HAND-OUT

for

A Short Course
on
Mobile Processes

University of Lisbon

February, 1997

K. Honda.

Notes:

This is a post-lecture handout for a short course on mobile processes given at University of Mobile Processes on February 1997. 5 lectures each 3 hours, as well as 2 seminars, each 1 hour, were given during the period. Not all materials treated in the lectures are covered in this handout. In particular the latter part of the lecture on types is lacking, which will be supplemented later. I thank all the audience for attending the talks, and Vasco Vasconcelos for organising the course.

K.H. (Feb 27, 1997.)

Organisation of the Handout

- Outline of Lectures (N.B. the real lectures did not precisely follow these plans.)
- Basics of Mobile Processes
- Embedding Calculi and Languages (Only λ-calculus is treated in detail).
- Introduction to Asynchronous Calculus and Combinators.
- Types for Mobile Processes
- Symmetries on Processes.

A Short Course on Mobile Processes

80 minutes – 20 minutes ~ 80 minutes.

Prerequisites

Basic knowledge on formal systems and algebra
Intuitive Understanding on concurrent computation.

Plan of Lecture

2/12

- **General Introduction**
  - Computation by interaction
  - context of π-calculus
- **Syntax**
  - Syntax of Core Calculus
  - Extensions and Variants.
- **Structural Rules**
  - Definition
  - Examples
  - Discussion on Structural Rules
- **Reduction**
  - Definition
  - Examples
    - Scope Extrusion
    - Non-determinism
    - Basic Agents
- **Dynamics of Name Passing** //
- **Behaviour of Mobile Processes**
  - Input, output and bound output.
- **Labelled Transition Relation**
- **Strong Bisimilarity**
  - Definition
  - Examples
  - Laws of Equations
- **Weak Bisimilarity**
  - Definition
  - Examples
- **Application of Bisimulations**
  - Multiple Name Passing
  - Branching / Selection
  - Recursion
  - First Order Functions
  - Stateful Agents
  - Prelude to "Functions as Processes" //

2/13

- Embedding Calculi and Languages

- Functions as Processes

  Lazy λ-calculus

  Encoding

  Basic Syntactic Properties

  Observability and Fullabstraction

  Types for λ-like Agents (*)

- Parallel Imperative Language.

  Variables

  Control Constructs

  Procedures (*)

  Operational Correspondence

  Observability

  Turing Machine in π (*)

  Basic data structures

  Dynamics //

- Introduction to Asynchronous Calculus and Combinators

- Reduction of Syntax

  Asynchronous Message and Actors

- Basic "Asynchronous" agents

  Identity Receptors

  Link Agents and others

- Behavioral Equivalences on Asynchronous Mobile Processes.

  Congruence Results

  Asynchronous Observables

  Agents and Observability (*)

- Embedding Synchrony in Asynchrony

  Encoding

  Proof Outline

  Extensions and Refinement. //

(*) optional.

2/17

- Combinators in Concurrency Setting.

  - Context

  - Atoms for Interaction.

- Seven Atoms

- Eliminating Prefix (1)

  - Inductive elimination.

- Eliminating Prefix (2)

  - Pushing out bound names

  - Some equivalence laws.

- Representability Theorem

  - Theorem and proof outline.

- Eliminating Replication.

  - Variants of replication (*)

  - Basic ideas of decomposition.

- Representability Theorem, stronger version.

- Further Directions. /

(*) Optional.

2/17

- Introduction to Universality theorem.
- Generalised Concurrent Combinators
  - Definition
  - Examples
- Abstract Notion of Embeddings
  - Definition
  - Basic Properties
- Examples of Embeddings (*)
  - Computability
  - Shynchrony in Asyhnchrony
  - Cases for summations.
- Universality Theorem.
  - Basic constructions
  - Proof outline
  - Discussions and Comparisons.
- Remaining Issues.

(*) optional.

