Analysis of Mobile Ambients

Hanne Riis Nielson & Flemming Nielson

Informatics & Mathematical Modelling

Technical University of Denmark

Overview

- The aim of the analysis
- Specification of the analysis
- Properties of the analysis
- Implementation of the analysis
- Secure firewalls

The analysis is adapted from Flemming Nielson, Hanne Riis Nielson, René Rydhof Hansen: *Validating Firewalls using Flow Logic*, to appear in TCS.

The aim 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:

A: [p: [out A. in B. · · ·] | · · ·] | B: [open p. · · · | · · ·]

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

$\boxed{\boxed{\text{out A. in B. } \cdots}^{\,p} \mid \cdots}^{\,A} \quad \mid \quad \boxed{\text{open p. } \cdots \mid \cdots}^{\,B}$

Groups:

p: $\mathbb{P}$acket

A, B: $\mathbb{S}$ite

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 notion of approximation

A: [p: [out A. in B. ⋯] | ⋯] | B: [open p. ⋯ | ⋯]

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 $\mathbb{P}$

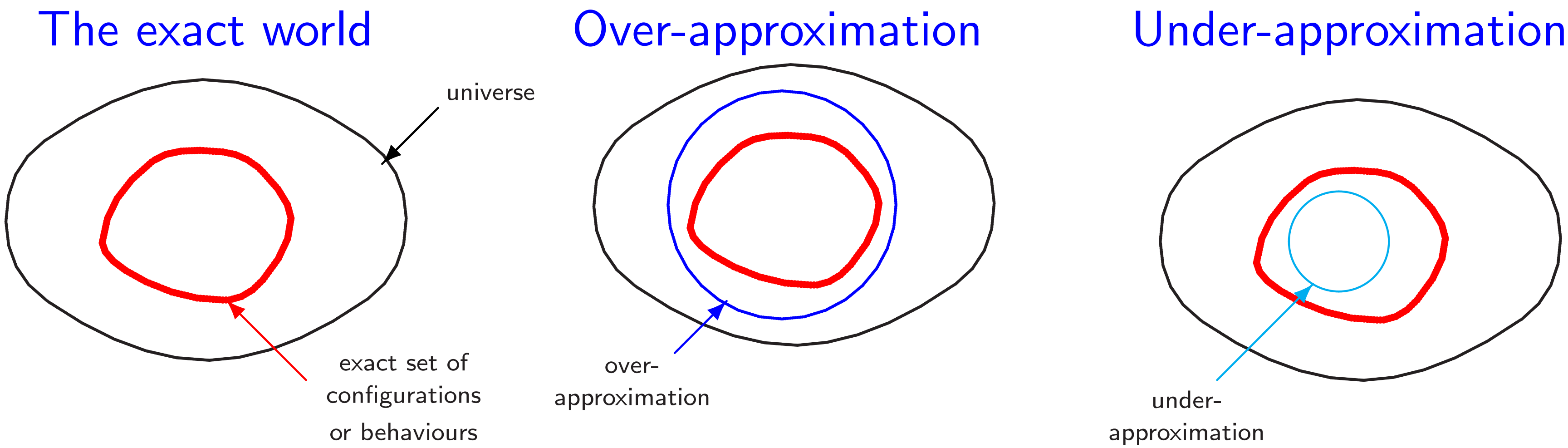
An unacceptable analysis result

– $\mathbb{S}$ may turn up inside $\mathbb{P}$

but

– $\mathbb{P}$ never turns up inside $\mathbb{S}$

The nature of approximation



Our analysis of mobile ambients: over-approximation

Specification of the analysis

- The analysis estimates
- The flow logic approach
- Syntax-directed definition

Analysis estimates

The analysis estimate

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

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 possed by an ambient in group μ

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

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

Judgements of the analysis

$$\boxed{\mathcal{I} \models^{\mu}_{\Gamma} P}$$

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

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

Example

$$\mathcal{I} \models^{\star}_{\Gamma} \mathsf{A}\,[\mathsf{p}\,[\mathsf{out\,A.\ in\,B}\,]\,] \mid \mathsf{B}\,[\mathsf{open\,p}]$$

holds for

Γ	**Name** $\to$ **Group**
A	$\mathbb{S}$
B	$\mathbb{S}$
p	$\mathbb{P}$

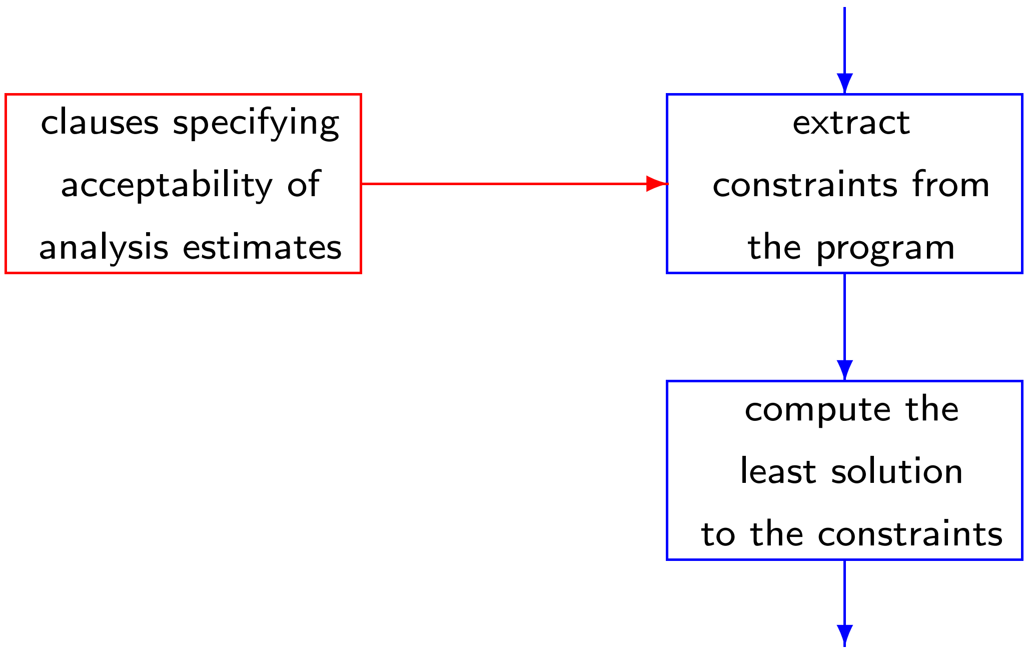
$\mathcal{I}$	**Group** $\to \mathcal{P}($**Group** $\cup$ **Cap**$)$
$\star$	$\{\mathbb{S}, \mathbb{P}\}$
$\mathbb{S}$	$\{\mathbb{P}, \mathbb{S}, \mathsf{in}\ \mathbb{S}, \mathsf{out}\ \mathbb{S}, \mathsf{open}\ \mathbb{P}\}$
$\mathbb{P}$	$\{\mathsf{in}\ \mathbb{S}, \mathsf{out}\ \mathbb{S}\}$

