Moore family* is a subset \widehat{V} of a complete lattice satisfying that whenever $Y \subseteq \widehat{V}$ then $\bigcap Y \in \widehat{V}$.

For the abstract succinct specification: The complete lattice of interest is $\widehat{\mathbf{Val}} \times \widehat{\mathbf{Env}}$ equipped with the pointwise subset ordering. Then:

Theorem: The set $\{(\widehat{\rho}, \widehat{v}) \mid \widehat{\rho} \models_{\text{as}} e : \widehat{v}\}$ is a Moore family for all e .

Efficient implementation via constraints

Turn the compositional verbose specification into an algorithm for computing a set of conditional constraints that subsequently can be solved using standard constraint solvers.

$$\mathcal{C}_{cv}[[c^\ell]]lst = \{ lst \Rightarrow \{\diamond\} \subseteq \mathbf{C}[\ell] \}$$

$$\mathcal{C}_{cv}[[x^\ell]]lst = \{ lst \Rightarrow \mathbf{R}[x] \subseteq \mathbf{C}[\ell] \}$$

$$\mathcal{C}_{cv}[(\lambda x_0.e_0^{\ell_0})^\ell]lst = \{ lst \Rightarrow \{\{\lambda x_0.e_0^{\ell_0}\}\} \subseteq \mathbf{C}[\ell] \} \cup \mathcal{C}_{cv}[e_0^{\ell_0}](lst \wedge (\mathbf{R}[x_0] \neq \emptyset))$$

$$\mathcal{C}_{cv}[(e_1^{\ell_1} e_2^{\ell_2})^\ell]lst = \mathcal{C}_{cv}[e_1^{\ell_1}]lst \cup \mathcal{C}_{cv}[e_2^{\ell_2}]lst$$

$$\cup \{ lst \wedge (\{\lambda x_0.e_0^{\ell_0}\} \in \mathbf{C}[\ell_1]) \Rightarrow \mathbf{C}[\ell_2] \subseteq \mathbf{R}[x_0] \mid \lambda x_0.e_0^{\ell_0} \text{ is in } e_\star \}$$

$$\cup \{ lst \wedge (\{\lambda x_0.e_0^{\ell_0}\} \in \mathbf{C}[\ell_1]) \Rightarrow \mathbf{C}[\ell_0] \subseteq \mathbf{C}[\ell] \mid \lambda x_0.e_0^{\ell_0} \text{ is in } e_\star \}$$

An alternative is to use logical formulae to be solved appropriate solvers.

Syntactic soundness and completeness

Any solution to the constraint system is also an acceptable analysis estimate according to the specification and vice versa

Lemma: $(\widehat{C}, \widehat{\rho}) \models_{cv} e_\star$ if and only if $(\widehat{\rho}, \widehat{C})$ satisfies the constraints of $\mathcal{C}_{cv}[[e_\star]]\epsilon$.

Hence it follows that a solution $(\widehat{\rho}, \widehat{C})$ to the constraints also will be an acceptable analysis result for the other three specifications.

Pragmatics

Language paradigms

- functional (λ -calculus)

$$e ::= c \mid x \mid \lambda x_0.e_0 \mid e_1 e_2$$

- imperative

$$S ::= x := e \mid S_1; S_2 \mid \dots$$

- object oriented (imperative object calculus)

$$O ::= [m_i = \varsigma(x_i).O_i]_{i=1}^n \mid O.m \mid O.m := \varsigma(x_0).O_0 \mid \dots$$

- concurrent (π -calculus)

$$P ::= \bar{u}t.P \mid u(x).P \mid (\nu n)P \mid P_1 \mid P_2 \mid \dots$$

Imperative constructs

We can model a classical forward analysis using a complete lattice $(\widehat{\mathbf{St}}, \sqsubseteq)$ of abstract states and with transfer functions $\phi_{x:=e} : \widehat{\mathbf{St}} \rightarrow \widehat{\mathbf{St}}$ specifying how the assignments $x := e$ modify the abstract states.

