CAC, a Context-Aware Calculus

Mimosa Seminar – June 7th, 2005

Pascal Zimmer

Pascal.Zimmer@gmail.com

BRICS – University of Aarhus, Denmark

Motivation

- World of ubiquitous and pervasive computing
- Computation performed in many embedded and often invisible small devices, interconnected through a wireless network
- Different devices, different technologies, different notions of computation...
- ... still we would like to access them in a uniform way

Motivation

- We propose a new process calculus based on *context-awareness*.

Motivation

- We propose a new process calculus based on *context-awareness*.
- Definition:

  “Une femme qui est enceinte, par exemple, elle est aware qu’elle attend un enfant...” J.C. VanDamme

Motivation

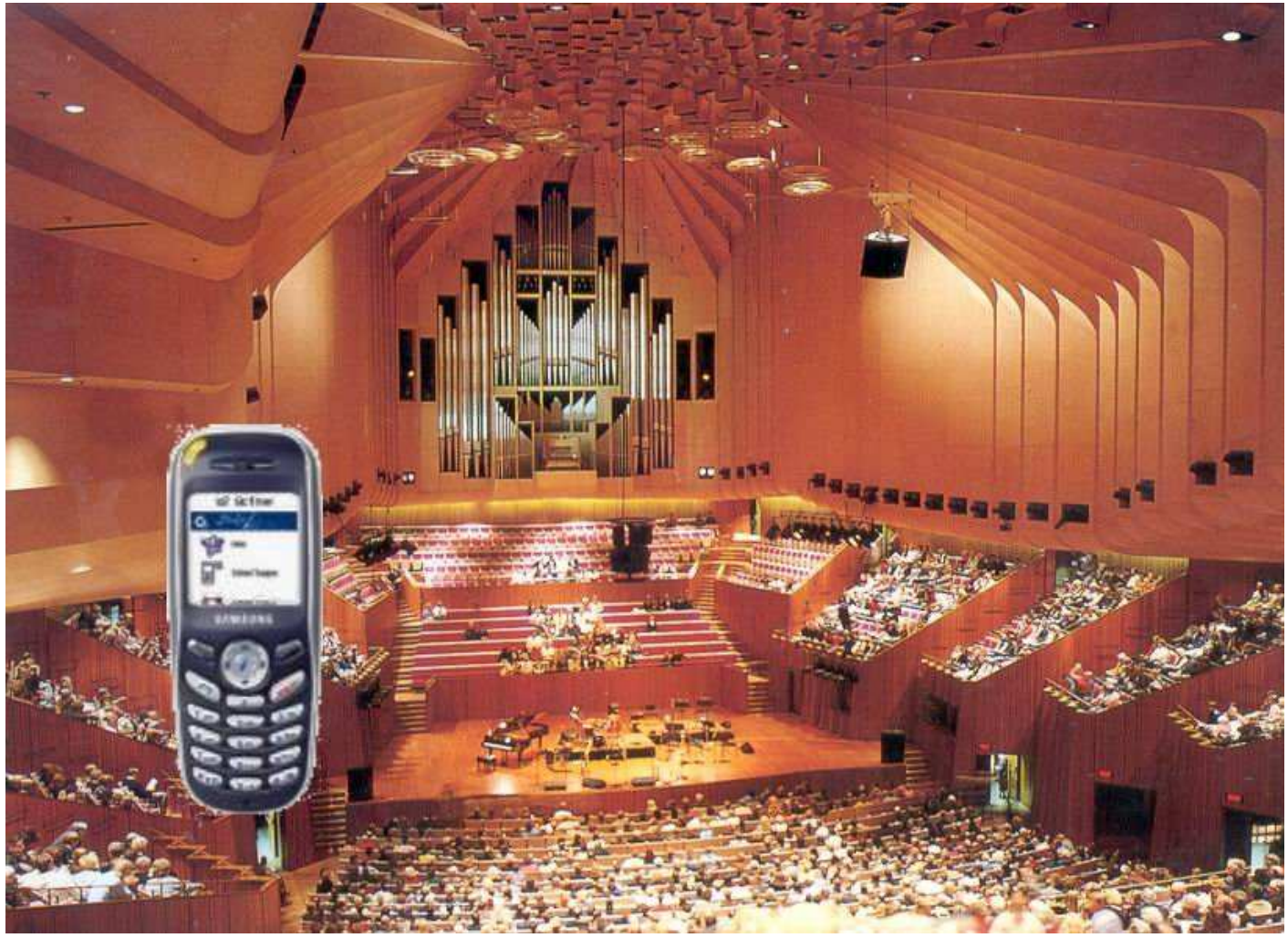
- We propose a new process calculus based on *context-awareness*.
- Definition:

  “Une femme qui est enceinte, par exemple, elle est aware qu’elle attend un enfant...” J.C. VanDamme

  Context-awareness denotes the ability for agents to react differently depending on their current environment.

