Informatics Education: Hopes or Hypes OR Ubiquitous computing will empower us, ONLY if we understand it!

Hussein Zedan

August 24, 2011

Informatics - The discipline I

- Informatics is a multi-disciplinary activity that requires the talent and expertise of various disciplines in a rather unified and integrated fashion: Computational Science, Engineering, Psychology, Sociology and Domain Experts (e.g., Biologist, Chemist, Physicist, Mathematician, Space Scientist Civil Engineer, Archaeologist, etc.).

Informatics - The discipline II

- *Computer Science* fundamentally concentrates on the theory and the basics/fundamentals of computations.
- Basics titles include Programming Paradigms, Specification, Verification, Logics and Semantics and systems and applications, such as Operating Systems, Compilers, Databases, Networking, Communication Protocols .
- The spectrum varies from the highly theoretical to the highly practical.

Computer Science is not Informatics

Informatics - The discipline III

- *Software Engineering* deals with the process and the engineering of building the artifact.
- This involves also standards .
- We find topics such as, Testing, Requirements, Validation, Certification, Maintenance, Management, Risk, etc.

Software Engineering is not Informatics

Informatics - The discipline IV

- *Information Systems* focuses on the totality and holistic nature of the artifact: what it is, who it is for and how .
- We find topics such as, Requirements, Methodologies, Analysis, etc.

Information Systems is not Informatics

Informatics - The discipline V

- Unlike those disciplines, Informatics is all of the above but in addition, domains play fundamental and crucial part in its treatment.
- We can see that in the proliferation of the *X-Informatics* subjects: for example, Bio-informatics, Health-informatics, Space-informatics, etc.
- Even those *X-Informatics* subjects, only utilises electronic computation techniques to solve their problems.
- However, domains bring extra models which need to be integrated with that of the electronic computation.

Informatics and Ubiquity - I

- Populations of computing entities will be a significant part of our environment, performing tasks that support us, and we shall be largely unaware of them. (Mark Weiser, 1994)
- In the next five to ten years the computer will be erased from our consciousness. We will simply not talk about it any longer, we will not read about it, apart from experts of course. (Joseph Weizenbaum (2001)
- **Ubiquitous (Informatics) computing will empower us, if we understand it.** (Robin Milner (2008))

Informatics and Ubiquity - II

- Informatics is the study of the structure, the behaviour, and the interactions of natural and engineered computational systems. (INFO, Edinbrugh)
- The central focus of Informatics is the transformation of information - whether by electronic computation or communication, whether by organisms or artefacts. Understanding informational phenomena - such as computation, cognition, and communication - enables technological advances.
- **Informatic, as a discipline, should be treated as Ubiquitous Computing**

Qualities of an informatics system

What is new about an informatics system?

- It will continually make decisions hitherto made by us
- It will be vast, maybe 100 times todays systems
- It must continually adapt, on-line, to new requirements
- Individual system will interact with one another

Can traditional software engineering cope?

An Informatics Model I

- Entities in a model *explain*, or are *realised by*, entities in the physical world (as in natural science).
- Just as we say that Newtons laws explain the movement of bodies with mass, so we can say in informatics that a model consisting of programs and their meaning explains the reality of computers and what their screens display.
- What distinguishes the *science of informatics* is that its artifacts demand explanation at many levels.

ENTITIES: PROGRAMS $\overset{\textit{realised by}}{\longrightarrow}$ COMPUTERS

An informatics Model II

- Entities and Behaviour in a model *explain*, or are *realised by*, entities in the physical worldas in natural science.

ENTITIES:

PROGRAMS $\overset{realised\ by}{\rightarrow}$ COMPUTERS

BEHAVIOUR:

actions on memory, i/o $\overset{realised\ by}{\rightarrow}$ keyboard & screen events

Layered informatics models with behaviour I

- Entities and Behaviour in a model *explain*, or are *realised by*, entities in the physical worldas in natural science, or in a lower model .

ENTITIES:

LOGICAL FORMULAS $\overset{specify}{\rightarrow}$ PROGRAMS $\rightarrow$ ASSEBLY CODE $\rightarrow$ HARDWARE DESIGN $\overset{realised\ by}{\rightarrow}$ COMPUTERS

BEHAVIOUR:

valuation set & predicates $\rightarrow$ action on memory, i/o $\rightarrow$ action on memory, i/o $\rightarrow$ voltage, bitmaps, switches $\rightarrow$ keyboard & screen events

Combining models I

- Real systems combine interacting sub-systems; we must also combine partial models.
- Thus, combine models of the electro-mechanical and informatic parts of an aircraft:

$$\begin{array}{ccc} \text{ELECTRO-MECH DESIGN} & \overset{\textit{realised by}}{\longrightarrow} & \text{PHYSICAL AIRCRAFT} \\ \| & & \\ \text{EMBEDDED SOFTWARE} & \overset{\textit{realised by}}{\longrightarrow} & \text{EMBEDDED COMPUTERS} \end{array}$$

Combining models II

- Real systems combine interacting sub-systems; we must also combine partial models. Also, combine models of artifactual and natural systems .

$$\begin{array}{ccc} \text{METEO-MODEL} & \overset{explains}{\longrightarrow} & \text{WEATHER} \\ \| & & \\ \text{ELECTRO-MECH DESIGN} & \overset{realised\ by}{\longrightarrow} & \text{PHYSICAL AIRCRAFT} \\ \| & & \\ \text{EMBEDDED SOFTWARE} & \overset{realised\ by}{\longrightarrow} & \text{EMBEDDED COMPUTERS} \end{array}$$

Combining models II

- For a program, we may combine different explanatory models .
- INRIA did this for the **Airbus** using abstract interpretation, following successful analysis of the failure of the Ariane-5 rocket:

METEO-MODEL $\overset{explains}{\longrightarrow}$ WEATHER

$\|$

ELECTRO-MECH DESIGN $\overset{realised\ by}{\longrightarrow}$ PHYSICAL AIRCRAFT

$\|$

A_1 $\overset{explains}{\longrightarrow}$ EMBEDDED SOFTWARE $\overset{realised\ by}{\longrightarrow}$ EMBEDDED COMPUTERS

Models and their pyramids

- A model consists of some entities, and their behaviour.
  - EXAMPLE: flowcharts, and how to execute them.
- A pyramid of models is built by *explanation* and *combination* :
- Model *A* explains model *B* if
  - *A abstracts* from or *specifies B*, or if
  - B *implements* or *realises* A.
  - EXAMPLE: a specification logic specifies programs.
- Model C combines models A and B if
  - its entities and behaviours combine those of A and B.
  - EXAMPLE: combine distributed programs with a network model.

How do we validate an explanation?

- Natural science:
- Explanation of reality by a model can only be supported by observation. Complete validation impossible (Karl Popper).

- Informatics at lowest level:
- Similar (e,g. realisation of circuit diagrams by a computer).

- Informatics at higher levels:
- Higher levels abound in the model pyramid. Can aspire to complete validation between precise models.

- **PROPOSITION:** Informatics is a science just to the extent that it aspires to complete validation.

Scientific status of the Pyramid of Models

- Useful models, and validations, may well be informal
- Different models suit different people, including non-experts

- Many instances of models and validations exist
- Can we derive languages from models, not vice-versa?

Concepts for Informatics

Each informatics domain, hence each model, will involve several concepts. Here are a few:

locality security authenticity compilation intentions reflectivity specification beliefs encapsulation delegation provenance obligations data-protection continuous time role policy authorisation verification connectivity continuous space simulation mobility failure self-management negotiation trust stochastics

locality security authenticity compilation intentions reflectivity
specification beliefs encapsulation delegation provenance
obligations data-protection continuous time role policy
authorisation verification connectivity continuous space
simulation mobility failure self-management negotiation trust
stochastics

Topics and Laboratories for Informatics

- Mixed models
- Model Integration and/or linkage
- Engineering aspects
- Verification and Validation
- Language design and cross compilation
- Labs include: civil eng, space science, bio-medicine, health, etc
-

A Caculus of Context-aware Ambients

What is CCA?

- CCA stands for Calculus of Context-aware Ambients.
- It is a process calculus for modelling and reasoning about the behaviours of context-aware and mobile concurrent systems.
- A process calculus is a formalism for the high-level description of interactions, communications, and synchronizations between a collection of independent processes.
- CCA is developed in an effort of providing a suitable formalism for modelling interactions and context-awareness in pervasive systems.

Why CCA?

- To provide a process calculus where **mobility** and **context-awareness** are first-class citizens.
- To provide a formal notation for high-level description of mobile and context-aware concurrent systems.
- To enable formal reasoning about the properties of context-aware systems.
- To help understanding the behaviour of mobile and context-aware concurrent systems through simulation/animation.

Other Related Process Calculi

- π-calculus (Milner et al., 1992)
  - Milner(1992). A calculus of mobile processes. Information and Computation 100 (100): 1–40
  - http://en.wikipedia.org/wiki/Pi-calculus
- Ambient Calculus (Cardelli & Gordon, 2000)
  - Cardelli, L., Gordon, A. (2000). Mobile Ambients. Theoretical Computer Science 240, 1777–213.

Other Related Process Calculi (cont'd)

- Bigraphs (Milner, 2006)
  - Milner, R. (2006). Pure bigraphs: structure and dynamics. Information and Computation 204(1):60–122.
  - http://www.itu.dk/research/pls/wiki/index.php/A_Brief_Introduction_To_Bigraphs
- Calculus for Context-Awareness (Zimmer 2005, Bucur 2008)
  - Zimmer, P. (2005). A Calculus for Context-awareness. Tech. Rep., BRICS.
  - Bucur, D. Nielsen, M. (2008). Secure Data Flow in a Calculus for Context Awareness. Lecture Notes in Computer Science Vol. 5065. Springer, PP. 439–456.

Main Features of CCA

- A single concept, **ambient**, to model any entity of a UbiCom system: users, devices, location, sensors, buildings and so on.
- Mobility à la Ambient Calculus:
  - ambients can be mobile, e.g. a user, a robot or a car.
- A modal logic for context specification
- Context-awareness/-adaptation: two mechanisms
  - context guarded capabilities
  - process abstraction/call
- A theory of contextual equivalence to establish whether two context-aware systems are equivalent.

Syntax of CCA

3 main syntactic categories:

- Processes: which denote a compuation
- Capabilities: which are elementary actions a process can perform
- Context expression: which is a formula expressing some property of a process' context.

Syntax of Processes and Capabilities

$P, Q ::=$		**Process**	
	$\mathbf{0}$	inactivity	inherited from Ambient Calculus
	$n[P]$	ambient	
	$(\nu n)\ P$	restriction	
	$P \mid Q$	composition	
	$!P$	replication	
	$\kappa?M.P$	context-guarded capability	
	$x \triangleright (\tilde{y}).P$	process abstraction x	

$M ::= \quad \texttt{in}\ n \mid \texttt{out} \mid \texttt{del}\ n \mid \alpha\ x\langle\tilde{z}\rangle \mid \alpha\ (\tilde{y}) \mid \alpha\ \langle\tilde{z}\rangle$ **capabilities**

$\alpha ::= \quad \uparrow \mid n\uparrow \mid \downarrow \mid n\downarrow \mid :: \mid n :: \mid \epsilon$ **locations**

Syntax of Processes and Capabilities

$P, Q ::=$		**Process**	
	$\mathbf{0}$	inactivity	inherited from Ambient Calculus
	$n[P]$	ambient	
	$(\nu n)\ P$	restriction	
	$P \mid Q$	composition	
	$!P$	replication	
	$\kappa?M.P$	context-guarded capability	
	$x \triangleright (\tilde{y}).P$	process abstraction x	

$M ::= \quad \texttt{in}\ n \mid \texttt{out} \mid \texttt{del}\ n \mid \alpha\ x\langle\tilde{z}\rangle \mid \alpha\ (\tilde{y}) \mid \alpha\ \langle\tilde{z}\rangle$ **capabilities**

$\alpha ::= \quad \uparrow \mid n\uparrow \mid \downarrow \mid n\downarrow \mid :: \mid n :: \mid \epsilon$ **locations**

Syntax of Processes and Capabilities

$P, Q ::=$		**Process**	
	$\mathbf{0}$	inactivity	inherited from Ambient Calculus
	$n[P]$	ambient	
	$(\nu n)\ P$	restriction	
	$P \mid Q$	composition	
	$!P$	replication	
	$\kappa?M.P$	context-guarded capability	
	$x \triangleright (\tilde{y}).P$	process abstraction x	

$M ::= \quad \texttt{in}\ n \mid \texttt{out} \mid \texttt{del}\ n \mid \alpha\ x\langle\tilde{z}\rangle \mid \alpha\ (\tilde{y}) \mid \alpha\ \langle\tilde{z}\rangle$ **capabilities**

$\alpha ::= \quad \uparrow \mid n\uparrow \mid \downarrow \mid n\downarrow \mid :: \mid n :: \mid \epsilon$ **locations**

Inactive Process

The inactive process or null process **0**

- does nothing
- terminates immediately
- is the unit element for the parallel composition of processes, i.e.