The judgements of the (succinct) analysis have the form

$$\models S : \hat{\sigma} \twoheadrightarrow \hat{\sigma}'$$

and expresses that if $\hat{\sigma}$ describes the initial state then $\hat{\sigma}'$ will describe the possible final states.

$$\models x := e : \hat{\sigma} \twoheadrightarrow \hat{\sigma}' \quad \underline{\text{iff}} \quad \phi_{x:=e}(\hat{\sigma}) \sqsubseteq \hat{\sigma}'$$

$$\models S_1; S_2 : \hat{\sigma} \twoheadrightarrow \hat{\sigma}'' \quad \underline{\text{iff}} \quad \models S_1 : \hat{\sigma} \twoheadrightarrow \hat{\sigma}' \wedge \models S_2 : \hat{\sigma}' \twoheadrightarrow \hat{\sigma}''$$

Example: if $\widehat{\mathbf{St}} = \mathbf{Var} \rightarrow \widehat{\mathbf{Val}}$ then we may take $\phi_{x:=e}(\hat{\sigma}) = \hat{\sigma}[x \mapsto \widehat{\mathcal{E}}[e]\hat{\sigma}]$ where $\widehat{\mathcal{E}}$ defines the analysis of expressions.

An alternative:

A coarser analysis that does not distinguish between the program points.

Let a single abstract state $\hat{\sigma} \in \widehat{\mathbf{St}} = \mathbf{Var} \rightarrow \widehat{\mathbf{Val}}$ capture the information about *all* the states that may occur during execution.

The judgements of this (verbose) analysis have the form

$$\hat{\sigma} \models' S$$

and it is given by:

$$\hat{\sigma} \models' x := e \quad \underline{\text{iff}} \quad \widehat{\mathcal{E}}[e]\hat{\sigma} \sqsubseteq \hat{\sigma}(x)$$

$$\hat{\sigma} \models' S_1; S_2 \quad \underline{\text{iff}} \quad \hat{\sigma} \models' S_1 \wedge \hat{\sigma} \models' S_2$$

This may not seem very useful — but corresponds to what we have to do for concurrency.

Object-oriented constructs

The aim is to determine the set of objects that can reach various points in the program.

An object $[m_i = \varsigma(x_i).O_i]_{i=1}^n$ is represented by a tuple $\vec{m} = (m_1, \dots, m_n)$ listing the names of its methods; a method will be represented by an abstract closure $\{\varsigma(x_0).O_0\} \in \mathbf{Mt}$.

The judgements of the analysis have the form

$$(\hat{\rho}, \hat{\sigma}) \models O : \hat{v}$$

where $\hat{v} \in \widehat{\mathbf{Val}} = \mathcal{P}(\mathbf{Nam}^*)$ describes the set of abstract objects that O may evaluate to; $\hat{\rho} : \mathbf{Var} \rightarrow \widehat{\mathbf{Val}}$ describes the abstract values associated with the variables and $\hat{\sigma} \in \widehat{\mathbf{St}} = \mathbf{Nam}^* \times \mathbf{Nam} \rightarrow \mathcal{P}(\mathbf{Mt})$ represents *all* the states that may arise during the computation by a single abstract state.

Object definition:

$$\begin{aligned}
 (\hat{\rho}, \hat{\sigma}) \models [m_i =_{\varsigma} (x_i).O_i]_{i=1}^n : \hat{v} \quad \underline{\text{iff}} \quad & (m_1, \dots, m_n) \in \hat{v} \wedge \\
 & \forall i \in \{1, \dots, n\} : \\
 & \quad \{ \varsigma(x_i).O_i \} \in \hat{\sigma}((m_1, \dots, m_n), m_i)
 \end{aligned}$$

Note the similarity with the clause for function definition.

Method call:

$$\begin{aligned}
 (\hat{\rho}, \hat{\sigma}) \models O.m : \hat{v} \quad \underline{\text{iff}} \quad & (\hat{\rho}, \hat{\sigma}) \models O : \hat{v}' \wedge \\
 & \forall \vec{m} \in \hat{v}', \forall \{ \varsigma(x_0).O_0 \} \in \hat{\sigma}(\vec{m}, m) : \\
 & \quad \vec{m} \in \hat{\rho}(x_0) \wedge (\hat{\rho}, \hat{\sigma}) \models O_0 : \hat{v}_0 \wedge \hat{v}_0 \subseteq \hat{v}
 \end{aligned}$$

Note the similarity with the clause for function application.

Method update:

$$\begin{aligned} (\hat{\rho}, \hat{\sigma}) \models O.m := \varsigma(x_0).O_0 : \hat{v} \quad \underline{\text{iff}} \quad & (\hat{\rho}, \hat{\sigma}) \models O : \hat{v}' \wedge \\ & \forall \vec{m} \in \hat{v}' : \hat{\sigma}(\vec{m}, m) \neq \emptyset \Rightarrow \\ & \vec{m} \in \hat{v} \wedge \{\varsigma(x_0).O_0\} \in \hat{\sigma}(\vec{m}, m) \end{aligned}$$

Note the similarity with the clause for assignment in the courser analysis.