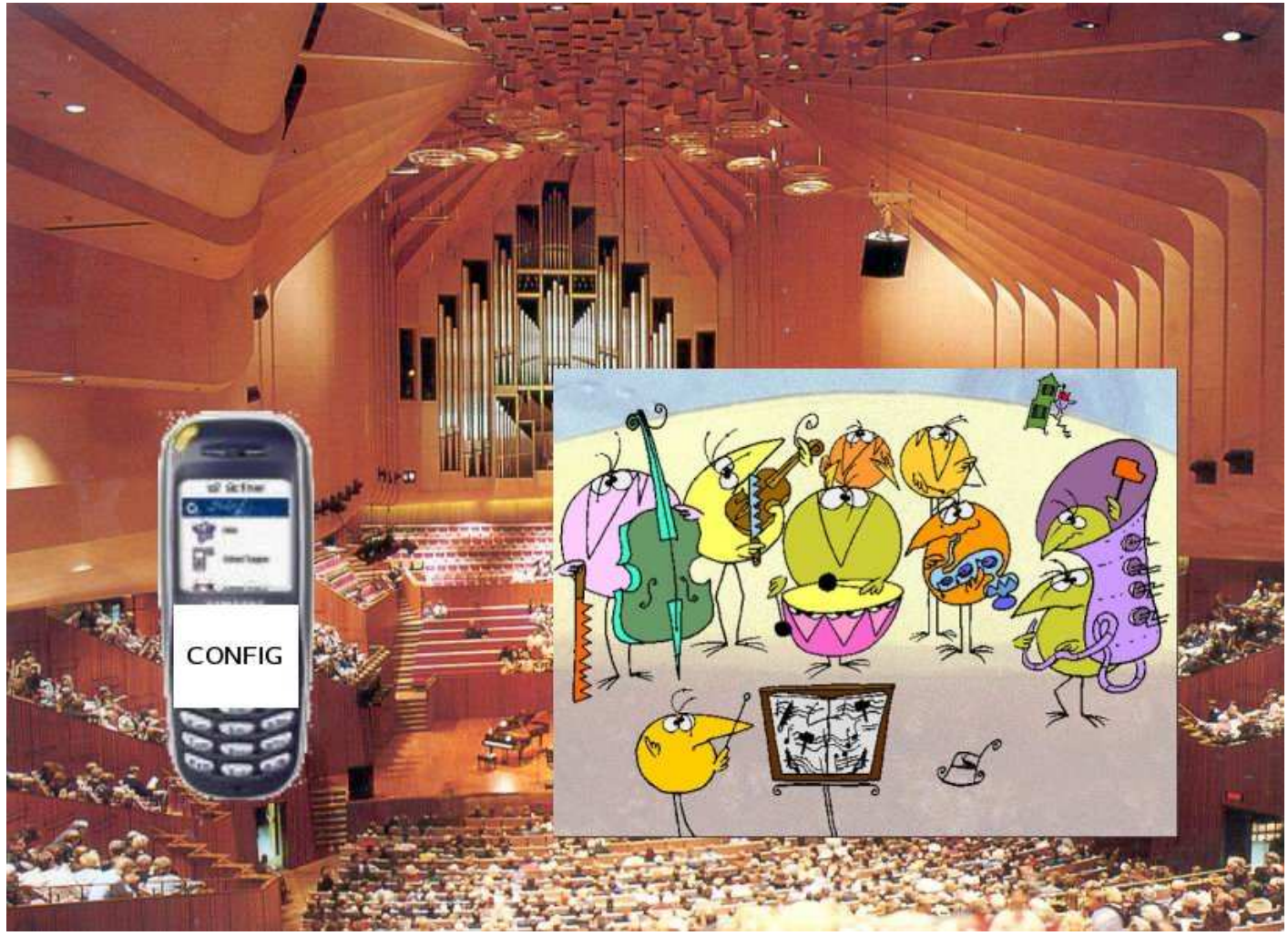
Example

Example

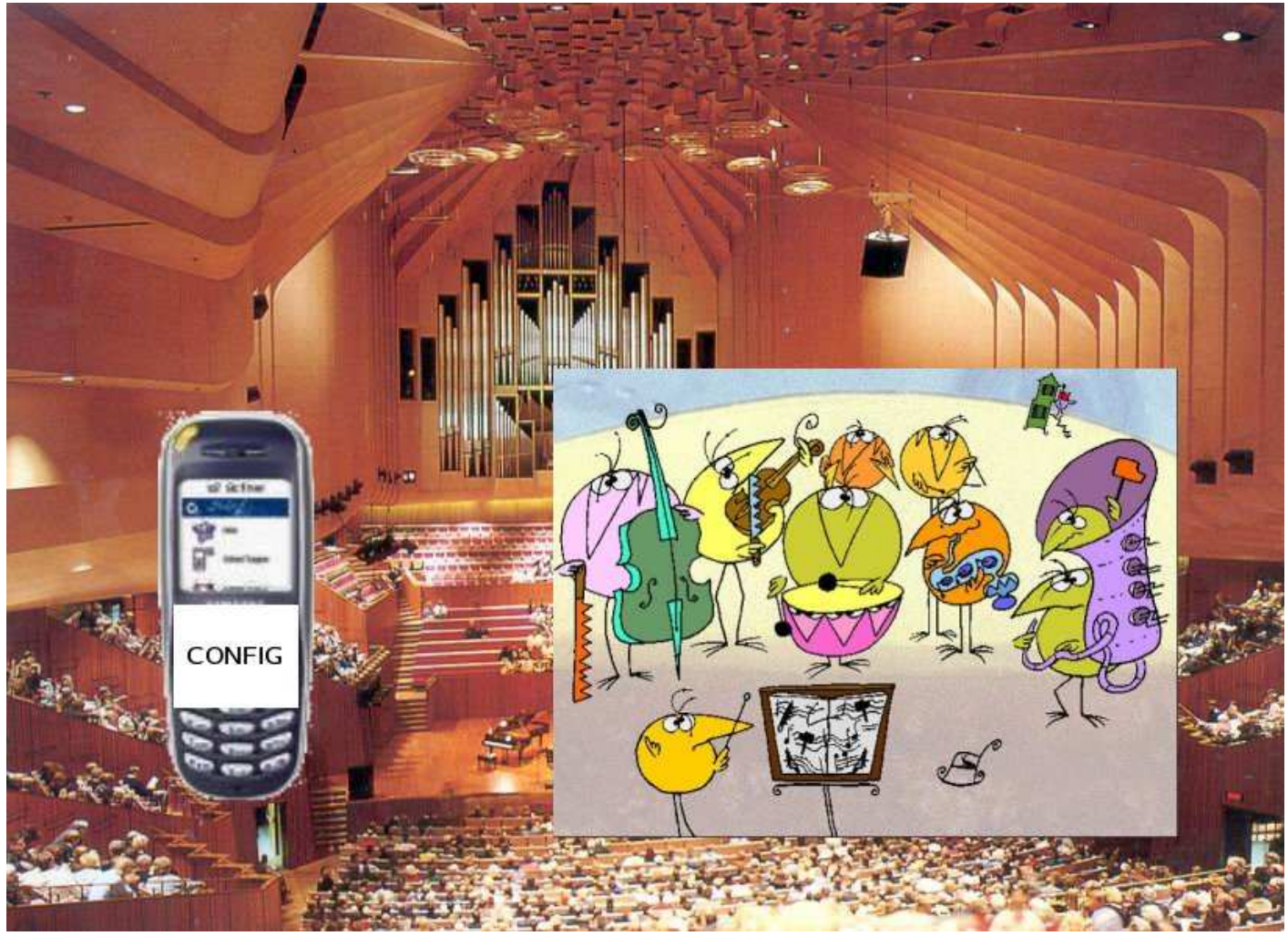


Example

Example



Example



$\Rightarrow$ environment = enclosing, surrounding and inner env.