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

where the symbol '$\equiv$' means 'equivalent'.

Parallel Composition of Processes

$P \mid Q$

- means that the process P and the process Q execute concurrently
- we also say P and Q execute in parallel
- i.e. P and Q execute at the same time
- during their execution, P and Q can communicate by message passing

Parallel Composition of Processes

Properties of parallel composition of processes

- $\mathbf{0}$ is the unit element

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

- Commutative

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

- Associative

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

Replication $!P$

- $!P$ denotes a process which can always creates a new copy of P
- $!P \equiv P \mid !P$
- Is used to model iteration and recursion
- First introduced by Milner (1999) in the π-Calculus
- Examples:
  - This process always send the value 2:

$$!\langle 2 \rangle.\mathbf{0}$$

  - This process always prompts to receive a message and then simply forwards the received message:

$$!(x).\langle x \rangle.\mathbf{0}$$

Restriction $(\nu n)\ P$

The restriction $(\nu n)\ P$

- states that the scope of the name n is limited to the process P.
- So the name n cannot be referred to outside P.
- Similar to the declaration of local variables in programming languages.
- Also use to hide the name n in P so that no other process can 'see' it.

What is an ambient?

An **ambient**

- is a bounded place in which computation take place
- has a name, a boundary and can be nested inside an other ambient
- can be mobile
- first introduce by Cardelli and Gordon (2000) in their Ambient Calclus

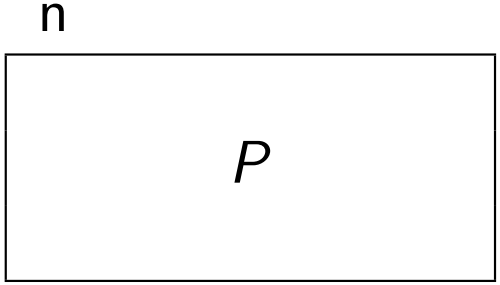
Syntax of ambient

- Textual:

$$n[P]$$

where n is the ambient's name and P is a computation (process) describing the behaviour of this ambient.

- Graphical



Modelling with ambients: smart devices and users

- A smart mobile phone **device** can be modelled as an ambient:

$$phone[P]$$

  where P encompasses the fonctionalities of the smart phone.

- the **user**, *bob*, carrying the phone can also be modelled as an ambient:

$$bob[phone[P] \mid Q]$$

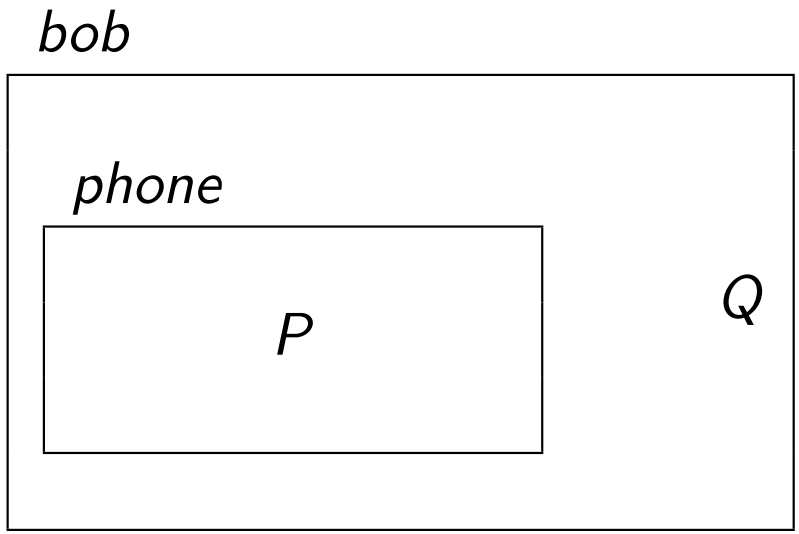
  Q here characterises the behaviour of the user, *bob*, including the user (implicit) interactions with the phone.

Modelling with ambients: smart devices and users

- So, Bob carrying a phone is modelled as

$$bob[phone[P] \mid Q]$$

- Graphically it looks like this:

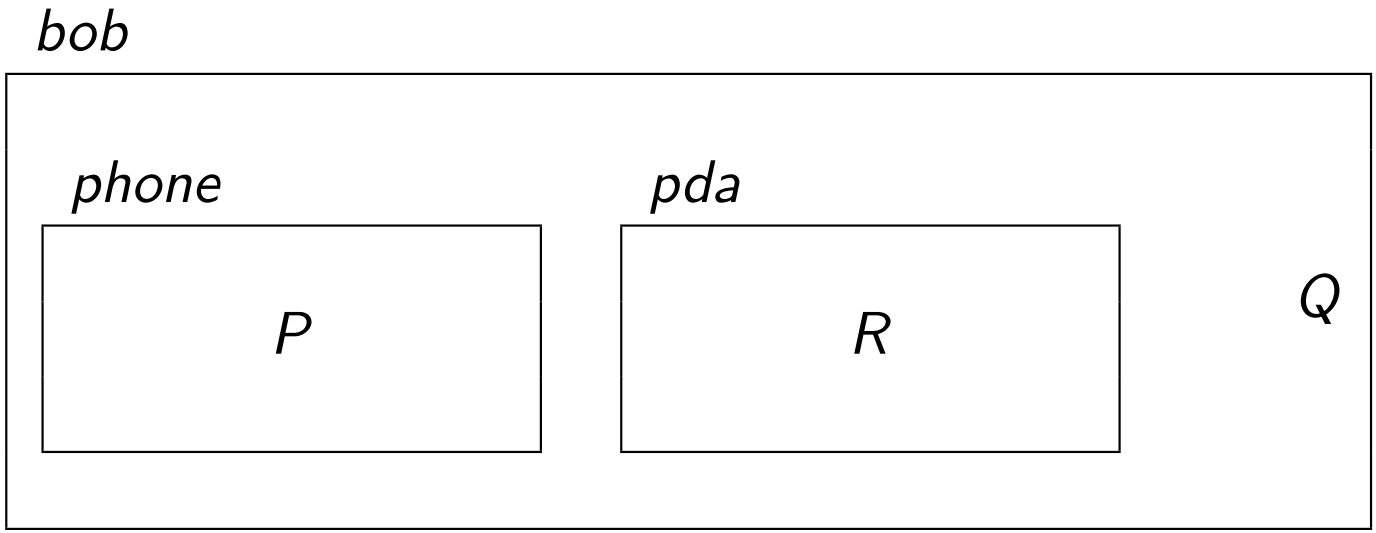


Modelling with ambients: smart devices and users

- Bob might carry more than one device: a phone and a PDA

$$bob[phone[P] \mid pda[R] \mid Q]$$

- Graphically it looks like this:

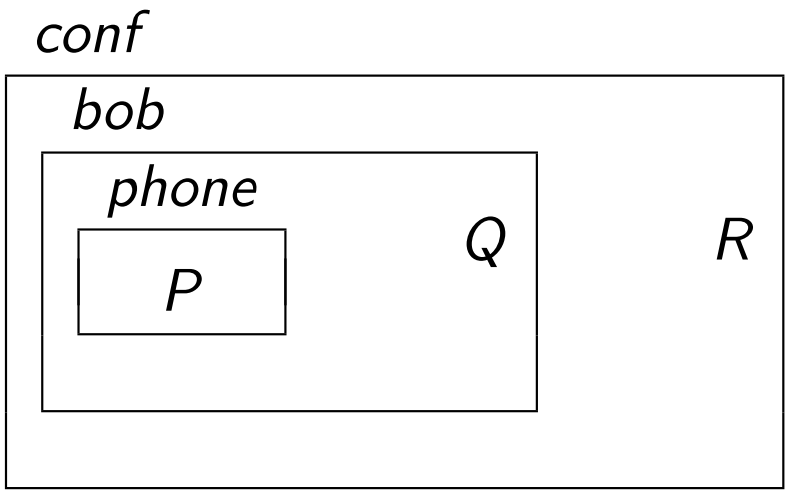


Modelling with ambients: user's location

- The **location** of the user can be modelled as an ambient
- E.g. Bob is carrying a phone at the conference room

$$conf\,[bob[phone[P] \mid Q] \mid R]$$

- Graphically it looks like this:

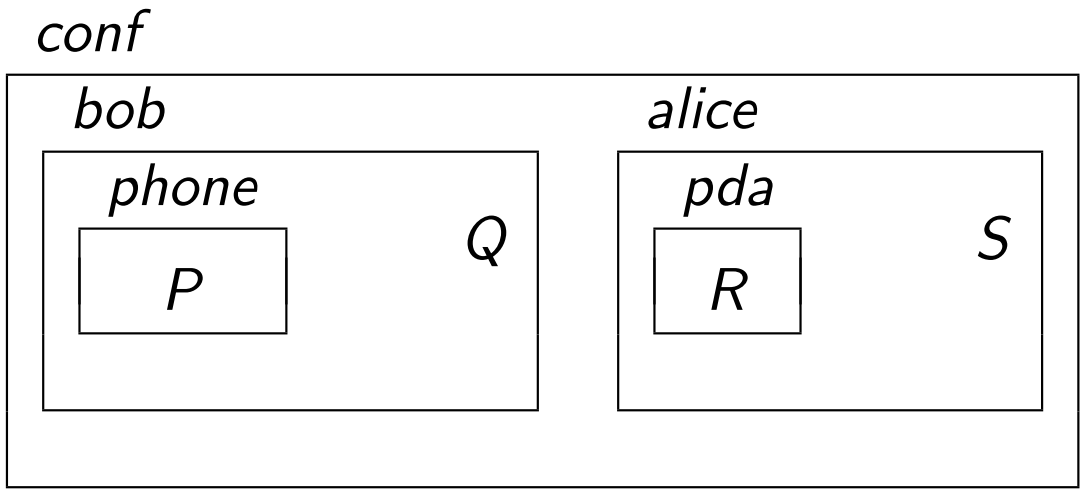


Modelling with ambients: user is with an other person

- Bob **is with** Alice at the conference room, Bob carrying a mobile phone while Alice holding a PDA

$$conf\,[bob[phone[P] \mid Q] \mid alice[pda[R] \mid S]]$$

- Graphically it looks like this:

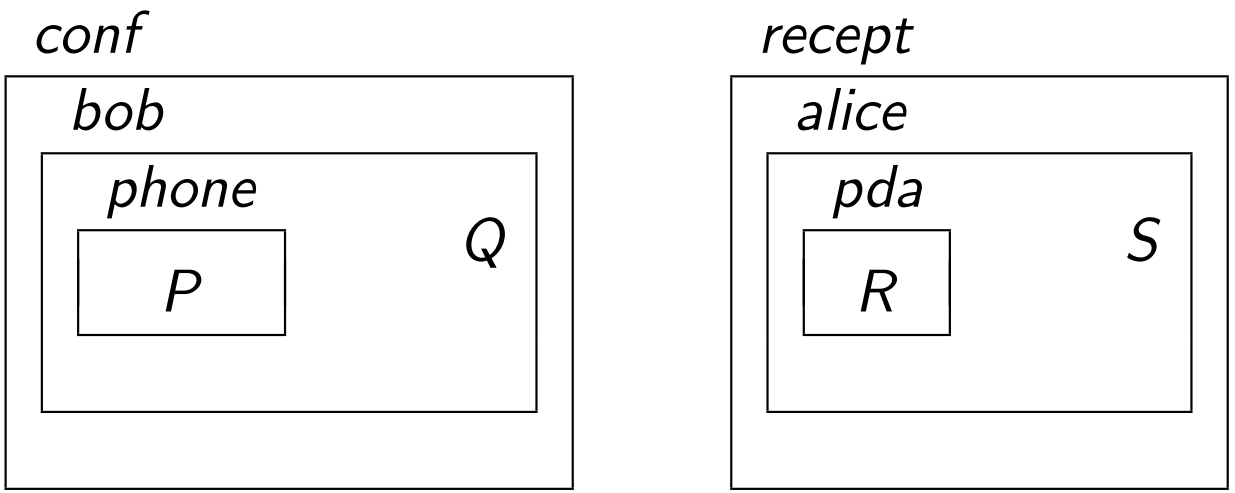


Modelling with ambients: user/device enviroment

- the conference room **is next to** the reception room
- Bob **is at** the conference room , **carrying** a mobile phone
- Alice **is at** the reception room , **carrying** a pda