The analysis result shows that

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

The Flow Logic approach

Specification Implementation



Acceptable analysis results

- Each acceptable analysis estimate for a composite program must also be an acceptable analysis estimate for its subprograms; 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

$\mathcal{I} \models^{\star}_{\Gamma} (\nu\, n\colon \mu)\, P$	iff	$\mathcal{I} \models^{\star}_{\Gamma[n \mapsto \mu]} P$	update group environment; check process
$\mathcal{I} \models^{\star}_{\Gamma} (\nu\, \mu)\, P$	iff	$\mathcal{I} \models^{\star}_{\Gamma} P$	check process
$\mathcal{I} \models^{\star}_{\Gamma} \mathbf{0}$	iff	true	nothing to check
$\mathcal{I} \models^{\star}_{\Gamma} P_1 \mid P_2$	iff	$\mathcal{I} \models^{\star}_{\Gamma} P_1 \ \wedge\ \mathcal{I} \models^{\star}_{\Gamma} P_2$	check both branches
$\mathcal{I} \models^{\star}_{\Gamma} !P$	iff	$\mathcal{I} \models^{\star}_{\Gamma} P$	check process; ignore multiplicity
$\mathcal{I} \models^{\star}_{\Gamma} n\,[P]$	iff	$\mu \in \mathcal{I}(\star) \ \wedge\ \mathcal{I} \models^{\mu}_{\Gamma} P$ where $\mu = \Gamma(n)$	μ is inside $\star$; check process

Example: subambient

Checking

$$\mathcal{I} \models^{\star}_{\Gamma} \mathsf{A}\,[\mathsf{p}\,[\mathsf{out\,A.\ in\,B}\,]\,] \mid \mathsf{B}\,[\mathsf{open\,p}]$$

involves checking

$$\mathbb{S} \in \mathcal{I}(\star) \text{ and } \mathcal{I} \models^{\mathbb{S}}_{\Gamma} \mathsf{p}\,[\mathsf{out\,A.\ in\,B}\,]$$

Γ	**Name** $\to$ **Group**
A	$\mathbb{S}$
B	$\mathbb{S}$
p	$\mathbb{P}$

$\mathcal{I}$	**Group** $\to \mathcal{P}($**Group** $\cup$ **Cap**$)$
$\star$	$\{\mathbb{S}, \mathbb{P}\}$
$\mathbb{S}$	$\{\mathbb{P}, \mathbb{S}, \mathsf{in}\ \mathbb{S}, \mathsf{out}\ \mathbb{S}, \mathsf{open}\ \mathbb{P}\}$
$\mathbb{P}$	$\{\mathsf{in}\ \mathbb{S}, \mathsf{out}\ \mathbb{S}\}$

Analysis of in-capability

$$\mathcal{I} \models^{\star}_{\Gamma} \mathsf{in}\, n.\, P \text{ iff } \quad \mathsf{in}\ \mu \in \mathcal{I}(\star) \wedge \mathcal{I} \models^{\star}_{\Gamma} P \wedge$$

$$\forall\, \mu^a, \mu^p : \ \mathsf{in}\ \mu \in \mathcal{I}(\mu^a) \wedge$$ — μ^a has the capability in μ

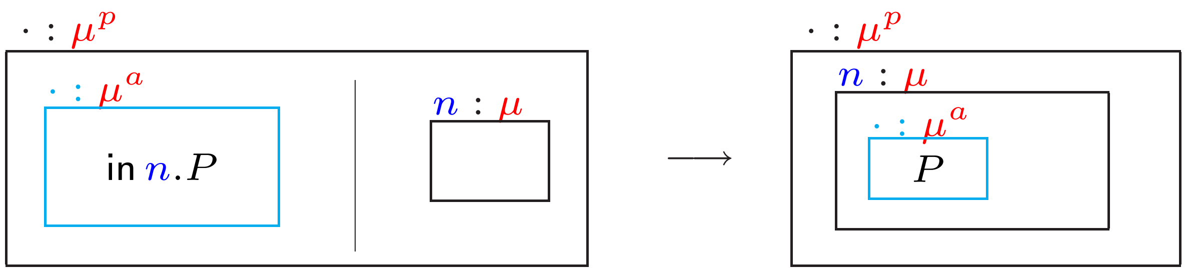
$$\mu^a \in \mathcal{I}(\mu^p) \wedge$$ — μ^p is parent of μ^a

$$\mu \in \mathcal{I}(\mu^p)$$ — μ^a has a sibling in group μ

$$\Rightarrow \ \mu^a \in \mathcal{I}(\mu)$$ — μ^a may move into μ

where $\mu = \Gamma(n)$

Mimicking the semantics:



Example: in-capability

Checking

$$\mathcal{I} \models^{\star}_{\Gamma} \mathsf{A}\,[\mathsf{p}\,[\mathsf{out}\,\mathsf{A}.\ \mathsf{in}\,\mathsf{B}\ ]\ ]\ \mid\ \mathsf{B}\,[\mathsf{open}\,\mathsf{p}]$$

involves checking

$$\mathcal{I} \models^{\mathbb{P}}_{\Gamma} \mathsf{in}\,\mathsf{B}$$

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

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

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

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

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

$\mathcal{I} \models^{\star}_{\Gamma} \mathsf{out}\, n.\ P$ <u>iff</u> $\mathsf{out}\ \mu \in \mathcal{I}(\star)\ \wedge\ \mathcal{I} \models^{\star}_{\Gamma} P\ \wedge$

$\forall\ \mu^a, \mu^g :\ \mathsf{out}\ \mu \in \mathcal{I}(\mu^a) \wedge$ — μ^a has the capability out μ

