

Secure Data Flow in a Calculus for Context Awareness

Doina Bucur

joint work with Mogens Nielsen

doina@daimi.au.dk

BRICS, Department of Computer Science
University of Aarhus, Denmark



UbiComp Applications

UbiComp:

- ★ **Implosion** in size: smaller (embedded) computers
- ★ **Diversification**: dedicated computers
- ★ **Explosion** in user interfaces, interaction, numbers and mobility
- ★ **Decentralization**: distribution of services

UbiComp Applications

UbiComp:

- ★ **Implosion** in size: smaller (embedded) computers
- ★ **Diversification**: dedicated computers
- ★ **Explosion** in user interfaces, interaction, numbers and mobility
- ★ **Decentralization**: distribution of services

	distributed system		
	traditional	nomadic	ad hoc
infrastructure	fixed	fixed	?
device	fixed	mobile	mobile
services	in the core	in the core	decentralized
connection	permanent	permanent	intermittent
communication	synchronous	asynchronous	asynchronous
context	static	dynamic	dynamic
middleware	transparency	transparent	awareness

UbiComp Applications

UbiComp:

- ★ **Implosion** in size: smaller (embedded) computers
- ★ **Diversification**: dedicated computers
- ★ **Explosion** in user interfaces, interaction, numbers and mobility
- ★ **Decentralization**: distribution of services

Two-tier architecture:

- ★ **Applications**: low local resources, networked, adaptive, cooperative, numerous, mobile
- ★ **Communication** protocols (of context): synchronous or asynchronous

Context Awareness Challenges

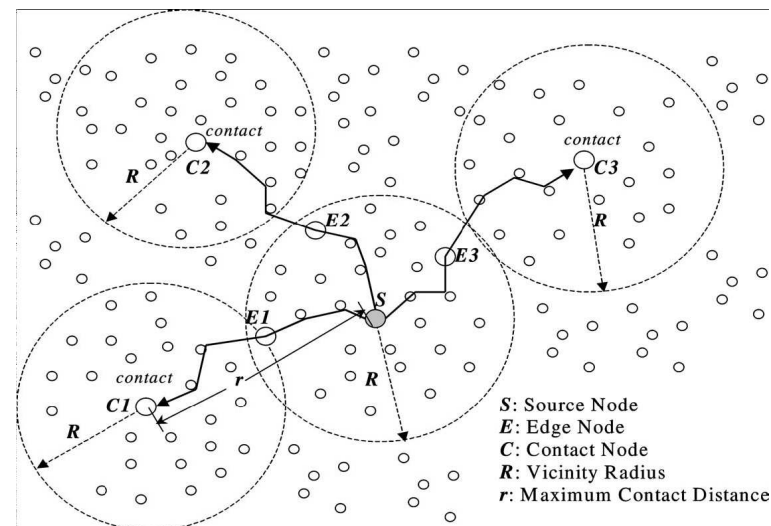
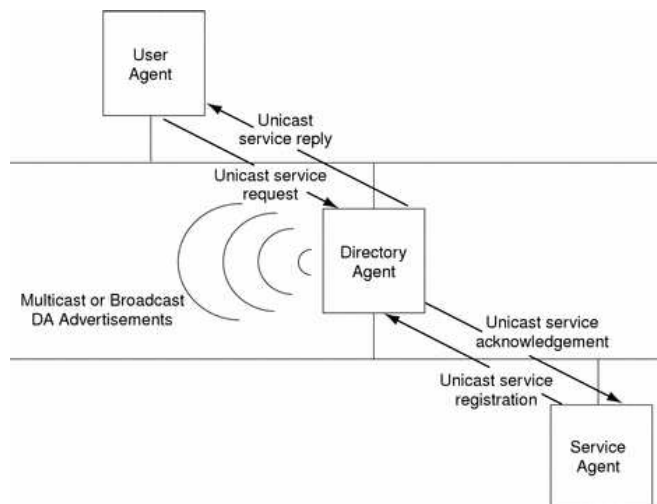
Context awareness:

- ★ Acquiring contextual information:
 - ✓ [Dey, Abowd] primary, secondary
 - ✓ [Pascoe, Dey] sensing
 - ✓ localization, activity recognition