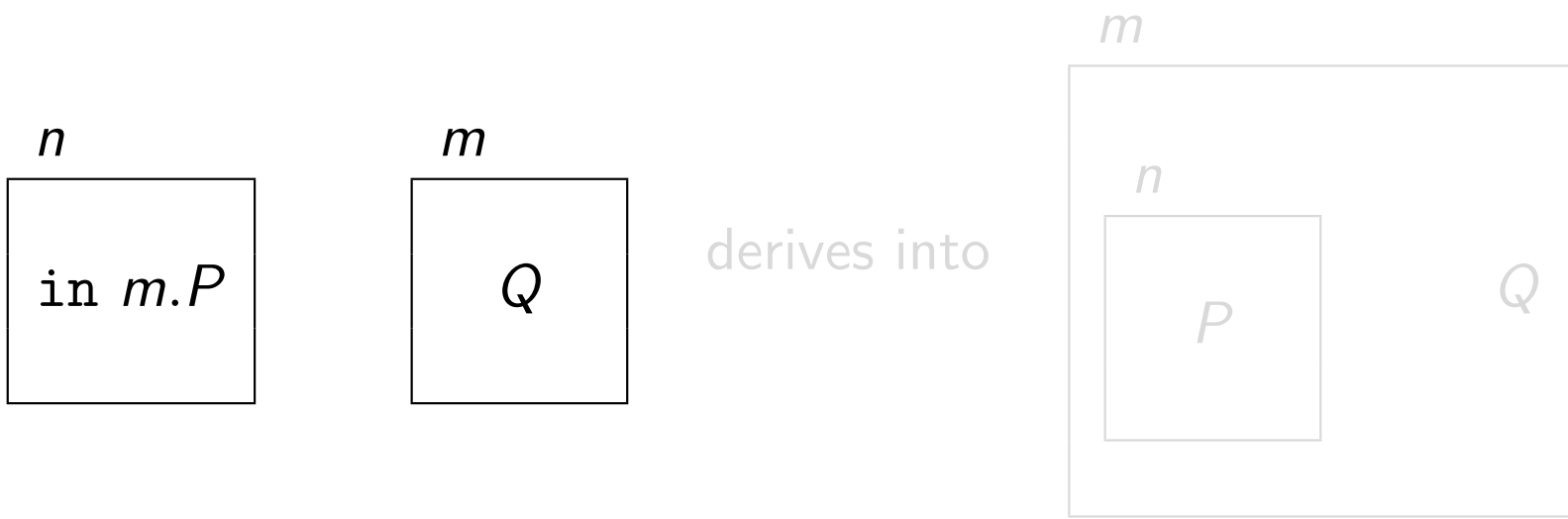
$$conf[bob[phone[P] \mid Q]] \mid recept[alice[pda[R] \mid S]]$$

- Graphically it looks like this:



Mobility Primitives

- Two mobility capabilities: `in` and `out`
- `in` m: allows an ambient to move into a sibling ambient m

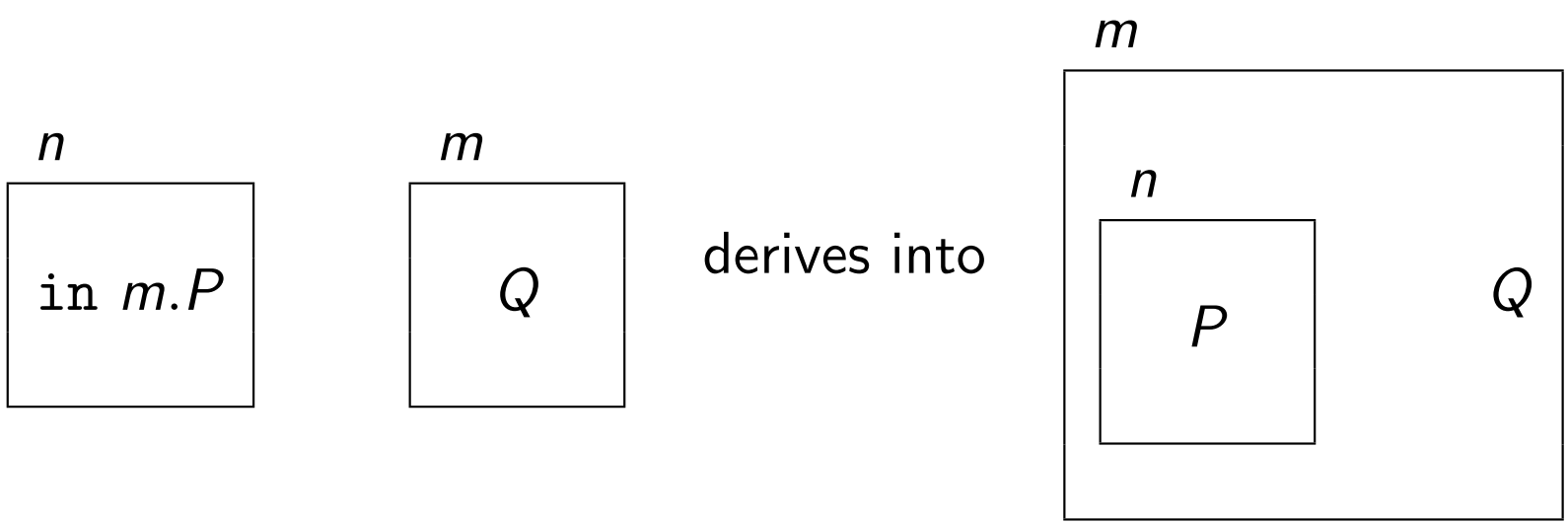


- The corresponding **reduction rule**

$$n[\texttt{in}\ m.P] \mid m[Q] \quad \longrightarrow \quad m[n[P] \mid Q]$$

Mobility Primitives

- Two mobility capabilities: `in` and `out`
- `in` m: allows an ambient to move into a sibling ambient m

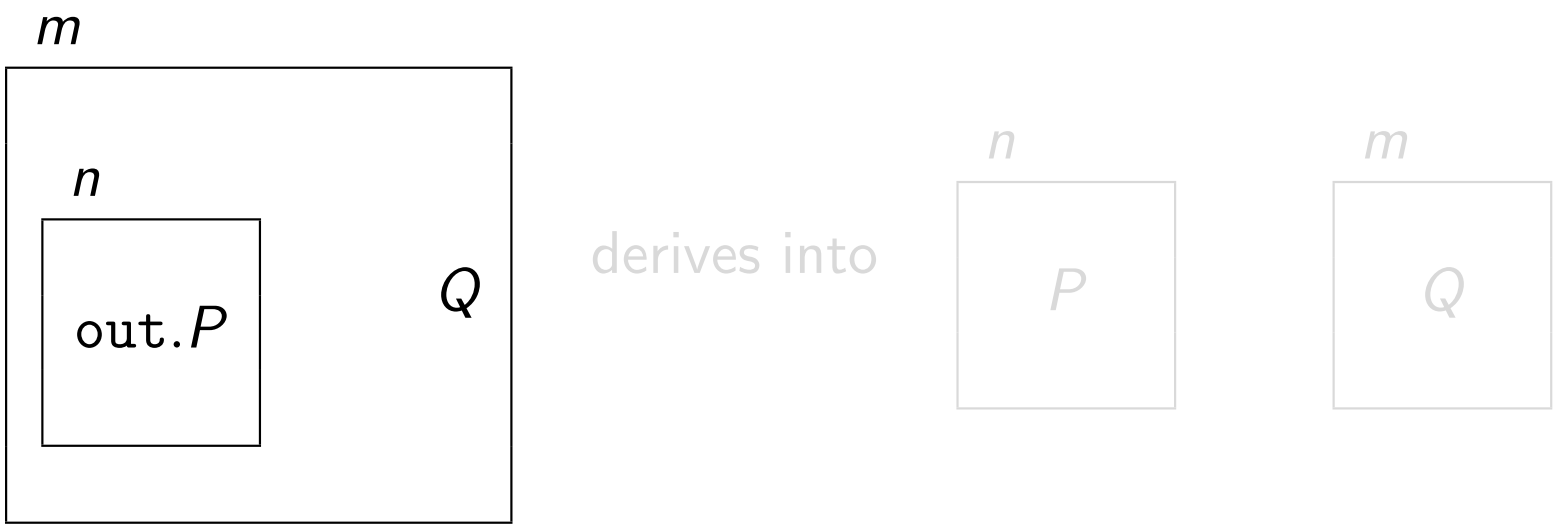


- The corresponding **reduction rule**

$$n[\texttt{in}\ m.P] \mid m[Q] \longrightarrow m[n[P] \mid Q]$$

Mobility Primitives

- out: allows an ambient to move out of its parent

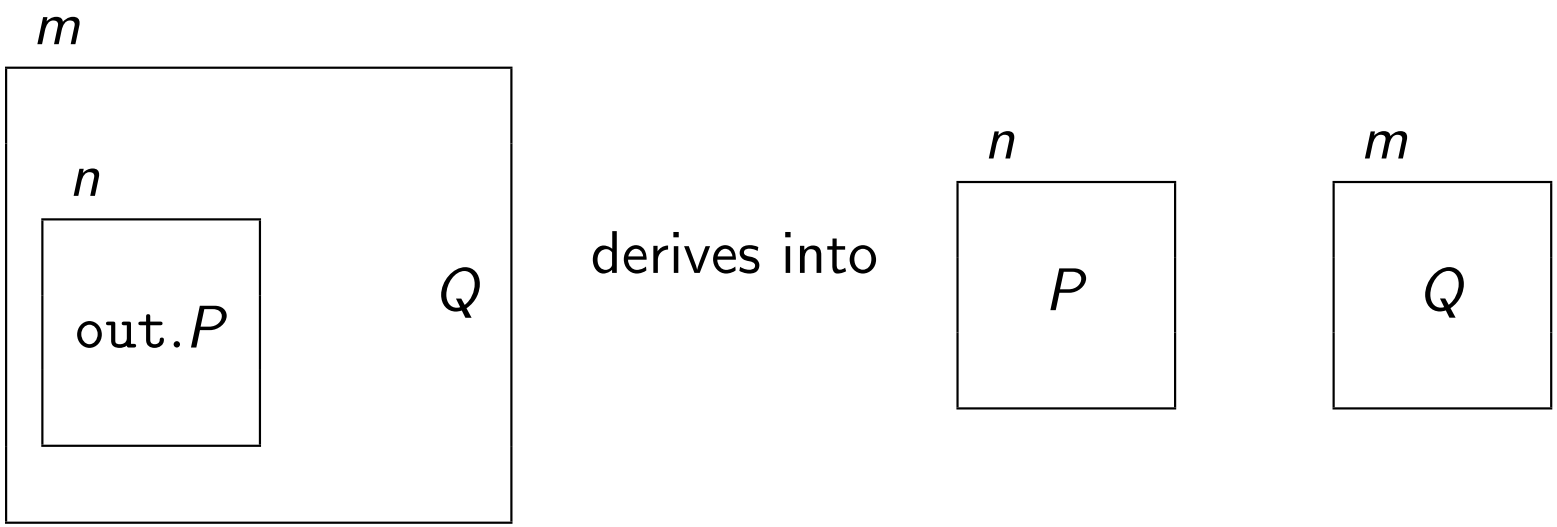


- The corresponding **reduction rule**

$$m[n[\texttt{out}.P] \mid Q] \quad \longrightarrow \quad n[P] \mid m[Q]$$

Mobility Primitives

- out: allows an ambient to move out of its parent



- The corresponding **reduction rule**

$$m[n[\texttt{out}.P] \mid Q] \quad \longrightarrow \quad n[P] \mid m[Q]$$

Mobility Primitives: Examples

- Agent *ag* moves into computer *win*.

$$ag[\texttt{in}\ win.\texttt{out}.\mathbf{0}] \mid win[\mathbf{0}] \quad \rightarrow \quad win[ag[\texttt{out}.\mathbf{0}]]$$

- Agent *ag* exits computer *win*.

$$win[ag[\texttt{out}.\mathbf{0}]] \quad \rightarrow \quad ag[\mathbf{0}] \mid win[\mathbf{0}]$$

Mobility Primitives: Examples

- Agent *ag* moves into computer *win*.

$$ag[\texttt{in}\ win.\texttt{out}.\mathbf{0}] \mid win[\mathbf{0}] \quad \rightarrow \quad win[ag[\texttt{out}.\mathbf{0}]]$$

- Agent *ag* exits computer *win*.

$$win[ag[\texttt{out}.\mathbf{0}]] \quad \rightarrow \quad ag[\mathbf{0}] \mid win[\mathbf{0}]$$

Communication Primitives

- Message passing communication: one process (**the sender**) sends a message and another one (**the receiver**) receives it.

$$\begin{array}{lcl} \textbf{Sender} & : & \langle y \rangle.P \\ \textbf{Receiver} & : & (x).Q \end{array}$$

- The **reduction rule** for message passing is

$$\langle y \rangle.P \mid (x).Q \quad \longrightarrow \quad P \mid Q\{x \leftarrow y\}$$

where $Q\{x \leftarrow y\}$ denotes the substitution of y for each free occurrence of x in Q.