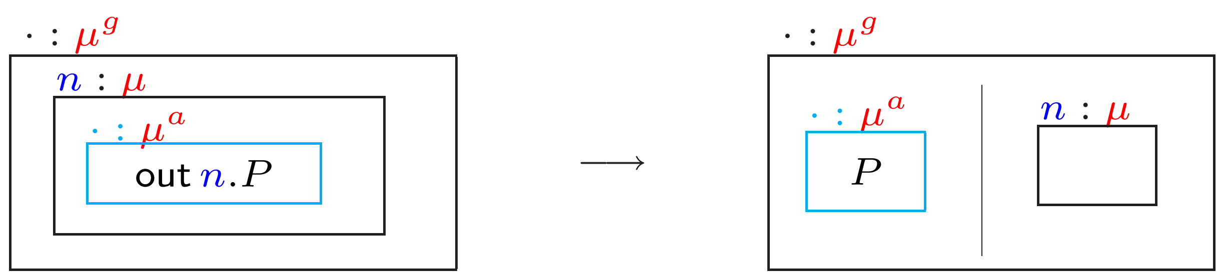
$\mu^a \in \mathcal{I}(\mu)\ \wedge$ — μ is parent of μ^a

$\mu \in \mathcal{I}(\mu^g)$ — 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

$$\mathcal{I} \models^{\star}_{\Gamma} \mathsf{A}\,[\mathsf{p}\,[\mathsf{out\,A.\ in\,B}]\,]\ \mid\ \mathsf{B}\,[\mathsf{open\,p}]$$

involves checking

$$\mathcal{I} \models^{\mathbb{P}}_{\Gamma} \mathsf{out\,A.\ in\,B}$$

Γ	**Name** $\to$ **Group**
A	$\mathbb{S}$
B	$\mathbb{S}$
p	$\mathbb{P}$

$\mathcal{I}$	**Group** $\to \mathcal{P}($**Group** $\cup$ **Cap**$)$
$\star$	$\{\mathbb{S}, \mathbb{P}\}$
$\mathbb{S}$	$\{\mathbb{P}, \mathbb{S}, \mathsf{in}\ \mathbb{S}, \mathsf{out}\ \mathbb{S}, \mathsf{open}\ \mathbb{P}\}$
$\mathbb{P}$	$\{\mathsf{in}\ \mathbb{S}, \mathsf{out}\ \mathbb{S}\}$

which holds because $\mathcal{I} \models^{\mathbb{P}}_{\Gamma} \mathsf{in\,B}$ and $\mathsf{out}\,\mathbb{S} \in \mathcal{I}(\mathbb{P})$ and

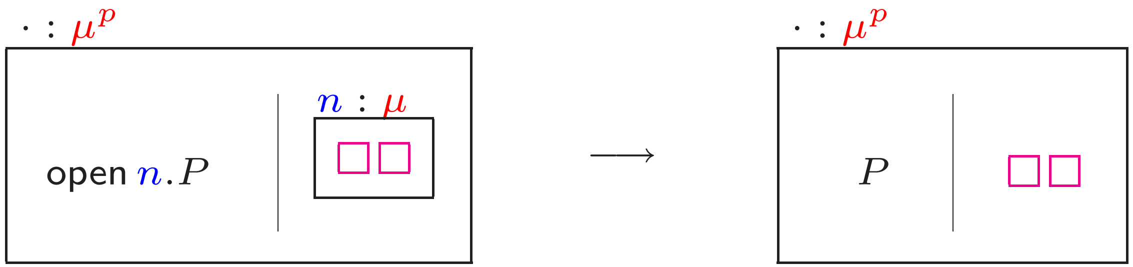
$$\mathsf{out}\,\mathbb{S} \in \mathcal{I}(\mu^a) \wedge \mu^a \in \mathcal{I}(\mathbb{S}) \wedge \mathbb{S} \in \mathcal{I}(\mu^g) \Rightarrow \mu^a \in \mathcal{I}(\mu^g)$$

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

$$\begin{array}{lll}
\mathcal{I} \models^{\star}_{\Gamma} \mathsf{open}\, n.\ P \ \underline{\text{iff}} & \mathsf{open}\ \mu \in \mathcal{I}(\star) \ \wedge\ \mathcal{I} \models^{\star}_{\Gamma} P \ \wedge & \\
& \forall\ \mu^p :\ \ \mathsf{open}\ \mu \in \mathcal{I}(\mu^p) \ \wedge & \mu^p \text{ has the capability open } \mu \\
& \qquad \mu \in \mathcal{I}(\mu^p) & \mu \text{ is sibling} \\
& \quad \Rightarrow\ \mathcal{I}(\mu) \subseteq \mathcal{I}(\mu^p) & \text{anything in } \mu \text{ may appear in } \mu^p \\
& \text{where } \mu = \Gamma(n) &
\end{array}$$

Mimicking the semantics:



Example: open-capability

Checking

$$\mathcal{I} \models^{\star}_{\Gamma} \mathsf{A}\,[\mathsf{p}\,[\mathsf{out\,A.\ in\,B}\,]] \mid \mathsf{B}\,[\mathsf{open\,p}]$$

involves checking

$$\mathcal{I} \models^{\mathbb{S}}_{\Gamma} \mathsf{open\,p}$$

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

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

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

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

holds for $\mu^p = \mathbb{S}$

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:

$$
\begin{array}{ccccccccc}
P_1 & \rightarrow & P_2 & \rightarrow & \cdots & \rightarrow & P_i & \rightarrow & \cdots \\
\updownarrow \models^\star_\Gamma & & \updownarrow \models^\star_\Gamma & & & & \updownarrow \models^\star_\Gamma & & \\
\mathcal{I} & & \mathcal{I} & & \cdots & & \mathcal{I} & & \cdots
\end{array}
$$

Subject reduction result:

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

Familiar from type systems

Proof of subject reduction result

Lemma: The analysis is invariant under the structural congruence:

$$\text{If} \quad P \equiv Q \quad \text{then} \quad \mathcal{I} \models^{\star}_{\Gamma} P \quad \text{if and only if} \quad \mathcal{I} \models^{\star}_{\Gamma} Q$$

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

Theorem: The analysis is preserved under the transition relation:

$$\text{If} \quad P \longrightarrow Q \quad \text{and} \quad \mathcal{I} \models^{\star}_{\Gamma} P \quad \text{then} \quad \mathcal{I} \models^{\star}_{\Gamma} Q$$

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

Moore family property

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

The set $\{\mathcal{I} \mid \mathcal{I} \models^{\star}_{\Gamma} P\}$ is a Moore family

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

The proof is by structural induction on P.