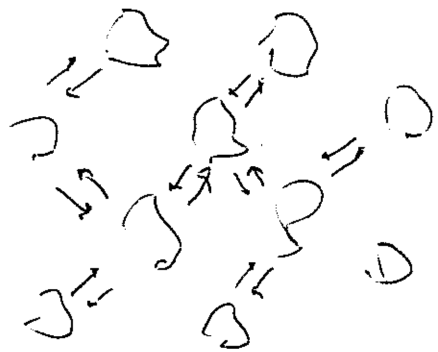
2/18 Types for Mobile Processes

- Backgrounds
  - Types for functions
  - Behaviroul Types
  - Programatics
- Polyadic π-calculus
  - Syntax
  - Bisimilarity
- Sorting
  - Definition
  - Simple Example
- Typing System for Sorting
  - Definition
  - Syntactic Properties
  - Examples
- Sorting for λ-agents
- Semantics of Sorting
- Refinament of Sorting (1)
  - Linearity
  - Replication (Server-Client Types)
  - I/O Types
- Refinament of Sorting (2)
  - Type System for Linearity
  - Inpaction Behaviroul Equality
- Beyond Sorting

Lecture I

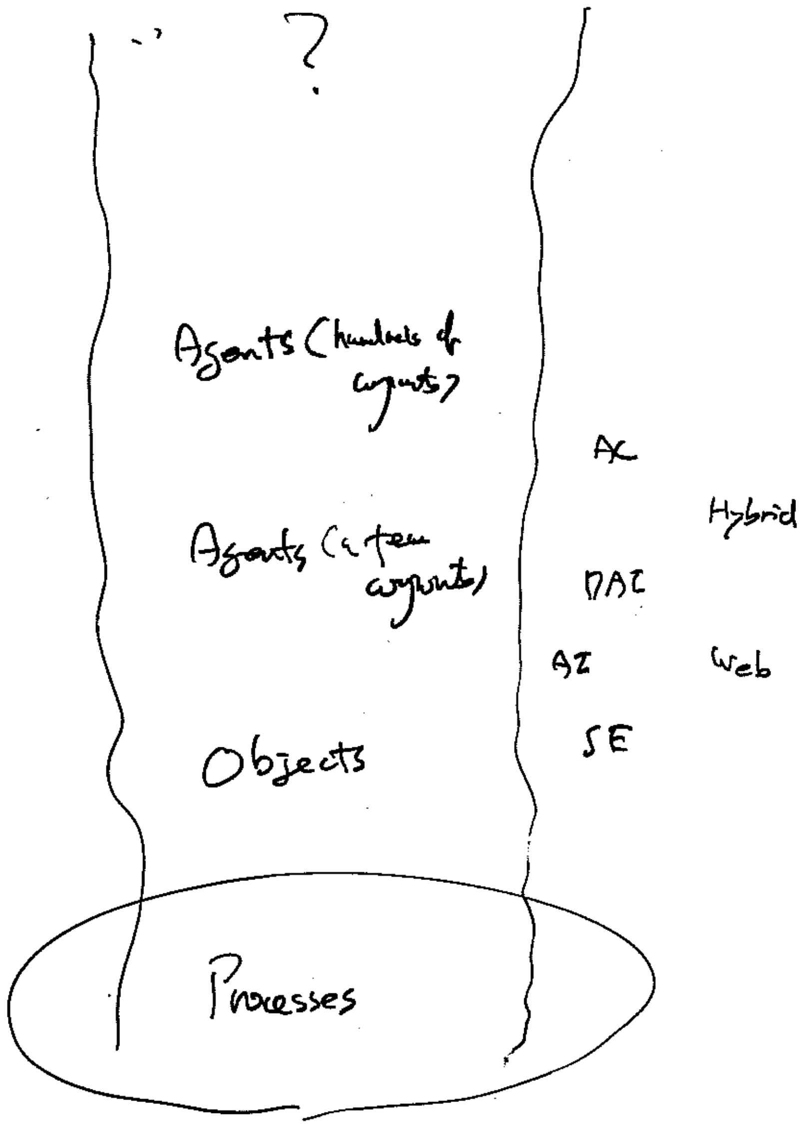
Basics of Mobile Processes

Abstraction as Interaction



- Decomposition of the "whole" into
  - components
  - their interaction.
- Examples
  - physics (particles)
  - theatre
  - object model (software eng.)
  - agent model (AI)
  - biology

Computation as Interaction.