- Example
  - $\langle n \rangle.\mathbf{0} \mid (x).x[\mathbf{0}] \quad \longrightarrow \quad n[\mathbf{0}]$
  - $\langle 5 \rangle.\mathbf{0} \mid (x).\mathtt{print}\langle x \rangle.\mathbf{0} \quad \longrightarrow \quad \mathtt{print}\langle 5 \rangle.\mathbf{0}$

Communication Primitives

- Communication can be **synchronous** or **asynchronous**
- **Synchronous** communication: both the sender and the receiver wait for each other

$$\begin{array}{lcl} \textbf{Sender} & : & \langle 2 \rangle.P \\ \textbf{Receiver} & : & (x).Q \end{array}$$

- **Asynchronous** communication: the sender does not wait for the communication to take place.

$$\begin{array}{lcl} \textbf{Sender} & : & \langle 2 \rangle.\mathbf{0} \mid P \\ \textbf{Receiver} & : & (x).Q \end{array}$$

Communication Primitives

Capabilities to send a message:

- $\langle a \rangle$: capability to send a message a to self
- $n \uparrow \langle a \rangle$: capability to send a message a to the parent ambient n
- $\uparrow \langle a \rangle$: capability to send a message a to any parent ambient
- $n \downarrow \langle a \rangle$: capability to send a message a to the child ambient n
- $\downarrow \langle a \rangle$: capability to send a message a to any child ambient
- $n :: \langle a \rangle$: capability to send a message a to the sibling ambient n
- $:: \langle a \rangle$: capability to send a message a to any sibling ambient

Communication Primitives

Capabilities to receive a message:

- (a): capability to receive a message a from self
- $n \uparrow (a)$: capability to receive a message a from the parent ambient n
- $\uparrow (a)$: capability to receive a message a from any parent ambient
- $n \downarrow (a)$: capability to receive a message a from the child ambient n
- $\downarrow (a)$: capability to receive a message a from any child ambient
- $n :: (a)$: capability to send receive a message a from the sibling ambient n
- $:: (a)$: capability to receive a message a from any sibling ambient

Communication Primitives

- Self communication:

- The corresponding **reduction rule**

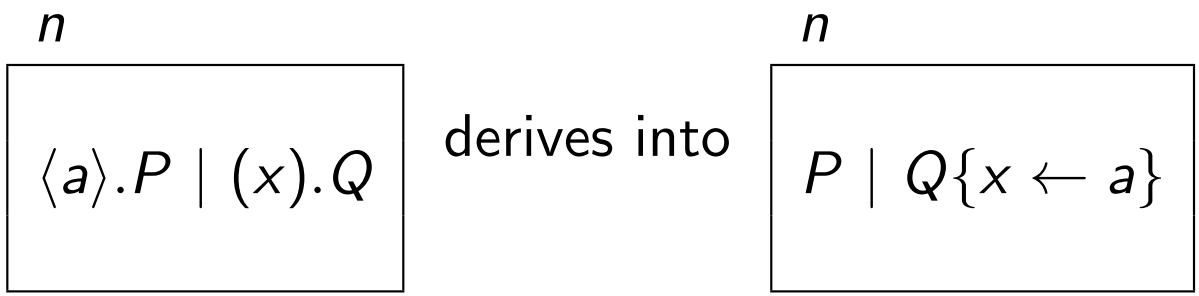
$$n[\langle a\rangle.P \mid (x).Q] \quad \longrightarrow \quad n[P \mid Q\{x \leftarrow a\}]$$

- Example

$$n[\langle 5\rangle.\mathbf{0} \mid (x).\mathtt{print}\langle x\rangle.\mathbf{0}] \quad \longrightarrow \quad n[\mathtt{print}\langle 5\rangle.\mathbf{0}]$$

Communication Primitives

- Self communication:



- The corresponding **reduction rule**

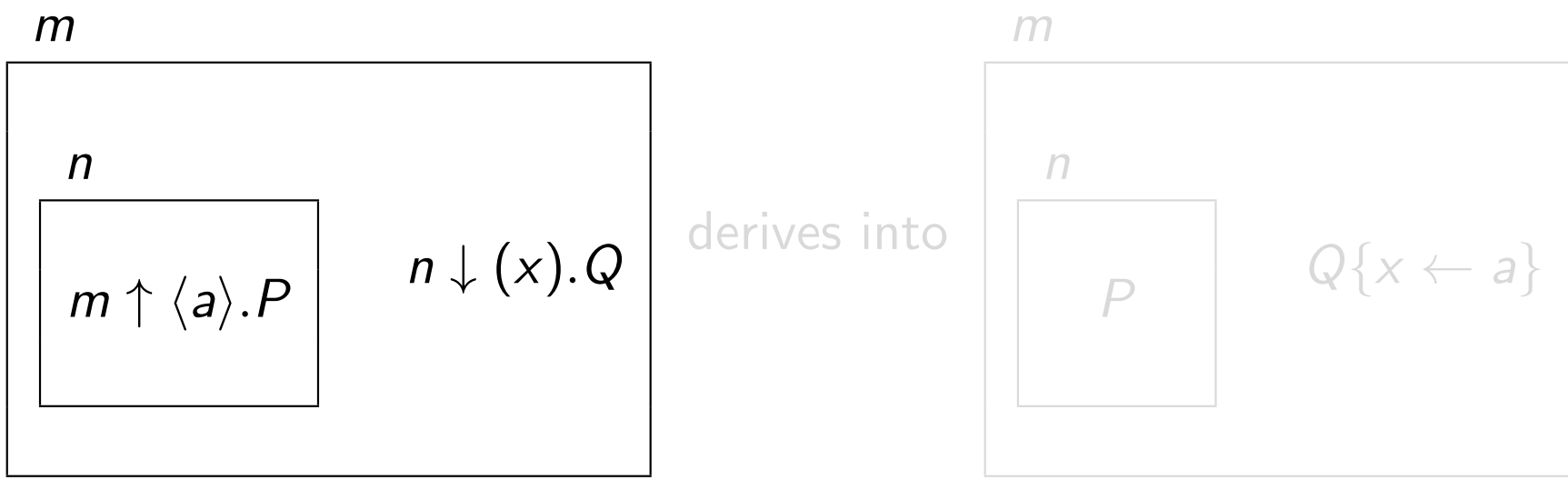
$$n[\langle a\rangle.P \mid (x).Q] \quad \longrightarrow \quad n[P \mid Q\{x \leftarrow a\}]$$

- Example

$$n[\langle 5\rangle.\mathbf{0} \mid (x).\mathtt{print}\langle x\rangle.\mathbf{0}] \quad \longrightarrow \quad n[\mathtt{print}\langle 5\rangle.\mathbf{0}]$$

Communication Primitives

- Child-parent communication:

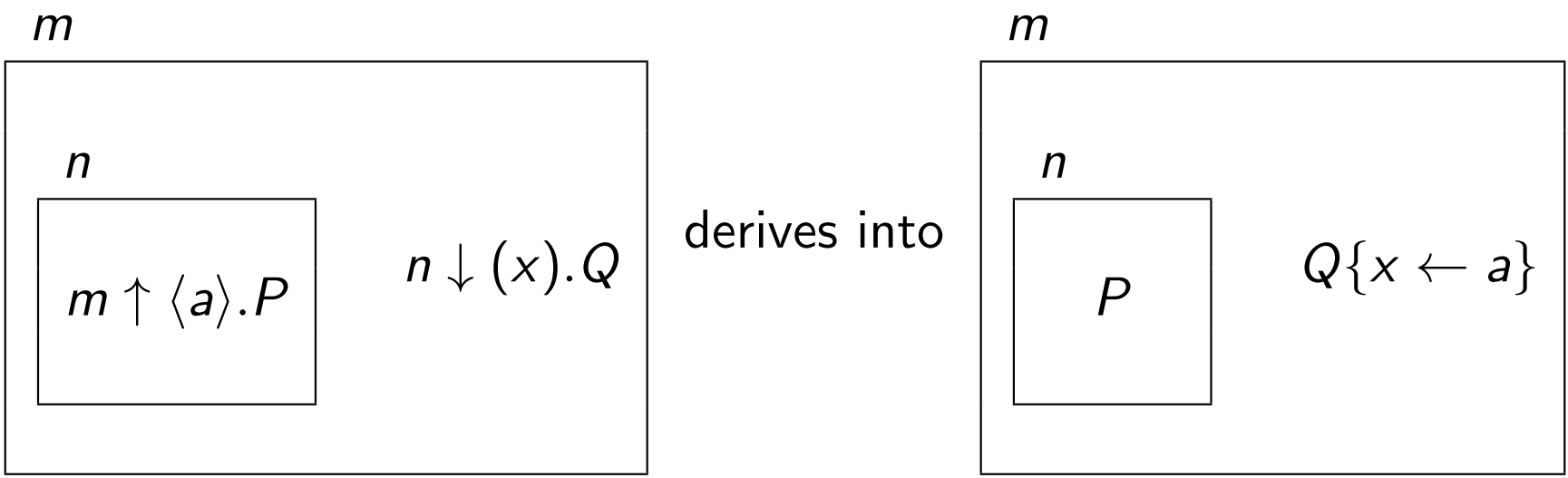


- The corresponding **reduction rule**

$$m[n[m \uparrow \langle a \rangle.P] \mid n \downarrow (x).Q] \longrightarrow m[n[P] \mid Q\{x \leftarrow a\}]$$

Communication Primitives

- Child-parent communication:



- The corresponding **reduction rule**

$$m[n[m \uparrow \langle a \rangle.P] \mid n \downarrow (x).Q] \quad \longrightarrow \quad m[n[P] \mid Q\{x \leftarrow a\}]$$

Communication Primitives

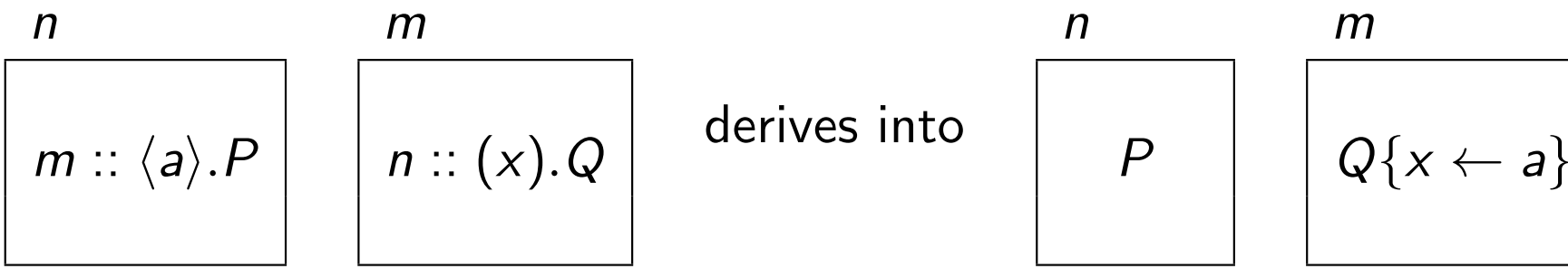
- Sibling-sibling communication:

- The corresponding **reduction rule**

$$n[m :: \langle a \rangle.P] \mid m[n :: (x).Q] \longrightarrow n[P] \mid m[Q\{x \leftarrow a\}]$$

Communication Primitives

- Sibling-sibling communication:



- The corresponding **reduction rule**

$$n[m :: \langle a \rangle.P] \mid m[n :: (x).Q] \longrightarrow n[P] \mid m[Q\{x \leftarrow a\}]$$

Communication Primitives: Exercises

- See the full semantics of communication primitives in the paper intitled "The calculus of Context-aware Ambients" available on Blackboard.
- Specify a simple Switch device with a single input port and four output ports connected to specific devices.
- Specify a simple Server that receives service requests from clients, invokes the requested services and send the results to corresponding clients.

Process Abstraction/Call

- A **process abstraction** links a name x to a process P using the syntax:

$$x \triangleright (y_1, y_2, \ldots, y_n).P$$

where $y_1, y_2, \ldots, y_n$ are formal parameters.