Introduction

Our calculus should provide two essential features:

- **a notion of location:**
  Each *agent* represents a unit of computation.
  We use a hierarchical structure as in mobile ambients (Cardelli & Gordon 1998).

Introduction

Our calculus should provide two essential features:

- **a notion of location:**
  Each *agent* represents a unit of computation.
  We use a hierarchical structure as in mobile ambients (Cardelli & Gordon 1998).
- **a multi-agent synchronization:**
  Each agent sends asynchronous *atoms* to inform the environment of its capabilities. Enclosing agents provide *pattern rules* and perform synchronization, in the same way as in join-calculus (Fournet & Gonthier 1996).

Outline

- Introduction
- Basic examples
- Syntax and semantics
- Further examples
- Expressiveness and encodings
- Remarks and conclusion

Single-Agent Synchronization

An agent, willing to print some document:

$$P = app()[\ print\langle letter\rangle\ ]$$

Single-Agent Synchronization

An agent, willing to print some document:

$$P = app()[\ print\langle letter\rangle\ ]$$

Running on the following computer a:

$$a(print\langle x\rangle \triangleright send\langle x, laser\_printer\rangle)[\ \ldots\ |\ P\ |\ \ldots\ ]$$

Single-Agent Synchronization

An agent, willing to print some document:

$$P = app()[\ print\langle letter\rangle\ ]$$

Running on the following computer a:

$$a(print\langle x\rangle \triangleright send\langle x, laser\_printer\rangle)[\ \ldots\ |\ P\ |\ \ldots\ ]$$

Printing is modelled by the reduction:

$$\begin{array}{ll} & a(print\langle x\rangle \triangleright send\langle x, laser\_printer\rangle)[\ app()[print\langle letter\rangle]\ ] \\ \rightarrow & a(print\langle x\rangle \triangleright send\langle x, laser\_printer\rangle) \\ & \quad [\ app()[send\langle letter, laser\_printer\rangle]\ ] \end{array}$$

Single-Agent Synchronization

Other possible running sites:

$$b(print\langle x\rangle \triangleright send\langle x, color\_printer\rangle)[\ \ldots\ \mid P \mid\ \ldots\ ]$$

$\Rightarrow$ context-dependency for accessing some resources

Single-Agent Synchronization

Other possible running sites:

$$b(print\langle x\rangle \triangleright send\langle x, color\_printer\rangle)[\ \ldots\ \mid P \mid\ \ldots\ ]$$

$\Rightarrow$ context-dependency for accessing some resources

$$c()[\ \ldots\ \mid P \mid\ \ldots\ ]$$

$\Rightarrow$ availability of resources

Single-Agent Synchronization

Other possible running sites:

$$b(print\langle x\rangle \triangleright send\langle x, color\_printer\rangle)[\ \ldots\ \mid P \mid\ \ldots\ ]$$

$\Rightarrow$ context-dependency for accessing some resources

$$c()[\ \ldots\ \mid P \mid\ \ldots\ ]$$

$\Rightarrow$ availability of resources

$$d(print\langle x\rangle \triangleright mail\langle root, \text{“Access violation}''\rangle)[\ \ldots\ \mid P \mid\ \ldots\ ]$$

$\Rightarrow$ access control

Multi-agent Synchronization

A more complex example:

$$\begin{array}{rl} & c(x\langle\rangle \mid\mid y\langle\rangle \triangleright P \mid\mid Q)\ [\ a()[x\langle\rangle] \mid b()[y\langle\rangle]\ ] \\ \rightarrow & c(x\langle\rangle \mid\mid y\langle\rangle \triangleright P \mid\mid Q)\ [\ a()[P] \mid b()[Q]\ ] \end{array}$$

Note that P and Q replace $x\langle\rangle$ and $y\langle\rangle$ in calling sites !

Multi-agent Synchronization

With values and variables:

$$\begin{array}{rl} & c(x\langle z\rangle \mid\mid y\langle t\rangle \triangleright P \mid\mid Q) \ [\ a()[x\langle u\rangle] \mid b()[y\langle v\rangle]\ ] \\ \rightarrow & c(x\langle z\rangle \mid\mid y\langle t\rangle \triangleright P \mid\mid Q) \ [\ a()[P\sigma] \mid b()[Q\sigma]\ ] \end{array}$$

where $\sigma = \{{}^{u}/_{z}\ ,\ {}^{v}/_{t}\}$
a and b may exchange information.