Context Awareness Challenges

Context awareness:

- ★ Acquiring contextual information:
 - ✓ [Dey, Abowd] primary, secondary
 - ✓ [Pascoe, Dey] sensing
 - ✓ localization, activity recognition
- ★ Designing infrastructure and protocols
 - ✓ [Pascoe, Dey] provision and discovery
 - ✓ service discovery, protocol verification



Context Awareness Challenges

Context awareness:

- ★ Acquiring contextual information:
 - ✓ [Dey, Abowd] primary, secondary
 - ✓ [Pascoe, Dey] sensing
 - ✓ localization, activity recognition
- ★ Designing infrastructure and protocols
 - ✓ [Pascoe, Dey] provision and discovery
 - ✓ service discovery, protocol verification
- ★ Dealing with accuracy/uncertainty
 - ✓ sensor fusion

Context Awareness Challenges

Context awareness:

- ★ Acquiring contextual information:
 - ✓ [Dey, Abowd] primary, secondary
 - ✓ [Pascoe, Dey] sensing
 - ✓ localization, activity recognition
- ★ Designing infrastructure and protocols
 - ✓ [Pascoe, Dey] provision and discovery
 - ✓ service discovery, protocol verification
- ★ Dealing with accuracy/uncertainty
 - ✓ sensor fusion
- ★ Achieving security/privacy, correctness
 - ✓ information flow (security), effect-based type systems

Application Features

[Schilit] Awareness:

- ★ passive (manual)
- ★ active (automatic)

Application Features

[Schilit] Awareness:

- ★ passive (manual)
- ★ active (automatic)

Features of context-aware applications:

	information	commands
manual	proximate selection	automatic contextual reconfiguration
automatic	contextual information and commands	context-triggered actions

Process calculi

Universal models of computation. Formally model concurrent systems:

★ Process terms

process	P
parallel composition	$P \mid Q$
replication	$!P$
name restriction	$(\nu z)P$

Process calculi

Universal models of computation. Formally model concurrent systems:

★ Process terms

process	P
parallel composition	$P \mid Q$
replication	$!P$
name restriction	$(\nu z)P$

★ Interaction and synchronization

asynchronous	$\langle M \rangle \mid (x).P \rightarrow \{^M / x\}P$
synchronous	$\bar{c}v.P \mid c(x).Q \rightarrow P \mid \{^v / x\}Q$

Process calculi

Universal models of computation. Formally model concurrent systems:

★ Process terms

process	P
parallel composition	$P \mid Q$
replication	$!P$
name restriction	$(\nu z)P$

★ Interaction and synchronization

asynchronous	$\langle M \rangle \mid (x).P \rightarrow \{^M / x\}P$
synchronous	$\bar{c}v.P \mid c(x).Q \rightarrow P \mid \{^v / x\}Q$

★ Mobility and topology

hierarchical cells	$a[P \mid b[Q]] \mid c[in a.R]$
dynamic process graphs	$\nu c \nu d (\bar{c}d.P \mid c(x).\bar{x}5.Q \mid d(y).R)$

Process calculi

Universal models of computation. Formally model concurrent systems:

- ★ Process terms

process	P
parallel composition	$P \mid Q$
replication	$!P$
name restriction	$(\nu z)P$

- ★ Interaction and synchronization

asynchronous	$\langle M \rangle \mid (x).P \rightarrow \{^M/x\}P$
synchronous	$\bar{c}v.P \mid c(x).Q \rightarrow P \mid \{^v/x\}Q$

- ★ Mobility and topology

hierarchical cells	$a[P \mid b[Q]] \mid c[in\ a.R]$
dynamic process graphs	$\nu c \nu d (\bar{c}d.P \mid c(x).\bar{x}5.Q \mid d(y).R)$

- ★ Algebraic laws to manipulate and analyze processes;
techniques to reason on process equivalence

A Calculus of Context Awareness

Process: first-class object

- ★ returnable from a function
- ★ transmissible among distributed processes
- ★ storable outside running processes

Named macros [Zimmer]:

$$f \triangleright P, \text{ with } f \in \mathcal{F}$$

A Calculus of Context Awareness

Process: first-class object

- ★ returnable from a function
- ★ transmissible among distributed processes
- ★ storable outside running processes

Named macros [Zimmer]:

$$f \triangleright P, \text{ with } f \in \mathcal{F}$$

Macro definition flavours:

- ★ **one-shot**, consumed at call (e.g. network packets)

$$f \triangleright P$$

- ★ **permanent**, infinitely called (e.g. a person's location)

$$!f \triangleright P$$

Examples.