- A **process call** invokes a process abstraction x using the syntax:

$$\alpha x\langle z_1, z_2, \ldots, z_n\rangle.$$

where $z_1, z_2, \ldots, z_n$ are actual parameters and α indicates the location of the process abstraction called and is defined as follows:

$$\alpha ::= \uparrow \mid n\uparrow \mid \downarrow \mid n\downarrow \mid :: \mid n :: \mid \epsilon$$

Process Abstraction/Call: Semantics

- Local process call

- The corresponding **reduction rule**

$$n[x \triangleright (y).P \mid x\langle a\rangle] \longrightarrow n[x \triangleright (y).P \mid P\{y \leftarrow a\}]$$

Process Abstraction/Call: Semantics

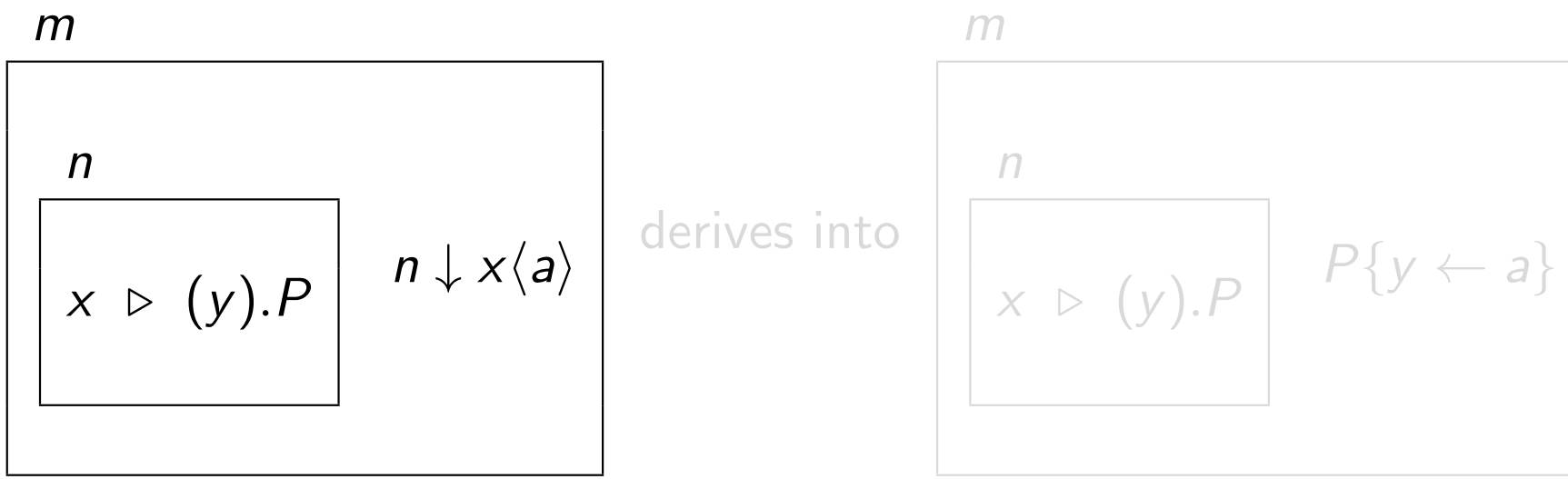
- Local process call

- The corresponding **reduction rule**

$$n[x \triangleright (y).P \mid x\langle a\rangle] \longrightarrow n[x \triangleright (y).P \mid P\{y \leftarrow a\}]$$

Process Abstraction/Call: Semantics

- Calling a process abstraction defined in a child ambient

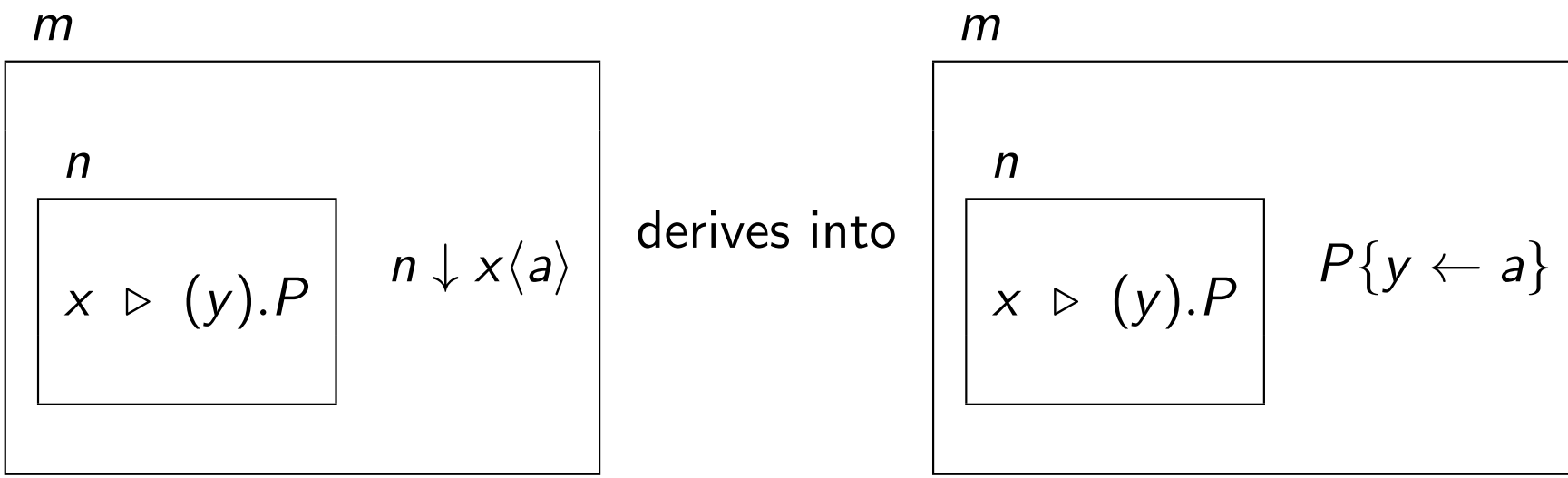


- The corresponding **reduction rule**

$$m[n[x \triangleright (y).P] \mid n \downarrow x\langle a\rangle] \longrightarrow m[n[x \triangleright (y).P] \mid P\{y \leftarrow a\}]$$

Process Abstraction/Call: Semantics

- Calling a process abstraction defined in a child ambient

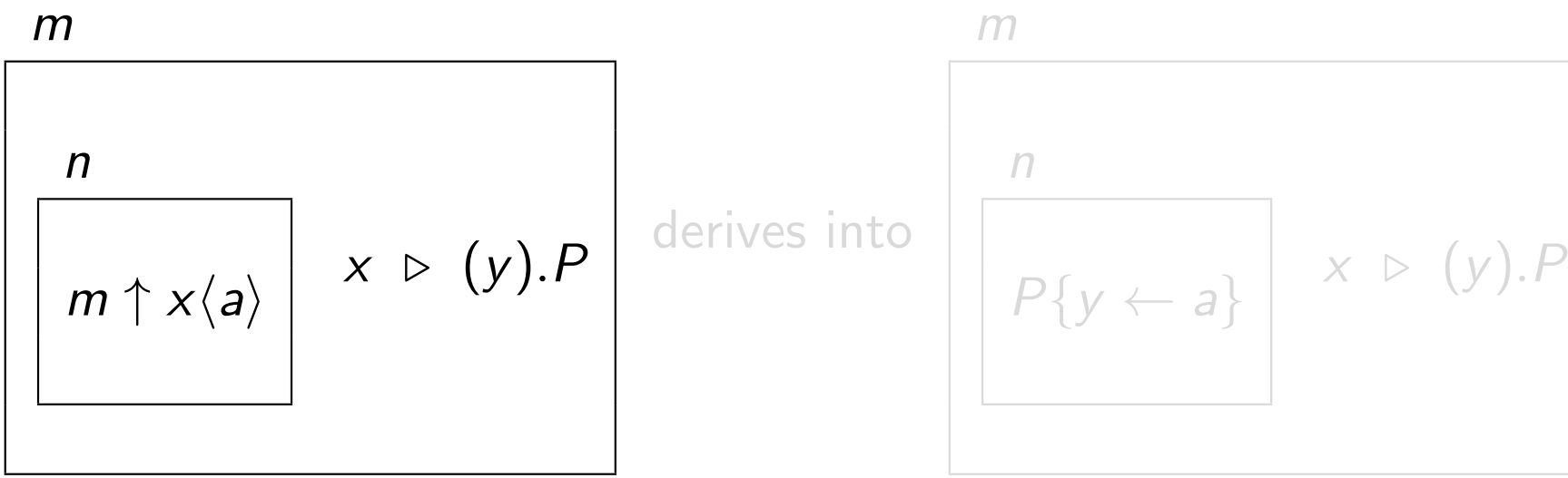


- The corresponding **reduction rule**

$$m[n[x \triangleright (y).P] \mid n \downarrow x\langle a \rangle] \longrightarrow m[n[x \triangleright (y).P] \mid P\{y \leftarrow a\}]$$

Process Abstraction/Call: Semantics

- Calling a process abstraction defined in a child ambient

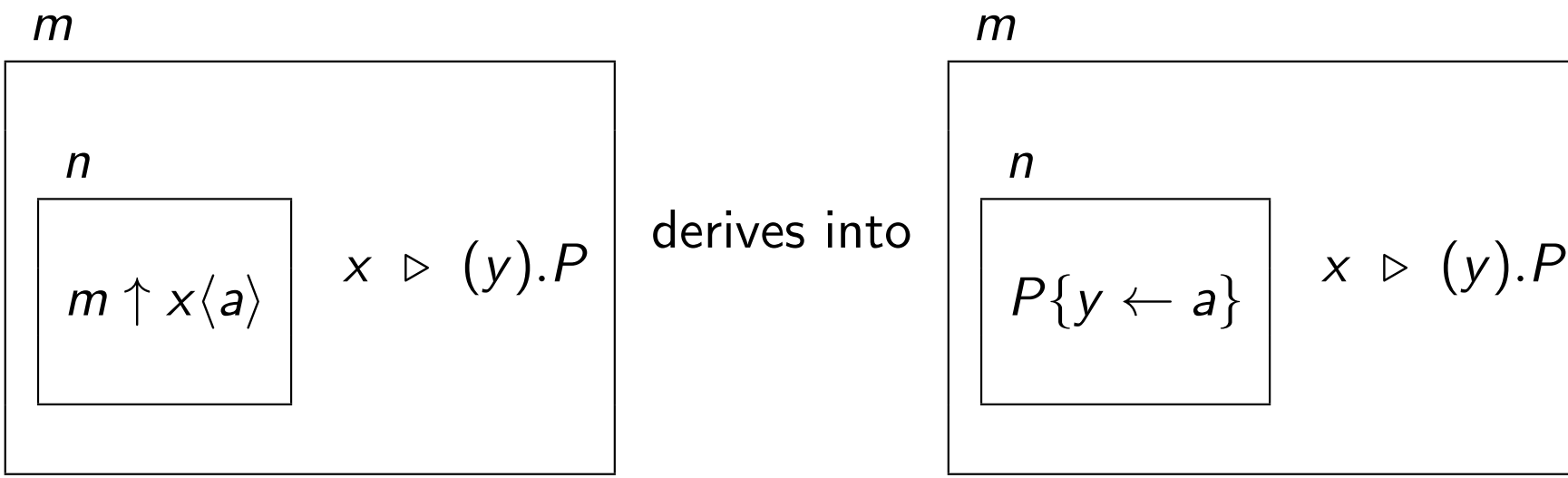


- The corresponding **reduction rule**

$$m[n[m \uparrow x\langle a \rangle] \mid x \triangleright (y).P] \longrightarrow m[n[P\{y \leftarrow a\}] \mid x \triangleright (y).P]$$

Process Abstraction/Call: Semantics

- Calling a process abstraction defined in a child ambient



- The corresponding **reduction rule**

$$m[n[m \uparrow x\langle a \rangle] \mid x \triangleright (y).P] \longrightarrow m[n[P\{y \leftarrow a\}] \mid x \triangleright (y).P]$$

Process Abstraction/Call: Semantics

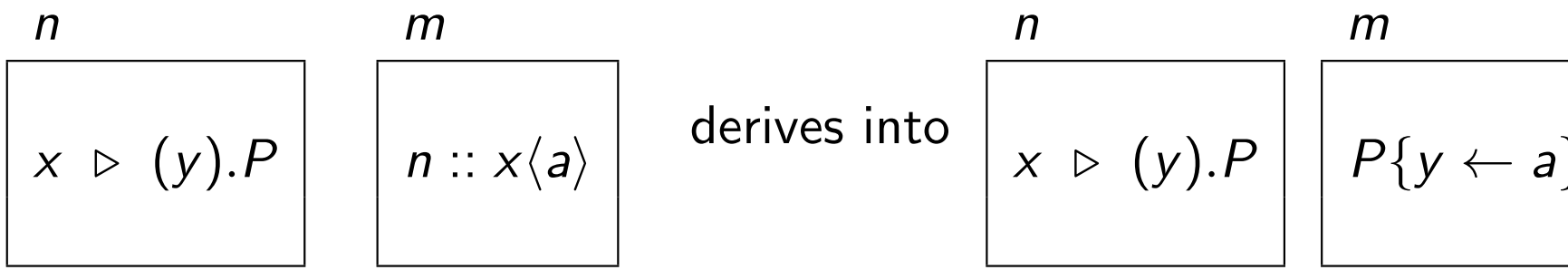
- Calling a process abstraction defined in a sibling ambient