Context of π-Calculus (1)

- Denotational Semantics [Scott-Strachey 70]
  - compositional approach to semantics, influenced by λ-calculus.
  - Mathematical foundations.
- Difficulty in treating Interfering Concurrency.

  $x := 1 \;||\; x := 2;\; x := 2x$
- CCS [Milner 80]
  - Return to syntax.
  - Calculus purely based on "interaction" rather than "function"

    $$a.P + Q \mid \bar{a}.R + S \rightarrow P \mid R$$ (communication " synchronization)
  - Basic theory of "behavioural equivalences": when can we say two processes are (essentially) the same?

Context of π-calculus (2)

- Issues in CCS
  - Not extendable to programming languages
  - Encoding is hard.
  - Summation is doubtful.
- π-calculus [Engberg-Nielsen 86; Milner-Parrow-Walker 88]
  - Another syntax for interaction, with a new equation:

    Communication = (Synchronisation +) Name Passing

    Thus we get:

    $$a(x).P \mid \bar{a}b.Q \rightarrow P\{b/x\} \mid Q$$
  - Solving the issues of CCS.
    - extendable.
    - encoding of data st.
    - simple, powerful op.
  - Links to other computational models.

    ..... We still do not have Scott Models...

π-Calculus.

- Syntax (terms) P, Q, ..
- Structural rules. $\equiv$
- Reduction $\rightarrow$
- Labelled Transition. $\xrightarrow{l}$ $(\xrightarrow{\tau} = \rightarrow)$
- Bisimilarities $\sim$, $\approx$

Basic constructs $a(x_1..x_n).P$ $\bar{a}\langle v_1..v_n\rangle.P$

$a:[(x).P_1] \& [(x).P_2]$ $\bar{a}:\mathrm{inl}[\langle e\rangle P]$

$\bar{a}:\mathrm{inr}[\langle e\rangle P]$

let $X(\tilde{x}) = P$ in Q.

Terms

- Throughout the lecture we fix the set of names $\mathcal{N}$, ranged over by $a, b, c, \ldots$ or $x, y, z, \ldots$

- Terms:

$$P ::= a x.P \mid \bar{a} b.P \mid P|Q \mid (\nu a)P \mid \emptyset \mid !a x.P$$

- Binders:

$$a x.P \qquad (\nu a)P \qquad !a x.P$$

Structural Equality

- $\equiv$ is the smallest congruence relation closed under:
  - $P \equiv_\alpha Q \Rightarrow P \equiv Q$
  - $P|\emptyset \equiv P \quad P|Q \equiv Q|P \quad (P|Q)|R \equiv P|(Q|R)$
  - $(\nu a)\emptyset \equiv \emptyset \quad (\nu a)(\nu a)P \equiv (\nu a)P \quad (\nu a)(\nu b)P \equiv (\nu b)(\nu a)P$
  - $(\nu a)P \mid Q \equiv (\nu a)(P|Q) \quad (a \notin FN(Q))$

    $((\nu a)P)|Q$

- $\equiv$ turns terms into certain (hyper) graphs.

Reduction.

- The <u>reduction relation</u> $\rightarrow$ is the smallest relation generated by:

(COM) $ax.P \mid \bar{a}b.Q \rightarrow P[b/x] \mid Q$

(REP) $!ax.P \mid \bar{a}b.Q \rightarrow P[b/x] \mid Q \mid !ax.P$

(PAR) $P \rightarrow Q \Rightarrow P|R \rightarrow Q|R$

(RES) $P \rightarrow Q \Rightarrow (\nu s)P \rightarrow (\nu s)Q$

(STR) $P \equiv P' \; P' \rightarrow Q' \; Q' \equiv Q \Rightarrow P \rightarrow Q$.

LTS and Bisimulations.

- Labels.

$$\ell ::= \bar{a}b \mid ab \mid \bar{a}(b) \mid \underbrace{a(b)}_{\text{optional.}} \mid \tau$$

- Given $\xrightarrow{\ell}$,

<u>Def</u> $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ is a strong $\boxed{\text{weak}}$ bisimulation

when $P \mathcal{R} Q$ and $P \xrightarrow{\ell} P'$ s.t. $FN(Q) \cap BN(\ell) = \emptyset$

we have, for some Q', $Q \xrightarrow{\ell} Q'$ $\boxed{Q \overset{\hat{\ell}}{\Rightarrow} Q'}$ and $P' \mathcal{R} Q'$.