Infrastructure-based setting

Standard:

Processes	P	$::=$	0	no process
			$P \mid P'$	parallel composition
			$a[P]$	mobile entity, $a \in \mathcal{A}$
			$\nu z P$	name restriction, $z \in \mathcal{F} \cup \mathcal{A}$
			$in\ a.P$	movement in
			$out.P$	movement out

Infrastructure-based setting

Standard:

Processes	$P ::= 0$	no process
	$ P \mid P'$	parallel composition
	$ a[P]$	mobile entity, $a \in \mathcal{A}$
	$ \nu z P$	name restriction, $z \in \mathcal{F} \cup \mathcal{A}$
	$ in a.P$	movement in
	$ out.P$	movement out

$$\text{IN} \quad a[P] \mid b[in a.Q \mid R] \longrightarrow a[P \mid b[Q \mid R]]$$

$$\text{OUT} \quad a[P \mid b[out.Q \mid R]] \longrightarrow a[P] \mid b[Q \mid R]$$

$$\text{STRUCT} \quad \frac{P \equiv P' \quad P \longrightarrow Q \quad Q \equiv Q'}{P' \longrightarrow Q'}$$

$$\text{CONTEXT} \quad \frac{P \longrightarrow Q}{\mathbf{C}[P] \longrightarrow \mathbf{C}[Q]}$$

Infrastructure-based setting

Services: naturally layered.

Processes	P	$::=$	f^a	macro call, $f \in \mathcal{F}$
			$def^a D \text{ in } P$	macro definition
			$E P$	public definitions
Definitions	E	$::=$	$(D)^a$	floating definition
	D	$::=$	$F \mid !F$	
	F	$::=$	$f \triangleright P$	macro definition

Infrastructure-based setting

Services: naturally layered.

Processes	P	$::=$	f^a	macro call, $f \in \mathcal{F}$
			$def^a D \text{ in } P$	macro definition
			$E P$	public definitions
Definitions	E	$::=$	$(D)^a$	floating definition
	D	$::=$	$F \mid !F$	
	F	$::=$	$f \triangleright P$	macro definition

$$\text{DEF } def^a f \triangleright Q \text{ in } P \longrightarrow (f \triangleright Q)^a P$$

$$\text{CALL } (f \triangleright Q)^a f^a \longrightarrow Q$$

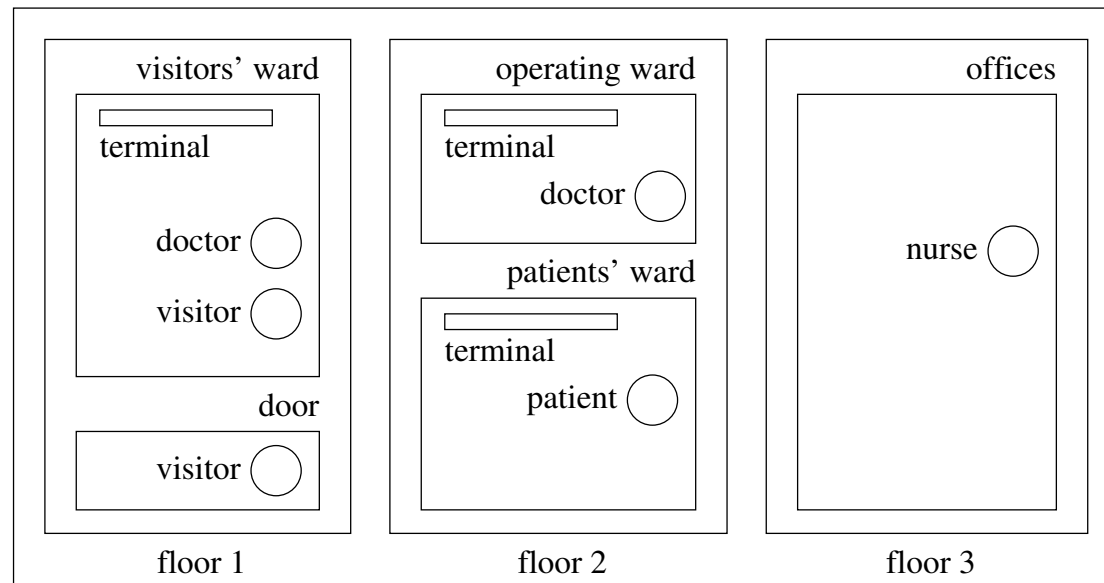
$$\text{UP, DOWN } (D)^b P \mid Q \equiv (D)^b (P \mid Q) \quad a[(D)^b P] \equiv (D)^b a[P]$$