Priority

Intuition: the deepest rule that matches is activated first.

Priority

Intuition: the deepest rule that matches is activated first.

Example:

$$c(x\langle\rangle \mid\mid y\langle\rangle \triangleright P \mid\mid Q)\ [\ \ a(x\langle\rangle \triangleright R)[x\langle\rangle]\ \ \mid\ \ b()[y\langle\rangle]\ \ ]$$

Syntax of CAC

Processes:

P	$::=$	$\mathbf{0}$	nil process
	$\mid$	$P \mid P'$	parallel composition
	$\mid$	$(\nu x)P$	restriction of name
	$\mid$	$x\langle\tilde{v}\rangle$	atom
	$\mid$	$a(D)[P]$	agent
	$\mid$	$def\ D\ in\ P$	new definitions
	$\mid$	$go(*, P)$	movement
$*$	$::=$	$\uparrow$	move out
	$\mid$	a	move in a

Syntax of CAC

$$D \;::=\; J_1, \ldots, J_k \qquad \text{definitions}$$

$$J \;::=\; x_1\langle \tilde{y_1} \rangle \;||\; \ldots \;||\; x_n\langle \tilde{y_n} \rangle \;\; \triangleright_{\tilde{z}} \;\; P_1 \;||\; \ldots \;||\; P_n \qquad \text{pattern}$$

$\tilde{y_1}, \ldots, \tilde{y_n}, \tilde{z}$ are all different and bound in $P_1, \ldots, P_n$.

Syntax of CAC

- Processes+agents similar to boxed ambients (Bugliesi, Castagna & Crafa 2001)
- No open (unsafe)
- No way to discard agents, but the following garbage-collection equivalence should be true:

$$(\nu a)a(D)[\mathbf{0}] \;\simeq\; \mathbf{0}$$

Semantics

Move out:

$$a(D)[\ b(D')[\ go(\uparrow, P)\ |\ Q\ ]\ |\ R\ ] \quad \rightarrow \quad a(D)[\ R\ ]\ |\ b(D')[\ P\ |\ Q\ ]$$

Move in b:

$$a(D)[\ go(b, P)\ |\ Q\ ]\ |\ b(D')[\ R\ ] \quad \rightarrow \quad b(D')[\ a(D)[\ P\ |\ Q\ ]\ |\ R\ ]$$

Semantics

Move out:

$$\overline{a(D)[\ b(D')[\ go(\uparrow, P) \mid Q\ ] \mid R\ ] \quad \rightarrow \quad a(D)[\ R\ ] \mid b(D')[\ P \mid Q\ ]}$$

Move in b:

$$\overline{a(D)[\ go(b, P) \mid Q\ ] \mid b(D')[\ R\ ] \quad \rightarrow \quad b(D')[\ a(D)[\ P \mid Q\ ] \mid R\ ]}$$

Adding new definitions:

$$\overline{a(D)[\ def\ D'\ in\ P \mid Q\ ] \quad \rightarrow \quad a(D, D')[\ P \mid Q\ ]}$$

Semantics

The big stuff:

$$\frac{\begin{array}{l} Q = \mathbf{C}[\ x_1\langle\tilde{v_1}\rangle, \ldots, x_n\langle\tilde{v_n}\rangle\ ] \\ Q \text{ reduction-free for } \{x_1, \ldots, x_n\} \\ J = x_1\langle\tilde{y_1}\rangle || \ldots || x_n\langle\tilde{y_n}\rangle \ \triangleright_{\tilde{z}}\ P_1 || \ldots || P_n \\ |\tilde{v_i}| = |\tilde{y_i}| \ \text{ for } 1 \leq i \leq n \\ \sigma = \{^{\tilde{v_i}}/_{\tilde{y_i}}\}_{1 \leq i \leq n} \\ \tilde{z} \cap fn(\mathbf{C}) = \emptyset \end{array}}{a(J, J_1, \ldots, J_k)[\ Q\ ] \ \rightarrow \ a(J, J_1, \ldots, J_k)[\ (\nu\tilde{z})\ \mathbf{C}[\ P_1\sigma, \ldots, P_n\sigma\ ]\ ]}$$

Semantics

$$
\begin{array}{rcll}
\mathbf{C} & ::= & [\,] & \text{contexts} \\
& | & \mathbf{0} & \\
& | & \mathbf{C} \mid \mathbf{C}' & \\
& | & (\nu x)\ \mathbf{C} & \\
& | & x\langle \tilde{v} \rangle & \\
& | & a(D)[\mathbf{C}] & \\
& | & def\ D\ in\ P & \\
& | & go(*, P) &
\end{array}
$$

Reduction-freedom

- Intuitively: there should not be a possible reduction in Q that would involve only a subset of the atoms $x_1\langle\tilde{v_i}\rangle, \ldots, x_n\langle\tilde{v_n}\rangle$.