Concurrency

The aim is to determine which channels may be communicated over which other channels.

The judgements of the analysis have the form

$$(\hat{\rho}, \hat{\kappa}) \models P$$

and expresses that P has an acceptable analysis estimate in the context described by $\hat{\rho}$ and $\hat{\kappa}$. Here $\hat{\rho} : \mathbf{Var} \rightarrow \mathcal{P}(\mathbf{Ch})$ maps the variables to the sets of channels they may be bound to and $\hat{\kappa} : \mathbf{Ch} \rightarrow \mathcal{P}(\mathbf{Ch})$ maps the channels to the sets of channels that may be communicated over them.

When formulating the analysis we shall extend $\hat{\rho}$ to operate on channel names as well by taking $\hat{\rho}(n) = \{n\}$.

The specification:

$$\begin{aligned}(\widehat{\rho}, \widehat{\kappa}) \models \bar{u}t.P & \quad \underline{\text{iff}} \quad (\widehat{\rho}, \widehat{\kappa}) \models P \wedge \forall n \in \widehat{\rho}(u) : \widehat{\rho}(t) \subseteq \widehat{\kappa}(n) \\(\widehat{\rho}, \widehat{\kappa}) \models u(x).P & \quad \underline{\text{iff}} \quad (\widehat{\rho}, \widehat{\kappa}) \models P \wedge \forall n \in \widehat{\rho}(u) : \widehat{\kappa}(n) \subseteq \widehat{\rho}(x) \\(\widehat{\rho}, \widehat{\kappa}) \models (\nu n)P & \quad \underline{\text{iff}} \quad (\widehat{\rho}, \widehat{\kappa}) \models P \\(\widehat{\rho}, \widehat{\kappa}) \models P_1 \mid P_2 & \quad \underline{\text{iff}} \quad (\widehat{\rho}, \widehat{\kappa}) \models P_1 \wedge (\widehat{\rho}, \widehat{\kappa}) \models P_2\end{aligned}$$

For simplicity the above specification does not take into account that the semantics of the π -calculus allows α -renaming of names and variables — we can do so using a notion of canonical names and disciplined α -renaming.

Combining paradigms: CML

$$e ::= c \mid x \mid \text{fn } x_0 \Rightarrow e_0 \mid \text{fun } f x_0 \Rightarrow e_0 \mid e_1 e_2 \mid \text{if } e_0 \text{ then } e_1 \text{ else } e_2 \mid \dots$$
$$\mid e_1 := e_2 \mid e_1; e_2 \mid \text{let } x = \text{ref}_r \text{ in } e \mid \text{deref } e \mid \dots$$
$$\mid \text{send } e_1 \text{ on } e_2 \mid \text{receive } e \mid \text{let } x = \text{chan}_n \text{ in } e \mid \text{spawn } e \mid \dots$$

Judgements: $(\hat{\rho}, \hat{\sigma}, \hat{\kappa}) \models e : \hat{v}$

- $\hat{\rho} : \mathbf{Var} \rightarrow \widehat{\mathbf{Val}}$ records the abstract values bound to variables
- $\hat{\sigma} : \mathbf{Ref} \rightarrow \widehat{\mathbf{Val}}$ records the abstract values bound to reference cells
- $\hat{\kappa} : \mathbf{Ch} \rightarrow \widehat{\mathbf{Val}}$ records the abstract values communicated over channels.

Abstract values are constants \diamond , abstract closures $\langle \text{fn } x_0 \Rightarrow e_0 \rangle$, recursive abstract closures $\langle \text{fun } f x_0 \Rightarrow e_0 \rangle$, reference cells r and channels n .

Functional constructs:

$$(\hat{\rho}, \hat{\sigma}, \hat{\kappa}) \models c : \hat{v} \quad \underline{\text{iff}} \quad \diamond \in \hat{v}$$

$$(\hat{\rho}, \hat{\sigma}, \hat{\kappa}) \models x : \hat{v} \quad \underline{\text{iff}} \quad \hat{\rho}(x) \subseteq \hat{v}$$

$$(\hat{\rho}, \hat{\sigma}, \hat{\kappa}) \models \mathbf{fn} \ x_0 \Rightarrow e_0 : \hat{v} \quad \underline{\text{iff}} \quad \langle \mathbf{fn} \ x_0 \Rightarrow e_0 \rangle \in \hat{v}$$