A model of **contextual commands**, **context-triggered actions**, communication.

Example

ABC (Activity-Based Computing, [Bardram, Hansen 2004]). Deployed in a major Danish hospital.

hospital network infrastructure



Verification: distributed firewalls

Effects and policies, $a_G^\tau[P]$:

Entry policy	G	\subseteq	\mathcal{A}
Effects	\mathcal{E}	$=$	$\bigcup_{f \in \mathcal{F}} \{def(f), call(f)\}$
Security policy	τ	$=$	$\mathcal{A} \rightarrow (\mathcal{A} \rightarrow \mathcal{P}(\mathcal{E}))$

Verification: distributed firewalls

Effects and policies, $a_G^\tau[P]$:

Entry policy	$G \subseteq \mathcal{A}$
Effects	$\mathcal{E} = \bigcup_{f \in \mathcal{F}} \{def(f), call(f)\}$
Security policy	$\tau = \mathcal{A} \rightarrow (\mathcal{A} \rightarrow \mathcal{P}(\mathcal{E}))$

Write $a_G^\tau[P]$ if P should satisfy policies τ and G :

- ★ if $\tau(b)(c) \not\supseteq def(f)$, then P should not be $\equiv \mathbf{C}[b[def^c f \triangleright Q \text{ in } R \mid S]]$;
similarly for macro calls;
- ★ if $b \notin G$, then P should not be $\equiv \mathbf{C}[b[Q]]$.

Verification: distributed firewalls

Effects and policies, $a_G^\tau[P]$:

Entry policy	$G \subseteq \mathcal{A}$
Effects	$\mathcal{E} = \bigcup_{f \in \mathcal{F}} \{def(f), call(f)\}$
Security policy	$\tau = \mathcal{A} \rightarrow (\mathcal{A} \rightarrow \mathcal{P}(\mathcal{E}))$

Write $a_G^\tau[P]$ if P should satisfy policies τ and G :

- ★ if $\tau(b)(c) \not\supseteq def(f)$, then P should not be $\equiv \mathbf{C}[b[def^c f \triangleright Q \text{ in } R \mid S]]$;
similarly for macro calls;
- ★ if $b \notin G$, then P should not be $\equiv \mathbf{C}[b[Q]]$.

Fully-permissive policies: \mathcal{A}, ω .

Verification: distributed firewalls

Effects and policies, $a_G^\tau[P]$:

Entry policy	$G \subseteq \mathcal{A}$
Effects	$\mathcal{E} = \bigcup_{f \in \mathcal{F}} \{def(f), call(f)\}$
Security policy	$\tau = \mathcal{A} \rightarrow (\mathcal{A} \rightarrow \mathcal{P}(\mathcal{E}))$

Write $a_G^\tau[P]$ if P should satisfy policies τ and G :

- ★ if $\tau(b)(c) \not\supseteq def(f)$, then P should not be $\equiv \mathbf{C}[b[def^c f \triangleright Q \text{ in } R \mid S]]$;
similarly for macro calls;
- ★ if $b \notin G$, then P should not be $\equiv \mathbf{C}[b[Q]]$.

Fully-permissive policies: \mathcal{A}, ω .