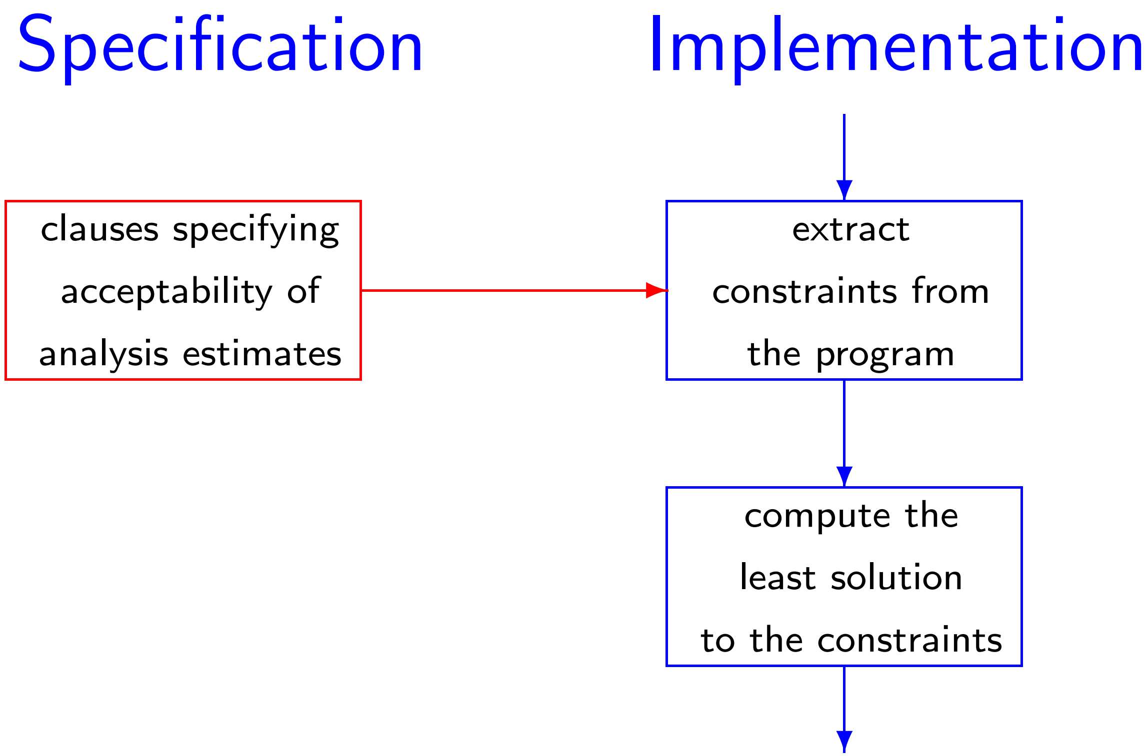
Corollaries:

- 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

Implementation of the analysis

- Generation of constraints
- Solving constraints

The Flow Logic approach



Generation of constraints

Analysis specification

$$
\begin{array}{lll}
\mathcal{I} \models^{\star}_{\Gamma} (\nu\, n : \mu)\, P & \underline{\text{iff}} & \mathcal{I} \models^{\star}_{\Gamma[n \mapsto \mu]} P \\
\mathcal{I} \models^{\star}_{\Gamma} (\nu\, \mu)\, P & \underline{\text{iff}} & \mathcal{I} \models^{\star}_{\Gamma} P \\
\mathcal{I} \models^{\star}_{\Gamma} \mathbf{0} & \underline{\text{iff}} & \text{true} \\
\mathcal{I} \models^{\star}_{\Gamma} P_1 \mid P_2 & \underline{\text{iff}} & \mathcal{I} \models^{\star}_{\Gamma} P_1 \wedge \mathcal{I} \models^{\star}_{\Gamma} P_2 \\
\mathcal{I} \models^{\star}_{\Gamma} !P & \underline{\text{iff}} & \mathcal{I} \models^{\star}_{\Gamma} P \\
\mathcal{I} \models^{\star}_{\Gamma} n\,[P] & \underline{\text{iff}} & \mu \in \mathcal{I}(\star) \wedge \mathcal{I} \models^{\mu}_{\Gamma} P \\
 & & \text{where } \mu = \Gamma(n)
\end{array}
$$

Constraint generation

$$
\begin{array}{rcl}
\mathcal{C}^{\star}_{\Gamma}[\![(\nu\, n : \mu)\, P]\!] & = & \mathcal{C}^{\star}_{\Gamma[n \mapsto \mu]}[\![P]\!] \\
\mathcal{C}^{\star}_{\Gamma}[\![(\nu\, \mu)\, P]\!] & = & \mathcal{C}^{\star}_{\Gamma}[\![P]\!] \\
\mathcal{C}^{\star}_{\Gamma}[\![\mathbf{0}]\!] & = & \emptyset \\
\mathcal{C}^{\star}_{\Gamma}[\![P_1 \mid P_2]\!] & = & \mathcal{C}^{\star}_{\Gamma}[\![P_1]\!] \cup \mathcal{C}^{\star}_{\Gamma}[\![P_2]\!] \\
\mathcal{C}^{\star}_{\Gamma}[\![!P]\!] & = & \mathcal{C}^{\star}_{\Gamma}[\![P]\!] \\
\mathcal{C}^{\star}_{\Gamma}[\![n\,[P]]\!] & = & \{\mu \in \mathfrak{I}(\star)\} \cup \mathcal{C}^{\mu}_{\Gamma}[\![P]\!] \\
 & & \text{where } \mu = \Gamma(n)
\end{array}
$$

Constraints for in-capability

Analysis specification

$$
\begin{aligned}
&\mathcal{I} \models^{\star}_{\Gamma} \mathsf{in}\, n.\ P \\
&\underline{\text{iff}} \quad \mathsf{in}\ \mu \in \mathcal{I}(\star)\ \wedge\ \mathcal{I} \models^{\star}_{\Gamma} P\ \wedge \\
&\qquad \forall\ \mu^a, \mu^p :\ \mathsf{in}\ \mu \in \mathcal{I}(\mu^a)\ \wedge \\
&\qquad\qquad \mu^a \in \mathcal{I}(\mu^p)\ \wedge \\
&\qquad\qquad \mu \in \mathcal{I}(\mu^p) \\
&\qquad\qquad \Rightarrow\ \mu^a \in \mathcal{I}(\mu) \\
&\qquad \text{where } \mu = \Gamma(n)
\end{aligned}
$$