$$(\hat{\rho}, \hat{\sigma}, \hat{\kappa}) \models \mathbf{fun} \ f \ x_0 \Rightarrow e_0 : \hat{v} \quad \underline{\text{iff}} \quad \langle \mathbf{fun} \ f \ x_0 \Rightarrow e_0 \rangle \in \hat{v}$$

$$(\hat{\rho}, \hat{\sigma}, \hat{\kappa}) \models e_1 \ e_2 : \hat{v} \quad \underline{\text{iff}} \quad (\hat{\rho}, \hat{\sigma}, \hat{\kappa}) \models e_1 : \hat{v}_1 \ \wedge \ (\hat{\rho}, \hat{\sigma}, \hat{\kappa}) \models e_2 : \hat{v}_2 \ \wedge$$

$$\forall \langle \mathbf{fn} \ x_0 \Rightarrow e_0 \rangle \in \hat{v}_1 : \hat{v}_2 \neq \emptyset \Rightarrow$$

$$[\hat{v}_2 \subseteq \hat{\rho}(x_0) \ \wedge \ (\hat{\rho}, \hat{\sigma}, \hat{\kappa}) \models e_0 : \hat{v}_0 \ \wedge \ \hat{v}_0 \subseteq \hat{v}] \ \wedge$$

$$\forall \langle \mathbf{fun} \ f \ x_0 \Rightarrow e_0 \rangle \in \hat{v}_1 : \hat{v}_2 \neq \emptyset \Rightarrow$$

$$[\langle \mathbf{fun} \ f \ x_0 \Rightarrow e_0 \rangle \in \hat{\rho}(f) \ \wedge \ \hat{v}_2 \subseteq \hat{\rho}(x_0) \ \wedge$$

$$(\hat{\rho}, \hat{\sigma}, \hat{\kappa}) \models e_0 : \hat{v}_0 \ \wedge \ \hat{v}_0 \subseteq \hat{v}]$$

Functional constructs (cont.):

$$(\hat{\rho}, \hat{\sigma}, \hat{\kappa}) \models \mathbf{if} \ e_0 \ \mathbf{then} \ e_1 \ \mathbf{else} \ e_2 : \hat{v} \ \underline{\mathbf{iff}} \ (\hat{\rho}, \hat{\sigma}, \hat{\kappa}) \models e_0 : \hat{v}_0 \wedge$$

$$\diamond \in \hat{v}_0 \Rightarrow [(\hat{\rho}, \hat{\sigma}, \hat{\kappa}) \models e_1 : \hat{v}_1 \wedge \hat{v}_1 \sqsubseteq \hat{v} \wedge$$

$$(\hat{\rho}, \hat{\sigma}, \hat{\kappa}) \models e_2 : \hat{v}_2 \wedge \hat{v}_2 \sqsubseteq \hat{v}]$$

Imperative constructs:

$$\begin{aligned}
 (\hat{\rho}, \hat{\sigma}, \hat{\kappa}) \models e_1 := e_2 : \hat{v} \quad \underline{\text{iff}} \quad & (\hat{\rho}, \hat{\sigma}, \hat{\kappa}) \models e_1 : \hat{v}_1 \wedge \\
 & (\hat{\rho}, \hat{\sigma}, \hat{\kappa}) \models e_2 : \hat{v}_2 \wedge \\
 & \diamond \in \hat{v} \wedge \forall r \in \hat{v}_1 : \hat{v}_2 \subseteq \hat{\sigma}(r)
 \end{aligned}$$

$$\begin{aligned}
 (\hat{\rho}, \hat{\sigma}, \hat{\kappa}) \models \text{deref } e : \hat{v} \quad \underline{\text{iff}} \quad & (\hat{\rho}, \hat{\sigma}, \hat{\kappa}) \models e : \hat{v}' \wedge \\
 & \forall r \in \hat{v}' : \hat{\sigma}(r) \subseteq \hat{v}
 \end{aligned}$$

$$(\hat{\rho}, \hat{\sigma}, \hat{\kappa}) \models \text{let } x = \text{ref}_r \text{ in } e : \hat{v} \quad \underline{\text{iff}} \quad r \in \hat{\rho}(x) \wedge (\hat{\rho}, \hat{\sigma}, \hat{\kappa}) \models e : \hat{v}$$

$$\begin{aligned}
 (\hat{\rho}, \hat{\sigma}, \hat{\kappa}) \models e_1; e_2 : \hat{v} \quad \underline{\text{iff}} \quad & (\hat{\rho}, \hat{\sigma}, \hat{\kappa}) \models e_1 : \hat{v}' \wedge \\
 & (\hat{\rho}, \hat{\sigma}, \hat{\kappa}) \models e_2 : \hat{v}
 \end{aligned}$$