Reduction-freedom

- Intuitively: there should not be a possible reduction in Q that would involve only a subset of the atoms $x_1\langle\tilde{v_i}\rangle, \ldots, x_n\langle\tilde{v_n}\rangle$.
- $msg(P) =$ the multiset of names for which P has some active atoms.
- For a pattern $J = x_1\langle\tilde{y_1}\rangle || \ldots ||x_n\langle\tilde{y_n}\rangle \ \triangleright_{\tilde{z}} \ P_1|| \ldots ||P_n$, the multiset of *pattern names* is $pn(J) = \{x_1, \ldots, x_n\}$.

Reduction-freedom

- Intuitively: there should not be a possible reduction in Q that would involve only a subset of the atoms $x_1\langle\tilde{v_i}\rangle, \ldots, x_n\langle\tilde{v_n}\rangle$.
- $msg(P) =$ the multiset of names for which P has some active atoms.
- For a pattern $J = x_1\langle\tilde{y_1}\rangle || \ldots || x_n\langle\tilde{y_n}\rangle \ \triangleright_{\tilde{z}} \ P_1 || \ldots || P_n$, the multiset of *pattern names* is $pn(J) = \{x_1, \ldots, x_n\}$.
- $Q = a(J_1, \ldots, J_k)[P]$ is reduction-free for S if the following two conditions are satisfied:
  - P is reduction-free for S
  - $(pn(J_i) \subseteq msg(P)) \Rightarrow (pn(J_i) \cap S = \emptyset)$ for $1 \leq i \leq k$ (i.e. if some pattern J_i can be triggered at this point, it is completely independent of any pattern on S)

The other cases are trivially defined by induction on Q.

Semantics

Structural congruence (standard definition):

$$\frac{P \equiv Q \qquad Q \rightarrow Q' \qquad Q' \equiv P'}{P \rightarrow P'}$$

Semantics

Structural congruence (standard definition):

$$\frac{P \equiv Q \qquad Q \to Q' \qquad Q' \equiv P'}{P \to P'}$$

$$\begin{array}{lcll} \mathbf{E} & ::= & [\,] & \text{evaluation contexts} \\ & | & \mathbf{E} \mid P & \\ & | & (\nu x)\ \mathbf{E} & \\ & | & a(D)[\mathbf{E}] & \end{array}$$

$$\frac{P \to Q}{\mathbf{E}[P] \to \mathbf{E}[Q]}$$

Contexts for implementations

Different implementations in different places:

$$a(prime\langle n\rangle \triangleright do\_sieve\langle n\rangle)[\ldots]$$

$$b(prime\langle n\rangle \triangleright lookup\_prime\_table\langle n\rangle)[\ldots]$$

Further examples

- Remote procedure call: when the local environment does not have an implementation for $prime$, but knows a location providing it
  $\Rightarrow$ forwarding of requests
- Packet routing: embed data into a packet before sending it to another location

Encoding π-calculus

Grammar for the monadic asynchronous π-calculus with replicated input:

$$P ::= \quad \mathbf{0} \;\mid\; P|P' \;\mid\; (\nu x)P \;\mid\; \bar{x}\langle v\rangle \;\mid\; x(v).P \;\mid\; !x(v).P$$