- The corresponding **reduction rule**

$$n[x \triangleright (y).P] \mid m[n :: x\langle a \rangle] \longrightarrow n[x \triangleright (y).P] \mid m[P\{y \leftarrow a\}]$$

Process Abstraction/Call: Semantics

- Calling a process abstraction defined in a sibling ambient



- The corresponding **reduction rule**

$$n[x \triangleright (y).P] \mid m[n :: x\langle a\rangle] \longrightarrow n[x \triangleright (y).P] \mid m[P\{y \leftarrow a\}]$$

Process Abstraction/Call: Example

Context Provision using Process Abstraction

- User preferences, Available services, Device configuration
- My favourite text editor on Windows is `notepad`

$$win[\texttt{edit} \triangleright (file).\texttt{notepad}\langle file\rangle.\mathbf{0}]$$

- My favourite text editor on Linux is `emacs`

$$lin[\texttt{edit} \triangleright (file).\texttt{emacs}\langle file\rangle.\mathbf{0}]$$

A Context-aware file editing agent can be specified as

$$ag[\uparrow \texttt{edit}\langle doc.txt\rangle.\mathbf{0}]$$

Process Abstraction/Call: Example

Context Provision using Process Abstraction

- User preferences, Available services, Device configuration
- My favourite text editor on Windows is `notepad`

$$win[\texttt{edit} \triangleright (file).\texttt{notepad}\langle file\rangle.\mathbf{0}]$$

- My favourite text editor on Linux is `emacs`

$$lin[\texttt{edit} \triangleright (file).\texttt{emacs}\langle file\rangle.\mathbf{0}]$$

A Context-aware file editing agent can be specified as

$$ag[\uparrow \texttt{edit}\langle doc.txt\rangle.\mathbf{0}]$$

Context Acquisition using Process Call

When the file editing agent runs

- on my Windows machine, notepad is used

$$win[\texttt{edit} \triangleright (file).\texttt{notepad}\langle file\rangle.\mathbf{0} \quad | \quad ag[\uparrow \texttt{edit}\langle doc.txt\rangle.\mathbf{0}]]$$

$$\rightarrow$$

$$win[\texttt{edit} \triangleright (file).\texttt{notepad}\langle file\rangle.\mathbf{0} \quad | \quad ag[\texttt{notepad}\langle doc.txt\rangle.\mathbf{0}]]$$

- on my Linux machine, emacs is used

$$lin[\texttt{edit} \triangleright (file).\texttt{emacs}\langle file\rangle.\mathbf{0} \quad | \quad ag[\uparrow \texttt{edit}\langle doc.txt\rangle.\mathbf{0}]]$$

$$\rightarrow$$

$$lin[\texttt{edit} \triangleright (file).\texttt{emacs}\langle file\rangle.\mathbf{0} \quad | \quad ag[\texttt{emacs}\langle doc.txt\rangle.\mathbf{0}]]$$

Context Acquisition using Process Call

When the file editing agent runs

- on my Windows machine, notepad is used

$$win[\texttt{edit} \triangleright (file).\texttt{notepad}\langle file\rangle.\mathbf{0} \quad | \quad ag[\uparrow \texttt{edit}\langle doc.txt\rangle.\mathbf{0}]]$$

$$\rightarrow$$

$$win[\texttt{edit} \triangleright (file).\texttt{notepad}\langle file\rangle.\mathbf{0} \quad | \quad ag[\texttt{notepad}\langle doc.txt\rangle.\mathbf{0}]]$$

- on my Linux machine, emacs is used

$$lin[\texttt{edit} \triangleright (file).\texttt{emacs}\langle file\rangle.\mathbf{0} \quad | \quad ag[\uparrow \texttt{edit}\langle doc.txt\rangle.\mathbf{0}]]$$

$$\rightarrow$$

$$lin[\texttt{edit} \triangleright (file).\texttt{emacs}\langle file\rangle.\mathbf{0} \quad | \quad ag[\texttt{emacs}\langle doc.txt\rangle.\mathbf{0}]]$$

Context-guarded Action Capabilities

- **Syntax:** $\kappa?M.P$
  where
  - κ is a condition over the environment, called context expression
  - M is an action capability
  - P is a continuation process
- **Semantics:**

  waits until the environment satisfies the context expression κ, then performs the capability M and continues like the process P.

Context-guarded Action Capabilities

- **Syntax:** $\kappa?M.P$
  where
  - κ is a condition over the environment, called context expression
  - M is an action capability
  - P is a continuation process
- **Semantics:**

  waits until the environment satisfies the context expression κ, then performs the capability M and continues like the process P.

Example: a Smart Phone

Requirements

- carried by the user
- silent when user at conference
- divert calls when user with a friend Alice say

$$phone \left[\begin{array}{l} !\,\mathtt{usr\_with}(alice)?\mathtt{switchto}\langle divert\rangle.\mathbf{0} \\ !\,(\mathtt{usr\_at}(conf) \wedge \neg\mathtt{usr\_with}(alice))?\mathtt{switchto}\langle silent\rangle.\mathbf{0} \\ !\,(\neg\mathtt{usr\_at}(conf) \wedge \neg\mathtt{usr\_with}(alice))?\mathtt{switchto}\langle normal\rangle.\mathbf{0} \end{array} \right]$$

Example: a Smart Phone

Requirements

- carried by the user
- silent when user at conference
- divert calls when user with a friend Alice say

$$\mathit{phone}\left[\begin{array}{l} !\ \mathtt{usr\_with}(\mathit{alice})?\mathtt{switchto}\langle \mathit{divert}\rangle.\mathbf{0} \\ !\ (\mathtt{usr\_at}(\mathit{conf}) \wedge \neg\mathtt{usr\_with}(\mathit{alice}))?\mathtt{switchto}\langle \mathit{silent}\rangle.\mathbf{0} \\ !\ (\neg\mathtt{usr\_at}(\mathit{conf}) \wedge \neg\mathtt{usr\_with}(\mathit{alice}))?\mathtt{switchto}\langle \mathit{normal}\rangle.\mathbf{0} \end{array}\right|$$

What is Context?

- Dey (1993):

  > **Context** *is any* **information** *that can be used to characterise the* **situation** *of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.*

- Important aspects of context include:
  - user location
  - nearby people
  - available resources
  - user activity

What is Context?

- Dey (1993):

  > **Context** *is any* **information** *that can be used to characterise the* **situation** *of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.*

- Important aspects of context include:
  - user location
  - nearby people
  - available resources
  - user activity

What is Context?

- Dey (1993):

  > **Context** *is any* **information** *that can be used to characterise the* **situation** *of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.*

- Important aspects of context include:
  - user location
  - nearby people
  - available resources
  - user activity

Context Model

- In CCA, a process is described as a hierarchy of nested ambients.
- So, the context of each individual (sub-)process correspond to the surrounding of that process in the hierarchy.
- This is obtained by replacing the process by a 'hole' in the hierarchy.
- A hole is represented by the symbol $\odot$ and can be thought of as a place holder.

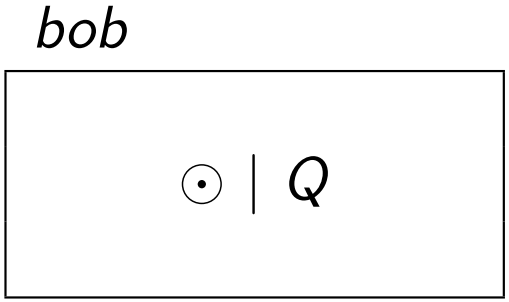
Examples of Context (cont'd)

- Context of a mobile device carried by a user, Bob:

$$bob[\odot \mid Q]$$

where Q is a continuation process.

- Graphically it looks like this:

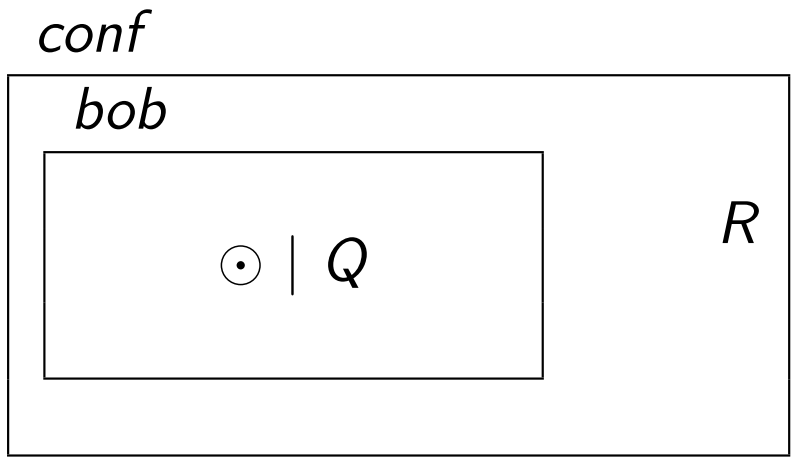


Examples of Context (cont'd)

- Context of a mobile device carried by Bob at the conference room:

$$conf\,[bob[\odot \mid Q] \mid R]$$

  where Q and R are continuation processes.
- Graphically it looks like this:

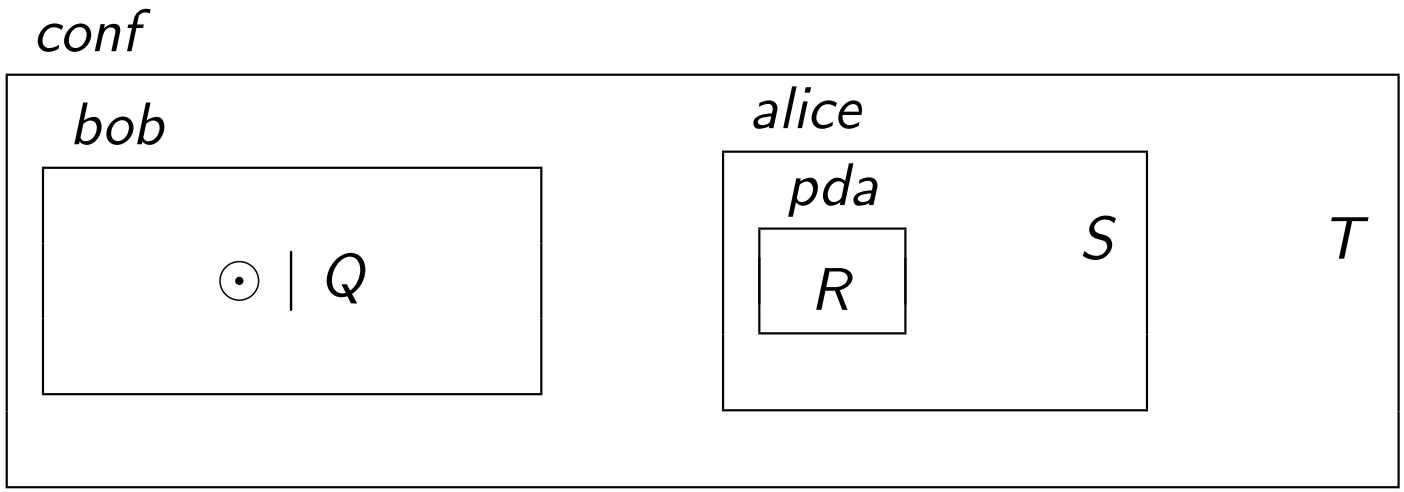


Examples of Context (cont'd)

- Context of a smart device carried by Bob who is with Alice at conference, Alice carrying a pda:

$$conf\,[bob[\odot \mid Q] \mid alice[pda[R] \mid S] \mid T]$$

- Graphically it looks like this:



Context Model: Syntax

In CCA, a context is described by the following syntax:

C	::=		**Context**
		$\mathbf{0}$	nil
		$\odot$	hole
		$n[C]$	location
		$C \mid P$	parallel composition
		$(\nu n)\ C$	restriction

where the symbol C stands for context (environment), n ranges over names and P ranges over processes.

Context Model: Syntax (cont'd)

- The context $\mathbf{0}$ is the empty context, also called the nil context. It contains no context information.
- The position of a process in that process' context is denoted by the symbol $\odot$. This is a special context called the hole context.
- The context $(\nu n)\ C$ means that the scope of the name n is limited to the context C.
- The context $n[C]$ means that the internal environment of the ambient n is described by the context C.
- The context $C \mid P$ says that the process P runs in parallel with the context C, and so C is part of process P's context.