Policy composition:

$$G \cap H$$

$$(\tau \circ \sigma)(b)(c) = \tau(b)(c) \cap \sigma(b)(c).$$

Type check: active code

$$\text{NULL } a_G^\tau \vdash 0 \quad \text{PAR } \frac{a_G^\tau \vdash P \quad a_G^\tau \vdash P'}{a_G^\tau \vdash P|P'}$$

Type check: active code

$$\text{NULL} \quad a_G^\tau \vdash 0$$

$$\text{PAR} \quad \frac{a_G^\tau \vdash P \quad a_G^\tau \vdash P'}{a_G^\tau \vdash P|P'}$$

$$\text{CALL} \quad \frac{\tau(a)(b) \ni \text{call}(f)}{a_G^\tau \vdash f^b}$$

$$\text{DEF} \quad \frac{\tau(a)(b) \ni \text{def}(f) \quad a_G^\tau \vdash P}{a_G^\tau \vdash \text{def}^b ? f \triangleright Q \text{ in } P}$$

Type check: active code

$$\text{NULL} \quad a_G^\tau \vdash 0$$

$$\text{PAR} \quad \frac{a_G^\tau \vdash P \quad a_G^\tau \vdash P'}{a_G^\tau \vdash P|P'}$$

$$\text{CALL} \quad \frac{\tau(a)(b) \ni \text{call}(f)}{a_G^\tau \vdash f^b}$$

$$\text{DEF} \quad \frac{\tau(a)(b) \ni \text{def}(f) \quad a_G^\tau \vdash P}{a_G^\tau \vdash \text{def}^b ? f \triangleright Q \text{ in } P}$$

$$\text{AMB} \quad \frac{b \in G \quad b_{G \cap H}^{\tau \circ \sigma} \vdash P}{a_G^\tau \vdash b_H^\sigma [P]}$$

Type check: active code

$$\begin{array}{c}
 \text{NULL} \quad a_G^\tau \vdash 0 \qquad \text{PAR} \quad \frac{a_G^\tau \vdash P \quad a_G^\tau \vdash P'}{a_G^\tau \vdash P|P'} \\
 \\
 \text{CALL} \quad \frac{\tau(a)(b) \ni \text{call}(f)}{a_G^\tau \vdash f^b} \qquad \text{DEF} \quad \frac{\tau(a)(b) \ni \text{def}(f) \quad a_G^\tau \vdash P}{a_G^\tau \vdash \text{def}^b ?f \triangleright Q \text{ in } P} \\
 \\
 \text{AMB} \quad \frac{b \in G \quad b_{G \cap H}^{\tau \circ \sigma} \vdash P}{a_G^\tau \vdash b_H^\sigma [P]} \\
 \\
 \text{MSG} \quad \frac{a_G^\tau \vdash P}{a_G^\tau \vdash (?f \triangleright Q)^{b, \sigma} P} \qquad \text{RES} \quad \frac{a_G^\tau \vdash P}{a_G^\tau \vdash \nu z P} \quad z \notin \text{fn}(a_G^\tau)
 \end{array}$$

Type check: active code

$$\begin{array}{c}
 \text{NULL} \quad a_G^\tau \vdash 0 \qquad \text{PAR} \quad \frac{a_G^\tau \vdash P \quad a_G^\tau \vdash P'}{a_G^\tau \vdash P|P'} \\
 \\
 \text{CALL} \quad \frac{\tau(a)(b) \ni \text{call}(f)}{a_G^\tau \vdash f^b} \qquad \text{DEF} \quad \frac{\tau(a)(b) \ni \text{def}(f) \quad a_G^\tau \vdash P}{a_G^\tau \vdash \text{def}^b ?f \triangleright Q \text{ in } P} \\
 \\
 \text{AMB} \quad \frac{b \in G \quad b_{G \cap H}^{\tau \circ \sigma} \vdash P}{a_G^\tau \vdash b_H^\sigma [P]} \\
 \\
 \text{MSG} \quad \frac{a_G^\tau \vdash P}{a_G^\tau \vdash (?f \triangleright Q)^{b, \sigma ?} P} \qquad \text{RES} \quad \frac{a_G^\tau \vdash P}{a_G^\tau \vdash \nu z P} \quad z \notin \text{fn}(a_G^\tau) \\
 \\
 \text{IN} \quad \frac{a_G^\tau \vdash P}{a_G^\tau \vdash \text{in } b.P} \qquad \text{OUT} \quad \frac{a_G^\tau \vdash P}{a_G^\tau \vdash \text{out}.P}
 \end{array}$$

Breaking scopes: the context function

So far, no checks on definitions.