Constraint generation

$$
\begin{aligned}
&\mathcal{C}^{\star}_{\Gamma}[\![\mathsf{in}\, n.\ P]\!] \\
&= \{\mathsf{in}\ \mu \in \mathfrak{I}(\star)\} \cup \mathcal{C}^{\star}_{\Gamma}[\![P]\!] \cup \\
&\quad \{\ \mathsf{in}\ \mu \in \mathfrak{I}(\mu^a)\ \wedge \\
&\quad\quad \mu^a \in \mathfrak{I}(\mu^p)\ \wedge \\
&\quad\quad \mu \in \mathfrak{I}(\mu^p) \\
&\quad\quad \Rightarrow\ \mu^a \in \mathfrak{I}(\mu) \\
&\quad\ \mid \mu, \mu^a, \mu^p \in \mathbf{Group}; \mu = \Gamma(n)\}
\end{aligned}
$$

Conditional constraints

Constraints for out-capability

Analysis specification

$$\mathcal{I} \models^{\star}_{\Gamma} \text{out}\, n.\ P$$

$$\begin{aligned}
\underline{\text{iff}}\quad & \text{out}\ \mu \in \mathcal{I}(\star)\ \wedge\ \mathcal{I} \models^{\star}_{\Gamma} P\ \wedge \\
& \forall\ \mu^a, \mu^g :\ \ \text{out}\ \mu \in \mathcal{I}(\mu^a) \wedge \\
& \qquad\qquad \mu^a \in \mathcal{I}(\mu)\ \wedge \\
& \qquad\qquad \mu \in \mathcal{I}(\mu^g) \\
& \qquad\quad \Rightarrow\ \ \mu^a \in \mathcal{I}(\mu^g) \\
& \text{where}\ \mu = \Gamma(n)
\end{aligned}$$

Constraint generation

$$\mathcal{C}^{\star}_{\Gamma}[\![\text{out}\, n.\ P]\!]$$

$$\begin{aligned}
= \quad & \{\text{out}\ \mu \in \mathfrak{I}(\star)\} \cup \mathcal{C}^{\star}_{\Gamma}[\![P]\!] \cup \\
& \{\ \text{out}\ \mu \in \mathfrak{I}(\mu^a)\ \wedge \\
& \quad \mu^a \in \mathfrak{I}(\mu)\ \wedge \\
& \quad \mu \in \mathfrak{I}(\mu^g) \\
& \ \Rightarrow \mu^a \in \mathfrak{I}(\mu^g) \\
& \mid \mu, \mu^a, \mu^g \in \mathbf{Group}; \mu = \Gamma(n)\}
\end{aligned}$$

Constraints for open-capability

Analysis specification

$$
\begin{aligned}
&\mathcal{I} \models^{\star}_{\Gamma} \mathsf{open}\, n.\ P \\
&\underline{\text{iff}} \quad \mathsf{open}\ \mu \in \mathcal{I}(l) \ \wedge\ \mathcal{I} \models^{\star}_{\Gamma} P \ \wedge \\
&\qquad \forall\ \mu^p :\ \ \mathsf{open}\ \mu \in \mathcal{I}(\mu^p) \ \wedge \\
&\qquad\qquad \mu \in \mathcal{I}(\mu^p) \\
&\qquad\quad \Rightarrow\ \ \mathcal{I}(\mu) \subseteq \mathcal{I}(\mu^p) \\
&\qquad \text{where } \mu = \Gamma(n)
\end{aligned}
$$

Constraint generation

$$
\begin{aligned}
&\mathcal{C}^{\star}_{\Gamma}[\![\mathsf{open}\, n.\ P]\!] \\
&= \quad \{\mathsf{open}\ \mu \in \mathfrak{I}(\star)\} \cup \mathcal{C}^{\star}_{\Gamma}[\![P]\!] \cup \\
&\qquad \{\ \mathsf{open}\ \mu \in \mathfrak{I}(\mu^p) \ \wedge \\
&\qquad\quad \mu \in \mathfrak{I}(\mu^p) \ \wedge \\
&\qquad\quad \xi \in \mathfrak{I}(\mu) \Rightarrow \xi \in \mathfrak{I}(\mu^p) \\
&\qquad \mid\ \mu, \mu^p \in \mathbf{Group}; \mu = \Gamma(n); \\
&\qquad\quad \xi \in \mathbf{Group} \cup \mathbf{Cap}\}
\end{aligned}
$$

Example:

For A [p [out A. in B]] | B [open p] we obtain the following constraints:

$$\mathbb{S} \in \mathfrak{I}(\star)$$
$$\mathbb{P} \in \mathfrak{I}(\mathbb{S})$$
$$\text{out } \mathbb{S} \in \mathfrak{I}(\mathbb{P})$$
$$\text{in } \mathbb{S} \in \mathfrak{I}(\mathbb{P})$$
$$\mathbb{S} \in \mathfrak{I}(\star)$$
$$\text{open } \mathbb{P} \in \mathfrak{I}(\mathbb{S})$$

$$\forall \mu^a, \mu^p : \text{ in } \mathbb{S} \in \mathfrak{I}(\mu^a) \wedge \mu^a \in \mathfrak{I}(\mu^p) \wedge \mathbb{S} \in \mathfrak{I}(\mu^p) \Rightarrow \mu^a \in \mathfrak{I}(\mathbb{S})$$

$$\forall \mu^a, \mu^g : \text{ out } \mathbb{S} \in \mathfrak{I}(\mu^a) \wedge \mu^a \in \mathfrak{I}(\mathbb{S}) \wedge \mathbb{S} \in \mathfrak{I}(\mu^g) \Rightarrow \mu^a \in \mathfrak{I}(\mu^g)$$

$$\forall \mu^p, \xi : \text{ open } \mathbb{P} \in \mathfrak{I}(\mu^p) \wedge \mathbb{P} \in \mathfrak{I}(\mu^p) \wedge \xi \in \mathfrak{I}(\mathbb{P}) \Rightarrow \xi \in \mathfrak{I}(\mu^p)$$

Summary:

- 3 sets $\mathfrak{I}(\star)$, $\mathfrak{I}(\mathbb{S})$ and $\mathfrak{I}(\mathbb{P})$ all being subsets of $\mathbf{Group} \cup \mathbf{Cap}$
- the constraints can be formulated as a function

  $$\text{F: } \mathcal{P}(\mathbf{Group} \cup \mathbf{Cap})^3 \rightarrow \mathcal{P}(\mathbf{Group} \cup \mathbf{Cap})^3$$
- we want the least fixed point of F — this is the best solution to the constraint system

Solving constraints

- Initialisation

  $\mathfrak{I}(\star) := \emptyset; \mathfrak{I}(\mathbb{S}) := \emptyset; \mathfrak{I}(\mathbb{P}) := \emptyset;$

- Iteration

  `while` $\mathfrak{I}(\ell) \neq \mathsf{F}_\ell(\mathfrak{I}(\star), \mathfrak{I}(\mathbb{S}), \mathfrak{I}(\mathbb{P}))$ for some $\ell \in \{\star, \mathbb{S}, \mathbb{P}\}$

  `do`

  $\qquad \mathfrak{I}(\ell) := \mathsf{F}_\ell(\mathfrak{I}(\star), \mathfrak{I}(\mathbb{S}), \mathfrak{I}(\mathbb{P}))$

The algorithm terminates and computes the least fixed point of F.

In practice we propagate smaller contributions than F_ℓ, e.g. one constraint at a time.

Solving the constraints

$\mathbb{S} \in \mathfrak{I}(\star)$

$\mathbb{P} \in \mathfrak{I}(\mathbb{S})$

$\text{out } \mathbb{S} \in \mathfrak{I}(\mathbb{P})$

$\text{in } \mathbb{S} \in \mathfrak{I}(\mathbb{P})$

$\mathbb{S} \in \mathfrak{I}(\star)$

$\text{open } \mathbb{P} \in \mathfrak{I}(\mathbb{S})$

$$\forall \mu^a, \mu^p : \text{ in } \mathbb{S} \in \mathfrak{I}(\mu^a) \wedge \mu^a \in \mathfrak{I}(\mu^p) \wedge \mathbb{S} \in \mathfrak{I}(\mu^p) \Rightarrow \mu^a \in \mathfrak{I}(\mathbb{S})$$

$$\forall \mu^a, \mu^g : \text{ out } \mathbb{S} \in \mathfrak{I}(\mu^a) \wedge \mu^a \in \mathfrak{I}(\mathbb{S}) \wedge \mathbb{S} \in \mathfrak{I}(\mu^g) \Rightarrow \mu^a \in \mathfrak{I}(\mu^g)$$

$$\forall \mu^p, \xi : \text{ open } \mathbb{P} \in \mathfrak{I}(\mu^p) \wedge \mathbb{P} \in \mathfrak{I}(\mu^p) \wedge \xi \in \mathfrak{I}(\mathbb{P}) \Rightarrow \xi \in \mathfrak{I}(\mu^p)$$

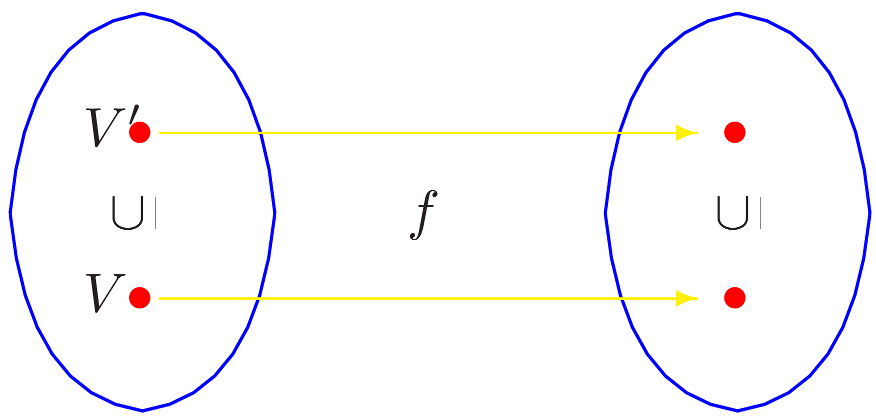
$\mathfrak{I}$	$\star$	$\mathbb{S}$	$\mathbb{P}$	
0	$\emptyset$	$\emptyset$	$\emptyset$	initialisation
5	$\{\mathbb{S}\}$	$\{\mathbb{P}, \text{open } \mathbb{P}\}$	$\{\text{in } \mathbb{S}, \text{out } \mathbb{S}\}$	constraints of left column
6	$\{\mathbb{S}, \mathbb{P}\}$	$\{\mathbb{P}, \text{open } \mathbb{P}\}$	$\{\text{in } \mathbb{S}, \text{out } \mathbb{S}\}$	out: $\mu^a = \mathbb{P}$; $\mu^g = \star$
7	$\{\mathbb{S}, \mathbb{P}\}$	$\{\mathbb{P}, \text{open } \mathbb{P}, \text{in } \mathbb{S}, \text{out } \mathbb{S}\}$	$\{\text{in } \mathbb{S}, \text{out } \mathbb{S}\}$	open: $\mu^p = \mathbb{S}$; $\xi \in \{\text{in } \mathbb{S}, \text{out } \mathbb{S}\}$
8	$\{\mathbb{S}, \mathbb{P}\}$	$\{\mathbb{S}, \mathbb{P}, \text{open } \mathbb{P}, \text{in } \mathbb{S}, \text{out } \mathbb{S}\}$	$\{\text{in } \mathbb{S}, \text{out } \mathbb{S}\}$	in: $\mu^a = \mathbb{S}$; $\mu^p = \star$

Why does it work? (1)

A function $f : \mathcal{P}(S) \to \mathcal{P}(S)$ is a monotone function if

$$V \subseteq V' \quad \Rightarrow \quad f(V) \subseteq f(V')$$

(the larger the argument – the larger the result)



$\mathcal{P}(S)$ is the set of all subsets of the set S; e.g. $\mathcal{P}(\{0,1\}) = \{\emptyset, \{0\}, \{1\}, \{0,1\}\}$.