Similarly, exchanging P and Q.

$\bar{a}(b)$

$a(c)$

$$\overset{\hat{\ell}}{\Longrightarrow} \overset{\text{def}}{=} \begin{cases} \xrightarrow{\tau}{}^{*} & \text{if } \ell = \tau \\ \xrightarrow{\tau}{}^{*} \xrightarrow{\ell} \xrightarrow{\tau}{}^{*} & \text{if } \ell \neq \tau. \end{cases}$$

Labeled Transition Relation.

[COMMON]

✓ (IN) $a x. P \xrightarrow{ab} P[b/x]$

✓ (OUT) $\bar{a} v. P \xrightarrow{\bar{a}v} P$

side condition? $a \neq b$

✓ (BOUT) $P \xrightarrow{\bar{a}b} P' \Rightarrow (\nu b) P \xrightarrow{\bar{a}(b)} P'$

✓ (PAR) $P \xrightarrow{\ell} P' \Rightarrow P | R \xrightarrow{\ell} P' | R$ $\quad BN(\ell) \cap FN(R) = \emptyset$

✓ (RES) $P \xrightarrow{\ell} P' \Rightarrow (\nu a) P \xrightarrow{\ell} (\nu a) P'$ $\quad \cancel{a \notin BN(\ell)}$ or $a \notin N(\ell)$!

(REP) $!a x. P \xrightarrow{ab} !a x. P \mid P[b/x]$

[CONSERVATIVE]

✓ (COM) $P \xrightarrow{ab} P' \quad Q \xrightarrow{\bar{a}b} Q' \Rightarrow P|Q \xrightarrow{\tau} P'|Q'$

✓ (BCOM)* $P \xrightarrow{ab} P' \quad Q \xrightarrow{\bar{a}(b)} Q' \Rightarrow P|Q \xrightarrow{\tau} (\nu b)(P'|Q')$

(α) $P \equiv_\alpha P' \quad P' \xrightarrow{\ell} Q' \quad Q' \equiv_\alpha Q \Rightarrow P \xrightarrow{\ell} Q$.

(PAR) symmetric of ~~PAR~~ rules involving |

$*\, b \notin FN(P)$.

[EASY] in TCS

(COM) $a x. P \mid \bar{a} b. Q \xrightarrow{\tau} P[b/x] \mid Q$

RCOM (COM2) $!a x. P \mid \bar{a} b. Q \xrightarrow{\tau} !a x. P \mid P[b/x] \mid Q$

(STR) $P \equiv P' \quad P' \xrightarrow{\ell} Q' \quad Q' \equiv Q \Rightarrow P \xrightarrow{\ell} Q$.

Prop.

(1) $P \xrightarrow{\ell}_{cons} P' \Rightarrow P \xrightarrow{\ell}_{EASY} P'$

(2) $P \xrightarrow{\ell}_{EASY} Q \Rightarrow P \xrightarrow{\ell}_{cons} P' \equiv Q$.

(∵) They give the same bisimilarities.

∵ $P \equiv P' \wedge P \xrightarrow{\ell}_{cons} Q \Rightarrow \exists Q'.\ P' \xrightarrow{\ell}_{cons} Q' \wedge Q' \equiv Q$.

[symmetric]

(BIN) $ax.P \xrightarrow{a(x)} P$

(BCOM) $P \xrightarrow{a(x)} P' \quad Q \xrightarrow{\bar{a}(x)} Q' \Rightarrow P|Q \xrightarrow{\tau} (\nu x)(P'|Q')$

<u>Prop</u>

Again we have the same bisimulations

!: Because of the side condition for bound actions.