\mathcal{C} maximizes the policies a definition body should comply with when moving up:

$$\mathcal{C} : \mathcal{A} \longrightarrow \frac{\mathcal{T}}{\mathcal{P}(\mathcal{A})}.$$

Example: given $def^b f \triangleright Q$ in P , its $\mathcal{C}(b)$ returns the collected policies of ambients above b .

Fully-permissive instantiation Ω :

$$\Omega(\mathcal{A}) = \omega_{\mathcal{A}}.$$

Notation abuse: $\mathcal{C}(H) = \tau_G$.

An instantiation: by writing updates upon Ω : $\Omega [H \rightarrow \tau_G]$.

Type check: inactive code

$$\text{DEF} \quad \frac{\mathcal{C}, a_G^\tau \vdash P \quad a_G^\tau[Q] \text{ well-typed}}{\mathcal{C}, a_G^\tau \vdash \text{def}^a ? f \triangleright Q \text{ in } P} \quad \frac{\mathcal{C}, a_G^\tau \vdash P \quad b\mathcal{C}(b)[Q] \text{ well-typed}}{\mathcal{C}, a_G^\tau \vdash \text{def}^b ? f \triangleright Q \text{ in } P}$$

Type check: inactive code

$$\text{DEF} \quad \frac{\mathcal{C}, a_G^\tau \vdash P \quad a_G^\tau[Q] \text{ well-typed}}{\mathcal{C}, a_G^\tau \vdash \text{def}^a ? f \triangleright Q \text{ in } P} \quad \frac{\mathcal{C}, a_G^\tau \vdash P \quad b\mathcal{C}(b)[Q] \text{ well-typed}}{\mathcal{C}, a_G^\tau \vdash \text{def}^b ? f \triangleright Q \text{ in } P}$$

$$\text{AMB} \quad \frac{\mathcal{C} [\{a\} \cup G \rightarrow \tau_G], b_{G \cap H}^{\tau \circ \sigma} \vdash P}{\mathcal{C}, a_G^\tau \vdash b_H^\sigma[P]}$$

Type check: inactive code

$$\text{DEF} \quad \frac{\mathcal{C}, a_G^\tau \vdash P \quad a_G^\tau[Q] \text{ well-typed}}{\mathcal{C}, a_G^\tau \vdash \text{def}^a ?f \triangleright Q \text{ in } P} \quad \frac{\mathcal{C}, a_G^\tau \vdash P \quad b\mathcal{C}(b)[Q] \text{ well-typed}}{\mathcal{C}, a_G^\tau \vdash \text{def}^b ?f \triangleright Q \text{ in } P}$$

$$\text{AMB} \quad \frac{\mathcal{C} [\{a\} \cup G \rightarrow \tau_G], b_{G \cap H}^{\tau \circ \sigma} \vdash P}{\mathcal{C}, a_G^\tau \vdash b_H^\sigma[P]}$$

Not interesting:

$$\text{NULL} \quad \mathcal{C}, a_G^\tau \vdash 0 \quad \text{PAR} \quad \frac{\mathcal{C}, a_G^\tau \vdash P \quad \mathcal{C}, a_G^\tau \vdash P'}{\mathcal{C}, a_G^\tau \vdash P|P'} \quad \text{CALL} \quad \mathcal{C}, a_G^\tau \vdash f^b$$

$$\text{MSG} \quad \frac{\mathcal{C}, a_G^\tau \vdash P}{\mathcal{C}, a_G^\tau \vdash (?f \triangleright Q)^{b, \sigma} P} \quad \text{RES} \quad \frac{\mathcal{C}, a_G^\tau \vdash P}{\mathcal{C}, a_G^\tau \vdash \nu z P} \quad z \notin \text{fn}(\mathcal{C}) \cup \text{fn}(a_G^\tau)$$

$$\text{IN} \quad \frac{\mathcal{C}, a_G^\tau \vdash P}{\mathcal{C}, a_G^\tau \vdash \text{in } b.P} \quad \text{OUT} \quad \frac{\mathcal{C}, a_G^\tau \vdash P}{\mathcal{C}, a_G^\tau \vdash \text{out}.P}$$