Encoding into CAC:

$$[\![\mathbf{0}]\!]_\pi = \mathbf{0}$$
$$[\![P \mid P']\!]_\pi = [\![P]\!]_\pi \mid [\![P']\!]_\pi$$
$$[\![(\nu x)\ P]\!]_\pi = (\nu x)\ [\![P]\!]_\pi$$
$$[\![\bar{x}\langle v\rangle]\!]_\pi = x\langle v\rangle$$
$$[\![!x(v).P]\!]_\pi = def\ x\langle v\rangle \triangleright [\![P]\!]_\pi\ in\ \mathbf{0}$$
$$[\![x(v).P]\!]_\pi = (\nu k)\ (def\ x\langle v\rangle \mid\mid k\langle\rangle \triangleright [\![P]\!]_\pi \mid\mid \mathbf{0}\ in\ k\langle\rangle)$$

with $k \neq x$ and $k \notin fn(P)$

Encoding λ-calculus

Grammar for the λ-calculus:

$$M ::= \; x \;\mid\; \lambda x M \;\mid\; M\ N$$

Encoding of functions and variables:

$$[\![\lambda x M]\!]_p = (\nu v)\ (\bar{p}\langle v\rangle \mid\ !v(x,r).\ [\![M]\!]_r)$$
$$[\![x]\!]_p = \bar{x}\langle p\rangle$$

Encoding of application for call-by-value λ-calculus:

$$[\![MN]\!]_p^{cbv} = (\nu q)\ ([\![M]\!]_q \mid q(v).\ (\nu r)\ ([\![N]\!]_r \mid r(w).\ (\nu x)\ (\bar{v}\langle x,p\rangle \mid\ !x(r').\ \bar{r'}\langle w\rangle)))$$

Encoding of application for call-by-name λ-calculus:

$$[\![MN]\!]_p^{cbn} = (\nu q)\ ([\![M]\!]_q \mid q(v).\ (\nu x)\ (\bar{v}\langle x,p\rangle \mid\ !x(r).\ [\![N]\!]_r))$$

Context-sensitive computation

We now delay application, using stubs:

$$[\![MN]\!]_p = (\nu m, n)\,(def\ m\langle q\rangle \triangleright [\![M]\!]_q\ ,\ n\langle q\rangle \triangleright [\![N]\!]_q\ in\ appl\langle m, n, p\rangle)$$

Context-sensitive computation

We now delay application, using stubs:

$$[\![MN]\!]_p = (\nu m, n)\ (def\ m\langle q\rangle \triangleright [\![M]\!]_q\ ,\ n\langle q\rangle \triangleright [\![N]\!]_q\ in\ appl\langle m, n, p\rangle)$$

It is the duty of the environment to provide a β-reduction strategy:

$$J_{cbv} = appl\langle m, n, p\rangle \ \triangleright\ [\![[\![m\ n]\!]_p^{cbv}]\!]_\pi$$
$$J_{cbn} = appl\langle m, n, p\rangle \ \triangleright\ [\![[\![m\ n]\!]_p^{cbn}]\!]_\pi$$

As a consequence, the process $world(J_{cbv})[\ a()[\ [\![M]\!]_p\ ]\ ]$ will reduce M following a call-by-value strategy, while its counterpart $world(J_{cbn})[\ a()[\ [\![M]\!]_p\ ]\ ]$ would follow call-by-name.

Implementation

- Sending a lot of atoms across the network can be quite slow, and cause interferences. The intended physical layout is the following:

$$world()[\ site_a(\ldots)[\ldots] \mid site_b(\ldots)[\ldots] \mid \ldots]$$

- With such a layout, we are sure that every atom can be sent only locally, and there is no communication on the wide-area network, except for agent movement.
- This still means we need a synchronizing scheduler on every local physical location, taking care of all local agents.

Implementation

- We can also define a more general model, where it is not necessary that physical locations appear at toplevel. All we need to do is to syntactically distinguish those locations with a specific notation such as: $a(D)[|P|]$.

Detecting Absence

- It is easy to detect the presence of a subagent...
- ...but it is much more difficult to detect an absence

Detecting Absence

- It is easy to detect the presence of a subagent...
- ...but it is much more difficult to detect an absence
- One possible way to achieve such a feature might be to add terms $\neg x$ in pattern rules, with the meaning that there should not be any atom on x available for the rule to match.
  For example, in the following process:

$$a(x\langle\rangle \mid\mid \neg y \triangleright P\ ,\ x\langle\rangle \mid\mid \neg z \triangleright Q)\ [\ x\langle\rangle \mid y\langle\rangle\ ]$$

  the former pattern cannot be activated, while the latter can be.