Why does it work? (2)

A set L equipped with an ordering $\subseteq$ satisfies the Ascending Chain Condition if all chains

$$V_0 \subseteq V_1 \subseteq V_2 \subseteq V_3 \subseteq \cdots$$

stabilise, that is, if there exists some n such that $V_n = V_{n+1} = V_{n+2} = \cdots$

If S is a finite set then $\mathcal{P}(S)$ equipped with the subset ordering $\subseteq$ satisfies the Ascending Chain Condition — the chains cannot grow forever since each element is a subset of a finite set.

Fact For a given process $\mathbf{Group} \cup \mathbf{Cap}$ will be a finite set so $\mathcal{P}(\mathbf{Group} \cup \mathbf{Cap})$ with the subset ordering satisfies the Ascending Chain Condition.

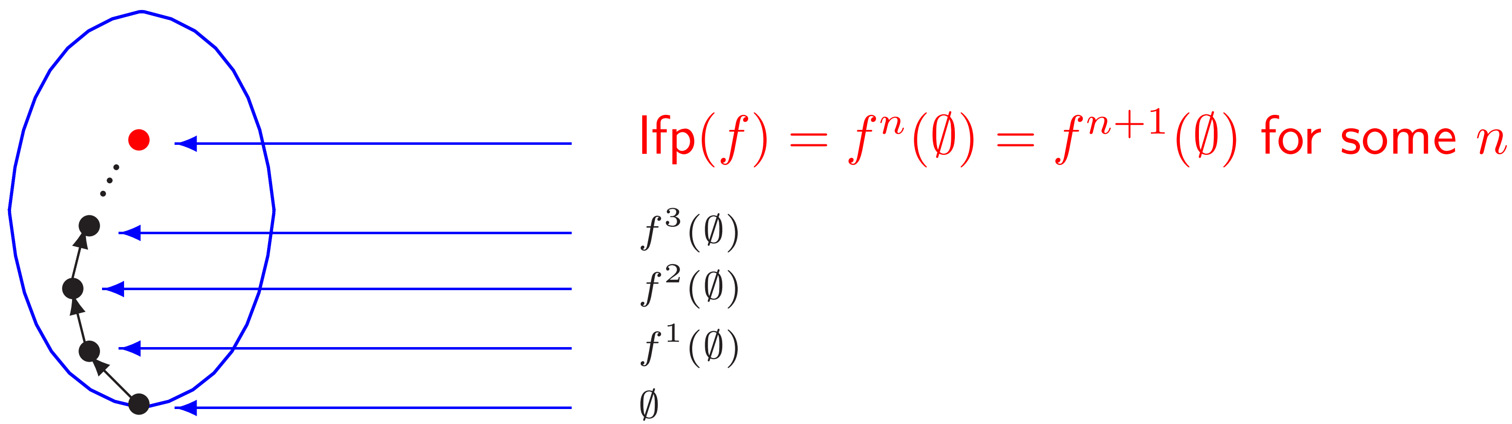
Why does it work? (3)

Let $f : \mathcal{P}(S) \to \mathcal{P}(S)$ be a monotone function. Then

$$\emptyset \subseteq f(\emptyset) \subseteq f^2(\emptyset) \subseteq f^3(\emptyset) \subseteq \cdots$$

Assume that S is a finite set; then the Ascending Chain Condition is satisfied. This means that the chain cannot grow infinitely so there exists n such that $f^n(\emptyset) = f^{n+1}(\emptyset) = \cdots$

$f^n(\emptyset)$ is the least fixed point of f



Correctness of the algorithm

- Initialisation

  $\mathfrak{I}(\star) := \emptyset; \mathfrak{I}(\mathbb{S}) := \emptyset; \mathfrak{I}(\mathbb{P}) := \emptyset;$

  Invariant: $\vec{\mathfrak{I}} \subseteq \mathsf{F}^n(\vec{\emptyset})$ since $\vec{\mathfrak{I}} = \vec{\emptyset}$ is the least element

- Iteration

  `while` $\mathfrak{I}(\ell) \neq \mathsf{F}_\ell(\mathfrak{I}(\star), \mathfrak{I}(\mathbb{S}), \mathfrak{I}(\mathbb{P}))$ for some $\ell \in \{\star, \mathbb{S}, \mathbb{P}\}$

  `do` assume $\vec{\mathfrak{I}}$ is $\vec{\mathfrak{I}}'$ and $\vec{\mathfrak{I}}' \subseteq \mathsf{F}^n(\vec{\emptyset})$

  $\mathfrak{I}(\ell) := \mathsf{F}_\ell(\mathfrak{I}(\star), \mathfrak{I}(\mathbb{S}), \mathfrak{I}(\mathbb{P}))$

  then $\vec{\mathfrak{I}} \subseteq \mathsf{F}(\vec{\mathfrak{I}}') \subseteq \mathsf{F}^{n+1}(\vec{\emptyset}) = \mathsf{F}^n(\vec{\emptyset})$ when $\mathsf{lfp}(\mathsf{F}) = \mathsf{F}^n(\vec{\emptyset})$

If the algorithm terminates then it computes the least fixed point of F.

The algorithm terminates because $\mathfrak{I}(\ell) \subset \mathsf{F}_\ell(\mathfrak{I}(\star), \mathfrak{I}(\mathbb{S}), \mathfrak{I}(\mathbb{P}))$ is only possible finitely many times since $\mathcal{P}(\mathbf{Group} \cup \mathbf{Cap})^3$ satisfies the Ascending Chain Condition.

Secure firewalls

- Analysing software in an unknown environment
  - program analysis traditionally applies to closed systems
- Identify hardest attackers
  - how to turn an open system into a closed system

A model of a firewall

$$(\nu\ w : \omega)\ \boxed{\boxed{\text{out } w.\text{in } k'.\text{in } w}^{\,k} \mid \text{open } k'.\text{open } k''.P}^{\,w}$$

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 <u>not</u> knowing the passwords should <u>not</u> be allowed enter:
  - hard: the firewall is an open system — we cannot inspect all agents
  - an agent that does not know the passwords might learn them
  - the firewall might contain a trap door through which agents might enter

An agent knowing the passwords

$\overset{\text{w}}{\boxed{\overset{\text{k}}{\boxed{\text{out w. in k'. in w}}} \mid \text{open k'. open k''. } P}} \mid \overset{\text{k'}}{\boxed{\text{open k. } \overset{\text{k''}}{\boxed{Q}}}}$

$\longrightarrow \overset{\text{w}}{\boxed{\text{open k'. open k''. } P}} \mid \overset{\text{k}}{\boxed{\text{in k'. in w}}} \mid \overset{\text{k'}}{\boxed{\text{open k. } \overset{\text{k''}}{\boxed{Q}}}}$