Well-typedness

System $a_G^\tau[P]$ well-typed if

$$a_G^\tau \vdash P \text{ and } \Omega, a_G^\tau \vdash \circ P$$

Well-typedness

System $a_G^\tau[P]$ well-typed if

$$a_G^\tau \vdash P \text{ and } \Omega, a_G^\tau \vdash\circ P$$

System P without a root ambient well-typed if

$$world_A^\omega \vdash P \text{ and } \Omega, world_A^\omega \vdash\circ P$$

Typed semantics: dynamic checks at moves

Safe:

$$\text{UP} \quad a_G^\tau[(D)^b P] \longrightarrow (D)^b a_G^\tau[P] \quad a_G^\tau[(D)^a P] \longrightarrow a_G^\tau[(D)^{a,\tau} P]$$

$$\text{OUT} \quad a_G^\tau[P \mid b_H^\sigma[out.Q \mid R]] \longrightarrow a_G^\tau[P] \mid b_H^\sigma[Q \mid R]$$

Typed semantics: dynamic checks at moves

Safe:

$$\text{UP} \quad a_G^\tau[(D)^b P] \longrightarrow (D)^b a_G^\tau[P] \quad a_G^\tau[(D)^a P] \longrightarrow a_G^\tau[(D)^{a,\tau} P]$$

$$\text{OUT} \quad a_G^\tau[P \mid b_H^\sigma[out.Q \mid R]] \longrightarrow a_G^\tau[P] \mid b_H^\sigma[Q \mid R]$$

Check:

$$\text{DOWN} \quad \frac{a_G^{\sigma \circ \tau}[Q] \text{ well-typed}}{(f \triangleright Q)^{b,\sigma} a_G^\tau[P] \longrightarrow a_G^\tau[(f \triangleright Q)^{b,\sigma \circ \tau} P]}$$

$$\text{IN} \quad \frac{b \in G \quad b_G^\tau \vdash Q \mid R \quad \Omega[\{a\} \cup G \rightarrow \tau_G], b_G^\tau \vdash \circ Q \mid R}{a_G^\tau[P] \mid b_H^\sigma[in a.Q \mid R] \longrightarrow a_G^\tau[P \mid b_H^\sigma[Q \mid R]]}$$

Typed semantics: dynamic checks at moves

Safe:

$$\text{UP} \quad a_G^\tau[(D)^b P] \longrightarrow (D)^b a_G^\tau[P] \quad a_G^\tau[(D)^a P] \longrightarrow a_G^\tau[(D)^{a,\tau} P]$$

$$\text{OUT} \quad a_G^\tau[P \mid b_H^\sigma[out.Q \mid R]] \longrightarrow a_G^\tau[P] \mid b_H^\sigma[Q \mid R]$$

Check:

$$\text{DOWN} \quad \frac{a_G^{\sigma \circ \tau}[Q] \text{ well-typed}}{(f \triangleright Q)^{b,\sigma} a_G^\tau[P] \longrightarrow a_G^\tau[(f \triangleright Q)^{b,\sigma \circ \tau} P]}$$

$$\text{IN} \quad \frac{b \in G \quad b_G^\tau \vdash Q \mid R \quad \Omega[\{a\} \cup G \rightarrow \tau_G], b_G^\tau \vdash \circ Q \mid R}{a_G^\tau[P] \mid b_H^\sigma[in a.Q \mid R] \longrightarrow a_G^\tau[P \mid b_H^\sigma[Q \mid R]]}$$

Inherently safe:

$$\text{DEF} \quad def^a D \text{ in } P \longrightarrow (D)^a P \quad \text{CALL} \quad (f \triangleright P)^{a,\tau} f^a \longrightarrow P$$

$$\text{STRUCT} \quad \frac{P \equiv P' \quad P \longrightarrow Q \quad Q \equiv Q'}{P' \longrightarrow Q'}$$

$$\text{CONTEXT} \quad \frac{P \longrightarrow Q}{\mathbf{C}[P] \longrightarrow \mathbf{C}[Q]}$$

Subject Reduction

If P is well-typed and $P \xrightarrow{*} Q$, then Q is well-typed.