Conclusion and Future Work

Conclusion:

- A new process calculus for context-awareness, based on a powerful multi-agent synchronization
- Suitable for different forms of context dependencies, including computation itself

Future work:

- Expressiveness
- Behavioural theory
- Trusting contents

Remote Procedure Call (RPC)

$$B = b()[\ (\nu k)\ def\ k\langle result\rangle \triangleright P\ in\ prime\langle n, k\rangle\ ]$$

$$A = a(\ prime\langle n, k\rangle\ ||\ local\langle\rangle \triangleright_{k'} k'\langle\rangle\ ||\ (Q \mid local\langle\rangle)\ )\ [\ local\langle\rangle \mid B\ ]$$

$$Q = (\nu k'')\ def\ k'\langle\rangle\ ||\ k''\langle x\rangle \triangleright k\langle x\rangle\ ||\ \mathbf{0}\ in\ R$$

$$R = (\nu p)\ p()[\ go(\uparrow.server, S)\ ]$$

$$S = (\nu k''')\ def\ k'''\langle x\rangle \triangleright go(\uparrow.a,\ k''\langle x\rangle)\ in\ prime\langle n, k'''\rangle$$

$$A \mid server(prime\langle n, k\rangle \triangleright k\langle ...\rangle)[\mathbf{0}]$$

Packet routing

$$
\begin{aligned}
P &= (\nu packet)\ packet(J)\ [\ packet\_ready\langle packet\rangle \mid wait\_dest\langle\rangle\ ] \\
J &= move\_to\langle y\rangle \parallel wait\_dest\langle\rangle \ \triangleright \\
&\qquad\qquad \mathbf{0} \parallel go(\uparrow.y,\ def\ is\_arrived\langle k\rangle \ \triangleright\ go(\uparrow, k\langle\rangle)\ in\ \mathbf{0}) \\
Q &= data()[\ data\_ready\langle\rangle \mid move\_to\langle b\rangle \mid is\_arrived\langle k\rangle\ ] \\
S &= a(packet\_ready\langle x\rangle \parallel data\_ready\langle\rangle \ \triangleright\ \mathbf{0} \parallel go(x, \mathbf{0}))[\ P \mid Q\ ]
\end{aligned}
$$

$$
\begin{aligned}
S \ \rightarrow^* \ & a(\ldots)[\mathbf{0}] \\
& \mid\ b(\ldots)[\ (\nu packet)packet(J)[\mathbf{0}]\ \mid\ data()[k\langle\rangle]\ \mid\ \ldots\ ]
\end{aligned}
$$

Citations

- *“Une femme qui est enceinte, par exemple, elle est aware qu’elle attend un enfant...”* J.C. VanDamme
- *“Tu regardes à l’intérieur de toi et tu deviens aware of your own body !”* J.C. VanDamme
- *“Ah non mais attention quand je parle de l’enveloppe tu vois, je parle pas de l’enveloppe que tu envoies par la poste. Je parle de l’enveloppe que tu vois. Celle qui enveloppe tout. Les paquets de biscuits, les sachets de cocaïne, ton esprit, etc ... Non, l’enveloppe c’est vraiment global.. Mais uniquement liée au spirit généralement. Oui alors un biscuit tu me diras ça n’a pas de spirit, c’est juste un biscuit. Mais avant, c’était du lait, des oeufs. Et dans les oeufs, il y a la vie potentielle... Le potential life dans une coquille, une enveloppe ... qui elle même était contenu dans la poule . eh oui... Non vraiment tout ça c’est une question d’awareness...”* J.C. Van Damme

Citations

- *“Les plantes par exemple, qui n’ont pas de mains, et pas d’oreilles,elles sentent les choses, les vibrations, elles sont plus aware que les autres species”* J.C. VanDamme
- *“Quand tu joues au Go.. faut être aware. Si t’es pas aware, tes pierres sont mortes, et toi avec.”* J.C VanDamme
- *“Y a des gens qui n’ont pas réussi parce qu’ils ne sont pas aware, ils ne sont pas ”au courant”. Ils ne sont pas à l’attention de savoir qu’ils existent. Les pauvres, ils savent pas. Il faut réveiller les gens. C’est-à-dire qu’y a des gens qui font leur travail, qui font leurs études, ils ont un diplôme, ils sont au contact tout ça. Tu as un rhume et tu fais toujours ”snif”. Faut que tu te mouches. Tu veux un mouchoir ? Alors y a des gens comme ça qui ne sont pas aware. Moi je suis aware tu vois, c’est un exemple, je suis aware.”* J.C. Van Damme