Bisimulations for:

* $P \oplus Q \sim Q \oplus P$

$$\{(P \oplus Q,\ Q \oplus P)\} \cup \{(P|Q,\ Q|P)\} \cup \equiv$$

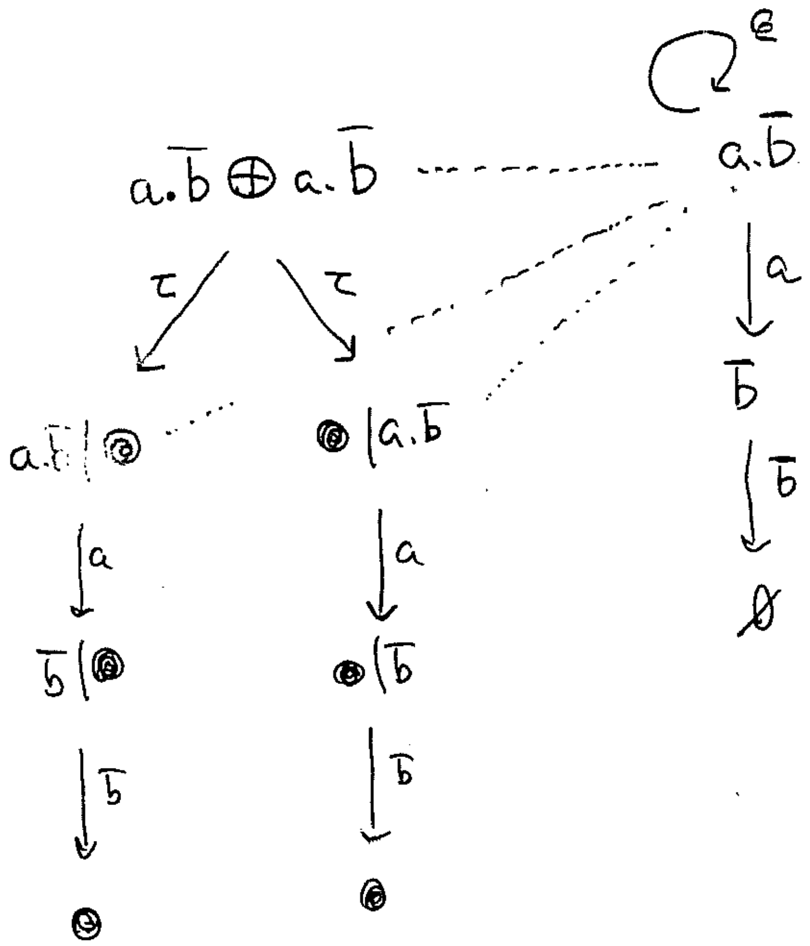
* $P \oplus P \approx P$

$$\{(P \oplus P,\ P)\} \cup \overset{GC}{\equiv} \cup \equiv$$

* $P \xrightarrow{\beta} Q \Rightarrow P \approx Q$ $\boxed{\longrightarrow_\beta \subseteq \approx}$

$$\{(P, Q) \mid P \xrightarrow{\beta} Q\} \cup \equiv$$

$$\begin{tikzcd} & C[\![(\nu c)(P_1\langle c\rangle \mid P_2\langle c\rangle)]\!] \quad P \arrow[dl, "\ell"'] \arrow[dr, "\beta"] & \\ Q' \not\equiv Q \arrow[dr, dashed, "\beta"'] & & Q \arrow[dl, dashed, "\ell"] \\ & R & \end{tikzcd}$$



$$(P \mid 0,\ P) \in \stackrel{GC}{\equiv}$$

Replication Law.

Prop If c occurs only as negative subjects in R_1, R_2 and P,

$$(\nu c)(!cx.P \mid R_1 \mid R_2) \stackrel{def}{\equiv} Q_1 \xrightarrow{\ell} Q_1'$$

$$\sim (\nu c)(!cx.P \mid R_1) \mid (\nu c)(!cx.P \mid R_2) \stackrel{def}{\equiv} Q_2$$

Proof: Show the following is a bisimulation:

$$\mathcal{R} = \{(Q_1', Q_2') \mid Q_1' \equiv Q_1,\ Q_2' \equiv Q_2,\ Q_1, Q_2 \text{ as above}\}$$

taking care of the invariance: c never becomes extruded nor become positive.

Counterexample.