Context Model: some definitions

- **Ground context**: is a context containing no holes

  Example: any process is a ground context.

- **Context evaluation**: Let C_1 and C_2 be contexts. The evaluation of the context C_1 at the context C_2, denoted by $C_1(C_2)$, is the context obtained by replacing the hole in C_1 (if any) by C_2, viz

$$C_1(C_2) = \begin{cases} C_1 & \textit{if } C_1 \textit{ is a ground context} \\ C_1\{\odot \leftarrow C_2\} & \textit{otherwise.} \end{cases}$$

  where $C_1\{\odot \leftarrow C_2\}$ is the substitution of C_2 for $\odot$ in C_1.

Context Model: Examples

- Let e_1 be the context of a mobile device carried by Bob at the conference room:

$$e_1 \;\widehat{=}\; conf\,[bob[\odot \mid Q]]$$

- Then the process

$$e_1(phone[P]) \;\widehat{=}\; conf\,[bob[phone[P] \mid Q]]$$

models Bob at the conference room carrying a mobile phone

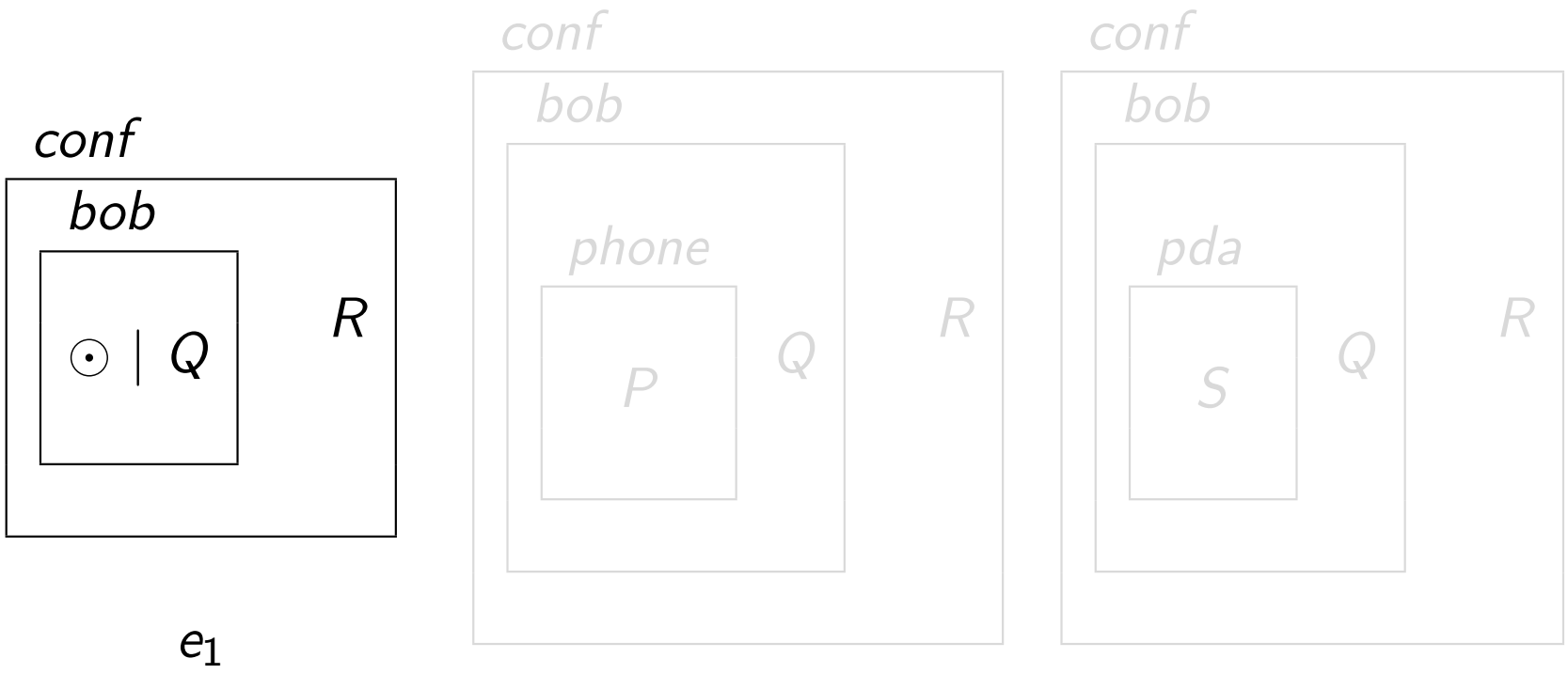
- while the process

$$e_1(pda[S]) \;\widehat{=}\; conf\,[bob[pda[S] \mid Q]]$$

models Bob at the conference room carrying a PDA

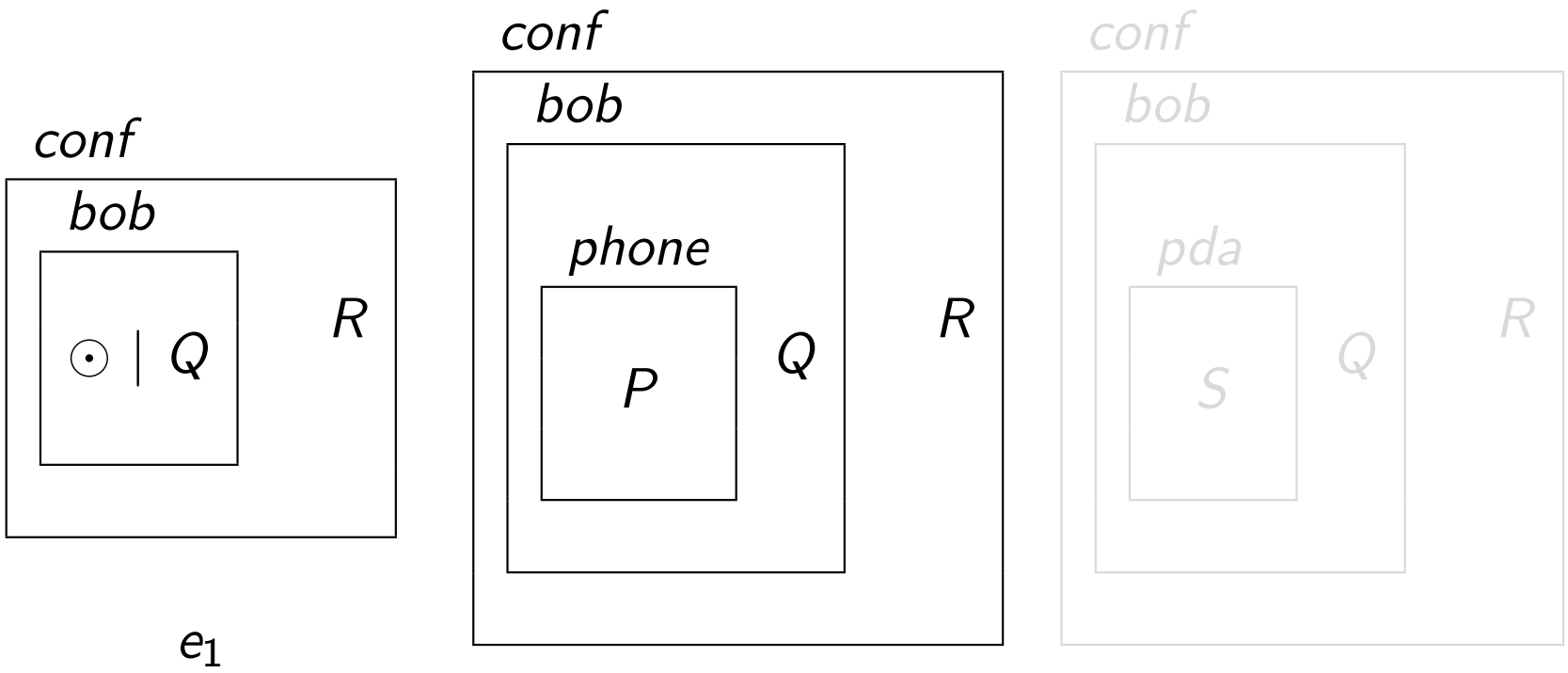
Context Model: Examples (cont'd)

Graphically it looks like this:



Context Model: Examples (cont'd)

Graphically it looks like this:



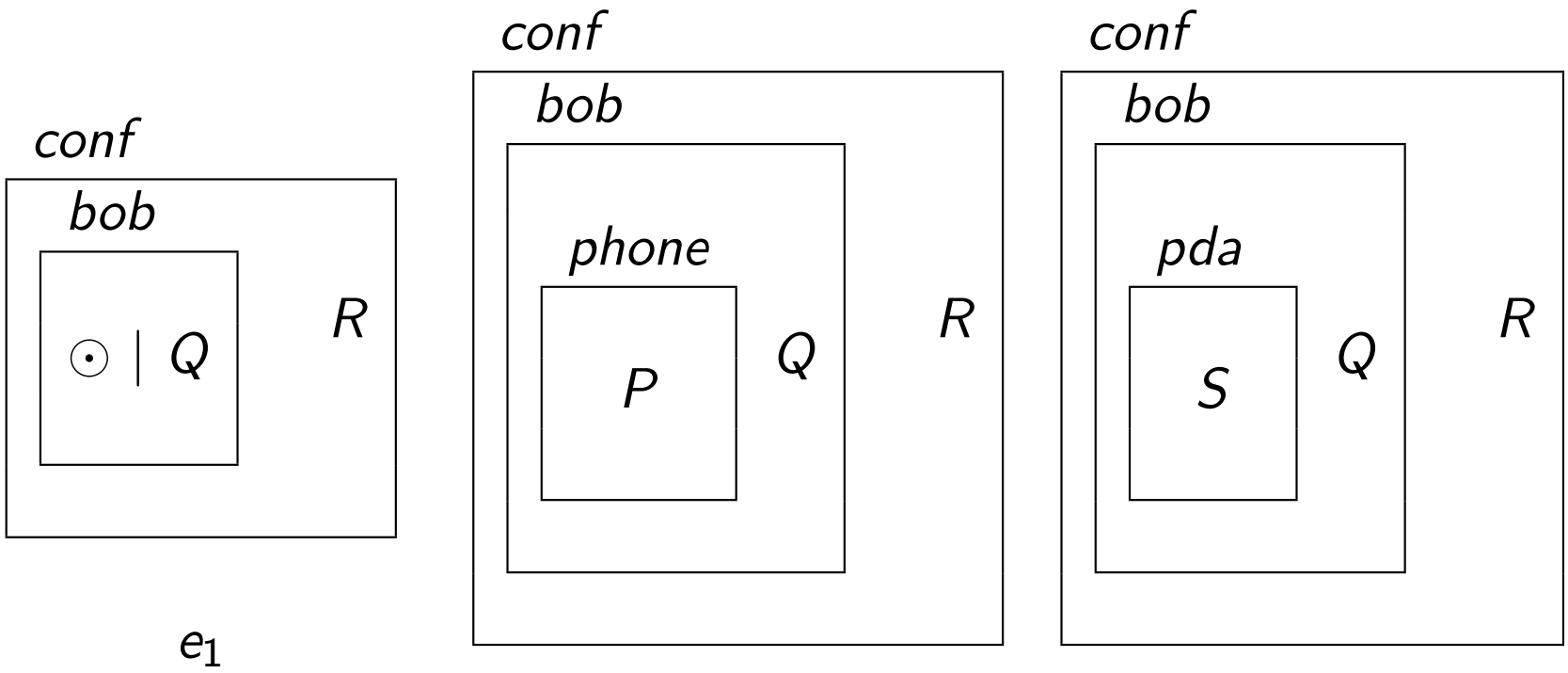
e_1

$e_1(phone[P])$

$e_1(pda[S])$

Context Model: Examples (cont'd)

Graphically it looks like this:



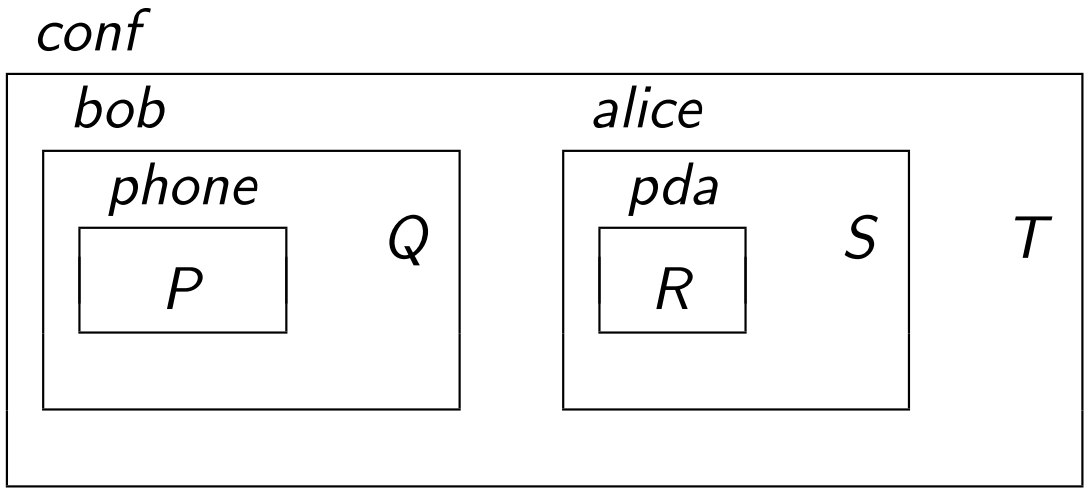
e_1

$e_1(phone[P])$

$e_1(pda[S])$

Context Model: Exercise

- Consider the following process:



- What is the context of the ambient *phone* in that process?
- What is the context of the ambient *alice* in that process?
- What is the context of the process T in that process?