$\longrightarrow \overset{\text{w}}{\boxed{\text{open k'. open k''. } P}} \mid \overset{\text{k'}}{\boxed{\overset{\text{k}}{\boxed{\text{in w}}} \mid \text{open k. } \overset{\text{k''}}{\boxed{Q}}}}$

$\longrightarrow \overset{\text{w}}{\boxed{\text{open k'. open k''. } P}} \mid \overset{\text{k'}}{\boxed{\text{in w} \mid \overset{\text{k''}}{\boxed{Q}}}}$

$\longrightarrow \overset{\text{w}}{\boxed{\text{open k'. open k''. } P \mid \overset{\text{k'}}{\boxed{\overset{\text{k''}}{\boxed{Q}}}}}} \longrightarrow \overset{\text{w}}{\boxed{\text{open k''. } P \mid \overset{\text{k''}}{\boxed{Q}}}} \longrightarrow \overset{\text{w}}{\boxed{P \mid Q}}$

Contextual equivalent processes

Agents <u>not</u> knowing the passwords

- an agent that does not initially know the passwords might learn them subsequently:

$$\overset{e}{\boxed{\text{in t. } R}} \;\Big|\; \overset{t}{\boxed{\text{open e. in k'}}} \;\Big|\; \overset{k'}{\boxed{\text{open t} \mid \text{open k.} \overset{k''}{\boxed{Q}}}}$$

$$\Big|\; (\nu\, \text{w} : \omega)\, \overset{w}{\boxed{\overset{k}{\boxed{\text{out w.in k}'\text{.in w}}} \mid \text{open k}'\text{.open k}''.P}}$$

- the firewall might contain a trap door through which agents might enter:

$$\overset{e}{\boxed{\text{in t. } R}} \;\Big|$$

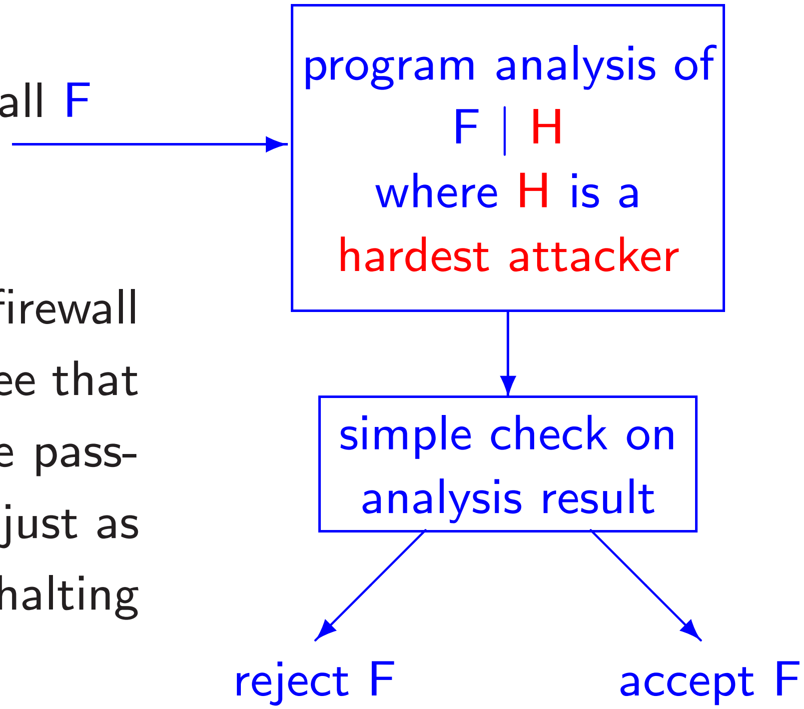
$$(\nu \text{w} : \omega)\, \overset{w}{\boxed{\overset{k}{\boxed{\text{out w.in k}'\text{.in w}}} \mid \text{open k}'\text{.open k}''.P \mid \text{open t} \mid \overset{t}{\boxed{\text{out w. in w. open e}}} \mid \overset{k'}{\boxed{}} \mid \overset{k''}{\boxed{}}}}$$

The core of the problem

- 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

Proposed firewall F



Cubic time analysis in the size of F

F might still be a firewall but we cannot guarantee that agents not knowing the passwords cannot enter — just as we cannot solve the halting problem.

An easy check shows whether an agent knows some of the passwords.

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

Terminology

A proposed firewall is given by

$$(\Gamma,\ (\nu\,\mathsf{w}\colon\omega)\,\mathsf{w}\,[F],\ \kappa)$$

where Γ assigns groups to the finitely many names occurring in $(\nu\,\mathsf{w}\colon\omega)\,\mathsf{w}\,[F]$ 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 mention the name of the firewall nor any of its passwords: $\{\Gamma(n) \mid n \in \mathsf{fn}(U)\} \cap \{\omega, \kappa\} = \emptyset$

OBS: a more refined notion of a proposed firewall may specify the format of a successful agent as $(\Gamma,\ (\nu\,\mathsf{w}\colon\omega)\,\mathsf{w}\,[F],\ A)$ 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

More terminology

- 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 unware attackers can enter the firewall
- Corollary of the subject reduction result:

$$\text{strongly protective} \quad \Rightarrow \quad \text{protective}$$

- 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 $(\Gamma,\ (\nu\, \mathsf{w} : \omega)\, \mathsf{w}\, [F],\ \kappa)$ 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 $(\nu\, \mathsf{w} : \omega)\, \mathsf{w}\, [F] \mid U$ is an element of

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

where $\mathbf{Group} =$ (groups mentioned in Γ) $\cup \{\omega, \kappa, \top\}$ is finite!

The hardest attacker

Key observations:

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

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

Details of the test

For the hardest attacker H:

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

$$\mathcal{I}_H \models^\star_\Gamma ((\nu\, \mathsf{w} : \omega)\, \mathsf{w}\, [F]) \;\mid H$$

  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

$$P = \mathsf{in}\, n \mid \mathsf{out}\, n \mid \mathsf{open}\, n$$

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:

$$\mathcal{I}_U \models^{\star}_{\Gamma} ((\nu\, \mathsf{w} : \omega)\, \mathsf{w}\, [F]) \;\mid U$$

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 iof constraints needed for calculating $\mathcal{I}_H$.)