⇒ If c occurs positively:

$$(\nu c)(!cx.\bar{e}x \mid \bar{c}v \mid cx.\bar{f}x) \xrightarrow{\tau} \xrightarrow{\bar{f}v}$$

$$\not\approx \quad (\nu c)(!cx.\bar{e}x \mid \bar{c}v) \mid$$

$$(\nu c)(!cx.\bar{e}x \mid cx.\bar{f}x)$$

⇒ If c occurs as object:

$$(\nu c)(!cx.\bar{e}x \mid \bar{c}v \mid \bar{l}c \mid ly.yx.\bar{f}x)$$

$$\longrightarrow (\nu c)(!cx.\bar{e}x \mid \bar{c}v \mid cx.\bar{f}x)$$

Pointedness.

Def a occurs active in P when:

$$P \equiv (\nu \tilde{c})(P_0 \mid R) \qquad P_0 \begin{cases} \left.\begin{array}{l} ax.P' \\ !ax.P' \end{array}\right\} \text{Positive} \\ \left.\bar{a}b.P'\right\} \text{Negative.} \end{cases}$$

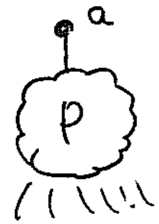
and $a \notin \{\tilde{c}\}$.

Def P is a-pointed when:

(1) There is a unique active occurrence of a in P

(2) No other names occur active in P

(3) $P \not\to$.



Behaviour of Agents (1)

$$
\begin{array}{cc}
a(x_1 x_2).P & \overline{a}[v_1 v_2].Q \\
\equiv & \equiv \\
az.zx_1.zx_2.P & (\nu z)(\overline{a}z.\overline{z}v_1.\overline{z}v_2.Q) \\
\downarrow a(z) & \downarrow \overline{a}(z) \\
\downarrow zv_1 & \downarrow \overline{z}v_1 \\
\downarrow zv_2 & \downarrow \overline{z}v_2 \\
P[v_1 v_2 / x_1 x_2] & Q
\end{array}
$$

Behaviour of Agents (2)

$$
(\nu c_1 c_2)\left(\overline{a}[c_1 c_2].\left(c_1 x.P \mid c_2 y.Q\right)\right)
$$

$$
\begin{array}{c}
\downarrow \overline{a}(z) \\
\downarrow \overline{z}(c_1) \\
\downarrow \overline{z}(c_2)
\end{array}
$$

$$
\begin{array}{ccc}
\swarrow c_1 v & & \searrow c_2 w \\
\left(\text{Behaves as } P[v/x]\right) & & \left(\text{Behaves as } Q[w/y]\right)
\end{array}
$$

Lecture II

Embedding Calculi and Languages.

Embedding Calculi and Machine.

- Calculi
  - λ-calculus. [MPW89, Milner90, ...]
  - Proof Nets [Abramsky 91]

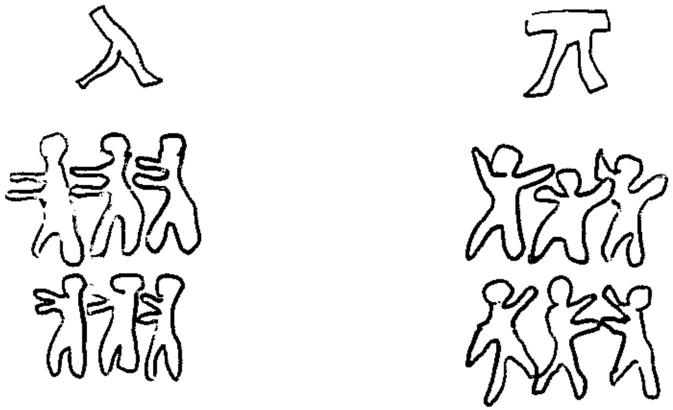
Representing Term Rewriting by Name Passing + Replication.