Concurrency constructs:

$$\begin{aligned}
 (\hat{\rho}, \hat{\sigma}, \hat{\kappa}) \models \text{send } e_1 \text{ on } e_2 : \hat{v} \quad \underline{\text{iff}} \quad & (\hat{\rho}, \hat{\sigma}, \hat{\kappa}) \models e_1 : \hat{v}_1 \wedge \\
 & (\hat{\rho}, \hat{\sigma}, \hat{\kappa}) \models e_2 : \hat{v}_2 \wedge \\
 & \diamond \in \hat{v} \wedge \forall n \in \hat{v}_2 : \hat{v}_1 \subseteq \hat{\kappa}(n)
 \end{aligned}$$

$$\begin{aligned}
 (\hat{\rho}, \hat{\sigma}, \hat{\kappa}) \models \text{receive } e : \hat{v} \quad \underline{\text{iff}} \quad & (\hat{\rho}, \hat{\sigma}, \hat{\kappa}) \models e : \hat{v}' \wedge \\
 & \forall n \in \hat{v}' : \hat{\kappa}(n) \subseteq \hat{v}
 \end{aligned}$$

$$(\hat{\rho}, \hat{\sigma}, \hat{\kappa}) \models \text{let } x = \text{chan}_n \text{ in } e : \hat{v} \quad \underline{\text{iff}} \quad n \in \hat{\rho}(x) \wedge (\hat{\rho}, \hat{\sigma}, \hat{\kappa}) \models e : \hat{v}$$

Concurrency constructs (cont.):

$$\begin{aligned}
 (\hat{\rho}, \hat{\sigma}, \hat{\kappa}) \models \text{spawn } e : \hat{v} \quad \underline{\text{iff}} \quad & \diamond \in \hat{v} \wedge (\hat{\rho}, \hat{\sigma}, \hat{\kappa}) \models e : \hat{v}' \wedge \\
 & \forall \langle \text{fn } x_0 \Rightarrow e_0 \rangle \in \hat{v}' : \\
 & \quad \diamond \subseteq \hat{\rho}(x_0) \wedge (\hat{\rho}, \hat{\sigma}, \hat{\kappa}) \models e_0 : \hat{v}_0 \\
 & \forall \langle \text{fun } f \ x_0 \Rightarrow e_0 \rangle \in \hat{v}' : \\
 & \quad \langle \text{fun } f \ x_0 \Rightarrow e_0 \rangle \in \hat{\rho}(f) \wedge \diamond \subseteq \hat{\rho}(x_0) \wedge \\
 & \quad (\hat{\rho}, \hat{\sigma}, \hat{\kappa}) \models e_0 : \hat{v}_0
 \end{aligned}$$

Further Ressources

Flow Logic has been used for a number of languages:

- analyses of classical languages:
 - functional, object-oriented, concurrent, imperative.
- analyses of process algebras:
 - pi, spi, ambients (several variants), LySa, μ Klaim, ...

Techniques have been developed for:

- ensuring well-definedness of specifications,
- proving semantic correctness,
- obtaining efficient implemenations (including techniques for reducing the complexity).

Flow Logic has been used for analyses in a variety of areas:

- showing that data flow analysis, control flow analysis, abstract interpretation all fit the framework,
- showing how to ensure security properties like confidentiality, integrity, authenticity, discretionary access control, mandatory access control.

Flow Logic is being used in the EU-project Sensoria for analysing service orientation of overlay computers.

For further information please consult:

<http://www2.imm.dtu.dk/~nielson/FlowLogic.html>

Optional Assignment

If you want to earn full credits for the summerschool by doing the assignment on Flow Logic:

- Find and read a suitable reference on the process algebra CSP;
 - make sure to select a version where values are passed as part of communication.
- Define the syntax and operational semantics of CSP;
 - explain the choices made;
 - informally argue why it captures the intention behind CSP.

(Continued on the next page.)

- Develop a simple control flow analysis in the Flow Logic format for determining which values reach what places;
 - explain the choices made;
 - illustrate the analysis on a well-chosen example;
 - argue in detail for the well-formedness of the specification;
 - formally prove the subject reduction result;
 - establish the Moore family result.

(Continued on the next page.)

Try to answer one of the following questions:

- Estimate the complexity of the Flow Logic specification.
- Develop an interesting analysis (perhaps dealing with security) on top of your simple control flow analysis.

Write the report in the form of a conference style paper.

It is quite acceptable to seek inspiration in these slides and in published papers.