Errors and Soundness

$$\text{ERR_DEF} \quad \frac{(\sigma \circ \tau)(a)(b) \not\equiv \text{def}(f)}{\mathbf{C}_H^\sigma [a_G^\tau [\text{def}^b f \triangleright Q \text{ in } P \mid R]] \longrightarrow \text{err}}$$

$$\text{ERR_CALL} \quad \frac{(\sigma \circ \tau)(a)(b) \not\equiv \text{call}(f)}{\mathbf{C}_H^\sigma [a_G^\tau [f^b \mid P]] \longrightarrow \text{err}}$$

$$\text{ERR_IN} \quad \frac{H \not\equiv a}{\mathbf{C}_H^\sigma [a_G^\tau [P]] \longrightarrow \text{err}}$$

$$\text{ERR_STR} \quad \frac{P \equiv P' \quad P' \longrightarrow \text{err}}{P \longrightarrow \text{err}}$$

Soundness: If P is well-typed then $P \not\overset{*}{\rightarrow} \text{err}$.

The End. Questions?

Currently: ad-hoc setting

Networks	$N ::= 0$	no network
	$ N \mid N'$	parallel composition
	$ a_l[P]$	node at l
Processes	$P ::= 0$	no process
	$ P \mid P'$	parallel composition
	$ \text{move } l.P$	movement

$$\text{MOVE} \frac{(l, l') \in G_{mob}^a}{a_l[\text{move } l'.P \mid Q] \longrightarrow a_{l'}[P \mid Q]}$$

Ad-hoc setting: proactive

Communication paradigm: **broadcast**.

Processes	P	$::=$	f	local macro call
			$\text{def}^{\odot n} D \text{ in } P$	proactive macro definition
			$E P$	public definitions
Definitions	E	$::=$	0	no definition
			$(D)^{\odot n}$	floating definition, $(D)^{\odot 0} \equiv 0$

Ad-hoc setting: proactive

Communication paradigm: **broadcast**.

Processes	P	$::=$	f	local macro call
			$\text{def}^{\odot n} D \text{ in } P$	proactive macro definition
			$E P$	public definitions
Definitions	E	$::=$	0	no definition
			$(D)^{\odot n}$	floating definition, $(D)^{\odot 0} \equiv 0$

DEF $\text{def}^{\odot n} D \text{ in } P \longrightarrow (D)^{\odot n} P$

DEFBCAST
$$\frac{n > 0 \quad (l, l') \in G_{com}^a}{a_l[(D)^{\odot n} P] \mid \prod_{b, l'} b_{l'}[Q] \longrightarrow a_l[P] \mid \prod_{b, l'} b_{l'}[(D)(D)^{\odot n-1} Q]}$$

CALL $(f \triangleright Q)f \longrightarrow Q$

Ad-hoc setting: reactive

Processes	$P ::= f^{\odot n}$	reactive macro call, $n \in \mathbb{N}$
	$\text{def } D \text{ in } P$	local macro definition
	$E P$	public definitions
Definitions	$E ::= 0$	no definition
	(D)	locally floating definition

Ad-hoc setting: reactive

Processes $P ::= f^{\odot n}$ reactive macro call, $n \in \mathbb{N}$
 $| \text{def } D \text{ in } P$ local macro definition
 $| E P$ public definitions

Definitions $E ::= 0$ no definition
 $| (D)$ locally floating definition

DEF $\text{def } D \text{ in } P \longrightarrow (D)P$

CALLBCAST

$$\frac{n > 0 \quad (l, l') \in G_{com}^a}{a_l[f^{\odot n} \mid P] \mid \prod_{b, l'} b_{l'}[Q] \longrightarrow a_l[P] \mid \prod_{b, l'} b_{l'}[f^{\odot n-1} \mid Q]}$$

$$(f \triangleright Q) f^{\odot n} \longrightarrow (f \triangleright Q)^{\odot n}$$

CALL $(f \triangleright Q) f \longrightarrow Q$