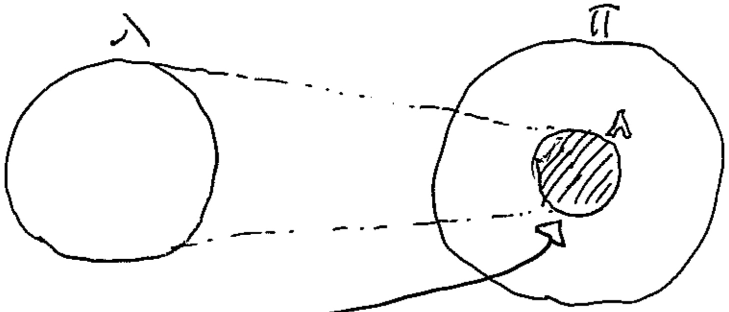
- Programming Languages
  - Procedural Languages [Walker90]
  - Functional Languages
  - Logic Languages.

- Machines
  - Turing Machines
  - SECD and other abstract machines.
  - various automata.

Prologue to λ-Calculus Embedding.

- Functions as processes: [Milner 90] [Milner 92]





✱ what is the interactive behaviour, or Rules of Interaction, of λ-agents?

Reducing Higher-Order Interaction to Name Passing.

$$\begin{array}{ccc} (\lambda x.M) & \cdot & N \\ \downarrow & & \downarrow \\ M[N/x] & & \ast \end{array}$$

becomes;

$$\begin{array}{ccc} [\![\lambda x.M]\!] & \cdot & [\![N]\!] \\ \downarrow & & \downarrow \\ P & & Q \\ \downarrow & & \downarrow \\ P' & & Q' \\ \downarrow & & \downarrow \\ P'' & & Q'' \end{array}$$

Encoding and its Behaviour (1)

- Encoding. [Milner 90].

$$[\![x]\!]u = \overline{x}u$$

$$[\![\lambda x.M]\!]u = u(xz).[\![M]\!]z$$

$$[\![MN]\!]u = (\nu c_1 c_2)([\![M]\!]c_1 \mid \overline{c_1}[c_2 u].[\![c_2 = N]\!])$$

$$\equiv (\nu c_1)([\![M]\!]c_1 \mid \underbrace{(\nu c_2)\overline{c_1}[c_2 u].[\![c_2 := N]\!]}_{Arg[c_1 u N]})$$

where $[\![x = N]\!] \stackrel{\text{def}}{=} !x(z).[\![N]\!]z$

- Dynamics. $Arg(c,u,N) = (\nu x)(\overline{c}[xu].[\![x=N]\!])$

(APP) $[\![\lambda x.M]\!]c \mid Arg(c,u,N)$
$\quad \to^+ (\nu x)([\![M]\!]u \mid [\![x := N]\!])$

(Fetch) $[\![x\tilde{N}]\!]u \mid [\![x = M]\!]$
$\quad \to^+ [\![M\tilde{N}]\!]u \mid [\![x = M]\!]$

Encoding and its Behaviour (2)

- Notice, with N_0 either a variable or an abstraction:

$$[\![N_0N_1 \dots N_n]\!]u$$
$$\stackrel{\text{def}}{=} (\nu\tilde{c})([\![N_0]\!]c_0 \mid Arg(c_0,c_1,N_1) \mid Arg(c_1,c_2,N_2) \mid \cdots \mid Arg(c_{n-1},u,N_n))$$

- This shows the only possible dynamics from the term of form:

$$(\nu\tilde{x})([\![N_0N_1\dots N_n]\!]u \mid [\![x_1=M_1]\!] \mid [\![x_2=M_2]\!] \mid \dots \mid [\![x_m=M_m]\!])$$

with $n \geq 0$, $m \geq 0$ is (APP) and (FETCH), which again reduces to the above form.