Algebraic Semantics of Context

- The algebraic semantics of the context model is given in terms of the following equalities

$$C \mid \mathbf{0} = C \qquad \text{(Cont-0)}$$

$$C_1 \mid C_2 = C_2 \mid C_1 \qquad \text{(Cont-1)}$$

$$C_1 \mid (C_2 \mid C_3) = (C_1 \mid C_2) \mid C_3 \qquad \text{(Cont-2)}$$

$$(\nu n)\ (\nu m)\ C = (\nu m)\ (\nu n)\ C \qquad \text{(Cont-3)}$$

$$(\nu n)\ C_1 \mid C_2 = (\nu n)\ (C_1 \mid C_2) \qquad \text{if } n \notin \mathbf{fn}(C_2) \qquad \text{(Cont-4)}$$

- where $\mathbf{fn}(C_2)$ is the set of free names in C_2

Algebraic Semantics of Context

- The algebraic semantics of the context model is given in terms of the following equalities

$$C \mid \mathbf{0} = C \qquad \text{(Cont-0)}$$

$$C_1 \mid C_2 = C_2 \mid C_1 \qquad \text{(Cont-1)}$$

$$C_1 \mid (C_2 \mid C_3) = (C_1 \mid C_2) \mid C_3 \qquad \text{(Cont-2)}$$

$$(\nu n)\ (\nu m)\ C = (\nu m)\ (\nu n)\ C \qquad \text{(Cont-3)}$$

$$(\nu n)\ C_1 \mid C_2 = (\nu n)\ (C_1 \mid C_2) \qquad \text{if } n \notin \mathbf{fn}(C_2) \qquad \text{(Cont-4)}$$

- where $\mathbf{fn}(C_2)$ is the set of free names in C_2

Algebraic Semantics of Context

- The algebraic semantics of the context model is given in terms of the following equalities

$$C \mid \mathbf{0} = C \qquad \text{(Cont-0)}$$

$$C_1 \mid C_2 = C_2 \mid C_1 \qquad \text{(Cont-1)}$$

$$C_1 \mid (C_2 \mid C_3) = (C_1 \mid C_2) \mid C_3 \qquad \text{(Cont-2)}$$

$$(\nu n)\ (\nu m)\ C = (\nu m)\ (\nu n)\ C \qquad \text{(Cont-3)}$$

$$(\nu n)\ C_1 \mid C_2 = (\nu n)\ (C_1 \mid C_2) \quad \text{if } n \notin \mathbf{fn}(C_2) \qquad \text{(Cont-4)}$$

- where $\mathbf{fn}(C_2)$ is the set of free names in C_2

Algebraic Semantics of Context

- The algebraic semantics of the context model is given in terms of the following equalities

$$C \mid \mathbf{0} = C \qquad \text{(Cont-0)}$$

$$C_1 \mid C_2 = C_2 \mid C_1 \qquad \text{(Cont-1)}$$

$$C_1 \mid (C_2 \mid C_3) = (C_1 \mid C_2) \mid C_3 \qquad \text{(Cont-2)}$$

$$(\nu n)\ (\nu m)\ C = (\nu m)\ (\nu n)\ C \qquad \text{(Cont-3)}$$

$$(\nu n)\ C_1 \mid C_2 = (\nu n)\ (C_1 \mid C_2) \qquad \text{if } n \notin \mathbf{fn}(C_2) \qquad \text{(Cont-4)}$$

- where $\mathbf{fn}(C_2)$ is the set of free names in C_2

Algebraic Semantics of Context

- The algebraic semantics of the context model is given in terms of the following equalities

$$C \mid \mathbf{0} = C \qquad \text{(Cont-0)}$$

$$C_1 \mid C_2 = C_2 \mid C_1 \qquad \text{(Cont-1)}$$

$$C_1 \mid (C_2 \mid C_3) = (C_1 \mid C_2) \mid C_3 \qquad \text{(Cont-2)}$$

$$(\nu n)\ (\nu m)\ C = (\nu m)\ (\nu n)\ C \qquad \text{(Cont-3)}$$

$$(\nu n)\ C_1 \mid C_2 = (\nu n)\ (C_1 \mid C_2) \qquad \text{if } n \notin \mathbf{fn}(C_2) \qquad \text{(Cont-4)}$$

- where $\mathbf{fn}(C_2)$ is the set of free names in C_2

Algebraic Semantics of Context

- The algebraic semantics of the context model is given in terms of the following equalities

$$C \mid \mathbf{0} = C \qquad \text{(Cont-0)}$$

$$C_1 \mid C_2 = C_2 \mid C_1 \qquad \text{(Cont-1)}$$

$$C_1 \mid (C_2 \mid C_3) = (C_1 \mid C_2) \mid C_3 \qquad \text{(Cont-2)}$$

$$(\nu n)\ (\nu m)\ C = (\nu m)\ (\nu n)\ C \qquad \text{(Cont-3)}$$

$$(\nu n)\ C_1 \mid C_2 = (\nu n)\ (C_1 \mid C_2) \qquad \text{if } n \notin \mathbf{fn}(C_2) \qquad \text{(Cont-4)}$$

- where $\mathbf{fn}(C_2)$ is the set of free names in C_2

Algebraic Semantics of Context (cont'd)

- Additional equalities

$$(\nu n)\ m[C] = m[(\nu n)\ C] \qquad \text{if } n \neq m \qquad \text{(Cont-5)}$$

$$(\nu n)\ \mathbf{0} = \mathbf{0} \qquad \text{(Cont-6)}$$

$$C_1 = C_2 \quad \Rightarrow \quad (\nu n)\ C_1 = (\nu n)\ C_2 \qquad \text{(Cont-7)}$$

$$C_1 = C_2 \quad \Rightarrow \quad C_1 \mid C_3 = C_2 \mid C_3 \qquad \text{(Cont-8)}$$

$$C_1 = C_2 \quad \Rightarrow \quad n[C_1] = n[C_2] \qquad \text{(Cont-9)}$$

Algebraic Semantics of Context (cont'd)

- Additional equalities

$$(\nu n)\ m[C] = m[(\nu n)\ C] \qquad \text{if } n \neq m \qquad \text{(Cont-5)}$$

$$(\nu n)\ \mathbf{0} = \mathbf{0} \qquad \text{(Cont-6)}$$

$$C_1 = C_2 \quad \Rightarrow \quad (\nu n)\ C_1 = (\nu n)\ C_2 \qquad \text{(Cont-7)}$$

$$C_1 = C_2 \quad \Rightarrow \quad C_1 \mid C_3 = C_2 \mid C_3 \qquad \text{(Cont-8)}$$

$$C_1 = C_2 \quad \Rightarrow \quad n[C_1] = n[C_2] \qquad \text{(Cont-9)}$$

Algebraic Semantics of Context (cont'd)

- Additional equalities

$$(\nu n)\ m[C] = m[(\nu n)\ C] \qquad \text{if } n \neq m \qquad \text{(Cont-5)}$$

$$(\nu n)\ \mathbf{0} = \mathbf{0} \qquad \text{(Cont-6)}$$

$$C_1 = C_2 \quad \Rightarrow \quad (\nu n)\ C_1 = (\nu n)\ C_2 \qquad \text{(Cont-7)}$$

$$C_1 = C_2 \quad \Rightarrow \quad C_1 \mid C_3 = C_2 \mid C_3 \qquad \text{(Cont-8)}$$

$$C_1 = C_2 \quad \Rightarrow \quad n[C_1] = n[C_2] \qquad \text{(Cont-9)}$$

Algebraic Semantics of Context (cont'd)

- Additional equalities

$$(\nu n)\ m[C] = m[(\nu n)\ C] \qquad \text{if } n \neq m \qquad \text{(Cont-5)}$$

$$(\nu n)\ \mathbf{0} = \mathbf{0} \qquad \text{(Cont-6)}$$

$$C_1 = C_2 \quad \Rightarrow \quad (\nu n)\ C_1 = (\nu n)\ C_2 \qquad \text{(Cont-7)}$$

$$C_1 = C_2 \quad \Rightarrow \quad C_1 \mid C_3 = C_2 \mid C_3 \qquad \text{(Cont-8)}$$

$$C_1 = C_2 \quad \Rightarrow \quad n[C_1] = n[C_2] \qquad \text{(Cont-9)}$$

Algebraic Semantics of Context (cont'd)

- Additional equalities

$$(\nu n)\ m[C] = m[(\nu n)\ C] \qquad \text{if } n \neq m \qquad \text{(Cont-5)}$$

$$(\nu n)\ \mathbf{0} = \mathbf{0} \qquad \text{(Cont-6)}$$

$$C_1 = C_2 \quad \Rightarrow \quad (\nu n)\ C_1 = (\nu n)\ C_2 \qquad \text{(Cont-7)}$$

$$C_1 = C_2 \quad \Rightarrow \quad C_1 \mid C_3 = C_2 \mid C_3 \qquad \text{(Cont-8)}$$

$$C_1 = C_2 \quad \Rightarrow \quad n[C_1] = n[C_2] \qquad \text{(Cont-9)}$$

Algebraic Semantics of Context (cont'd)

- Additional equalities

$$(\nu n)\ m[C] = m[(\nu n)\ C] \qquad \text{if } n \neq m \qquad \text{(Cont-5)}$$

$$(\nu n)\ \mathbf{0} = \mathbf{0} \qquad \text{(Cont-6)}$$

$$C_1 = C_2 \quad \Rightarrow \quad (\nu n)\ C_1 = (\nu n)\ C_2 \qquad \text{(Cont-7)}$$

$$C_1 = C_2 \quad \Rightarrow \quad C_1 \mid C_3 = C_2 \mid C_3 \qquad \text{(Cont-8)}$$

$$C_1 = C_2 \quad \Rightarrow \quad n[C_1] = n[C_2] \qquad \text{(Cont-9)}$$

Spatial Reduction Relation

The spatial reduction relation $\downdownarrows$

- allows the navigation through the hierarchical structure of context.
- is defined as follows:

$$C_1 \downdownarrows C_2 \quad \text{iff} \quad C_1 = (n[C_2] \mid C_3) \quad \text{for some name } n$$

Read: C_1 **contains** C_2.

Spatial Reduction Relation: Examples

1. $phone[\odot] \downdownarrows \odot$
2. $(phone[\odot] \mid pda[R]) \downdownarrows \odot$
3. $bob[phone[\odot] \mid Q] \downdownarrows phone[\odot]$
4. $bob[phone[\odot] \mid Q] \not\downdownarrows \odot$
5. $bob[phone[\odot] \mid Q] \downdownarrows Q$

Spatial Reduction: Reflexive and Transitive Closure

The **reflexive and transitive closure** of spatial reduction relation is denoted by $\downdownarrows^*$

- represents a finite iteration of $\downdownarrows$
- is defined as follows:

$$C_1 \downdownarrows^* C_2 \quad \text{iff} \quad C_1 = C_2 \text{ or } (C_1 \downdownarrows C_3 \text{ and } C_3 \downdownarrows^* C_2),$$

for some context C_3

Reflexive and Transitive Closure: Examples

1. $C \downdownarrows^* C$, for any context C
2. $(phone[\odot] \mid pda[R]) \downdownarrows^* \odot$
3. $bob[phone[\odot] \mid Q] \downdownarrows^* \odot$
4. $bob[phone[\odot] \mid Q] \downdownarrows^* Q$

Summary

- Context Model for CCA Processes has been presented
- Syntax and semantics of context are defined
- A **spatial modal logic** for specifying properties of contexts is given in the next Lecture.
- Formulas in this logic are called **context expressions**

SO....

- My Vision is that Informatics will empower us, only if we understand it.
- Can traditional software engineering cope?
- What we need is to organise the principles for an engineering science which will embed computing in our scientific culture