(It terminates when $n=0$, N_0 is an abstraction.) only for closed terms

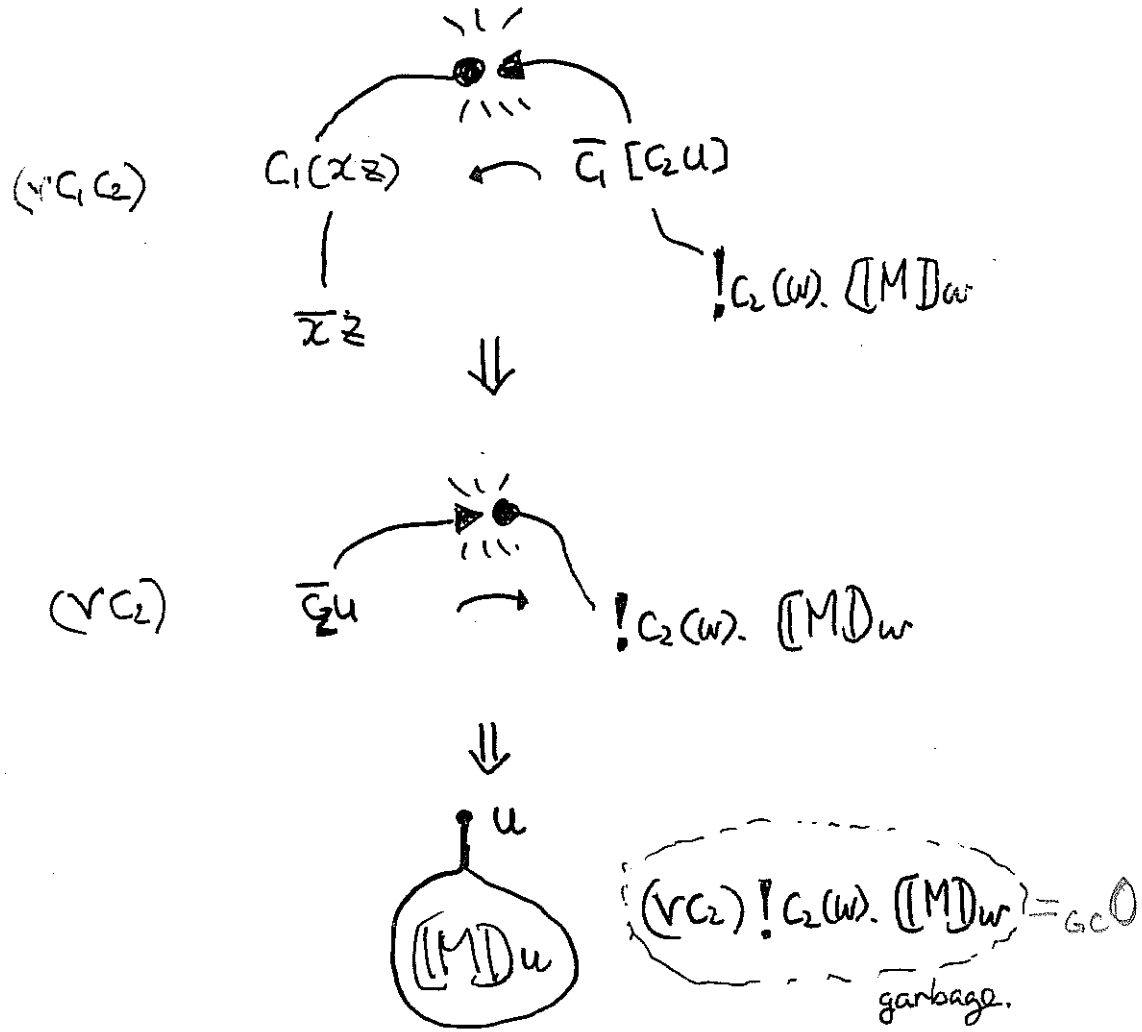
It may also terminate when $(\nu\tilde{x})([\![y\tilde{N}]\!] \mid [\![\tilde{x}=\tilde{M}]\!])$ and $y \notin \{\tilde{x}\}$

$IM \to M$ $\quad (I = \lambda x.x)$

$IM \to M$

$\llbracket \lambda x.x \rrbracket c_1 = c_1(xz).\overline{x}z$

$\llbracket IM \rrbracket u = (\nu c_1 c_2)(\llbracket \lambda x.x \rrbracket c_1 \mid \overline{c_1}[c_2 u].!c_2(w).\llbracket M \rrbracket w)$



Basic Syntactic Properties (1)

- By the preceding discussions we immediately know:

Prop $\llbracket M \rrbracket u \to^n P_1$ and $\llbracket M \rrbracket u \to^n P_2$ implies $P_1 \equiv P_2$.

- We also note:

Prop

(i) $\llbracket M \rrbracket u$, $Arg(c_1, c_2, N)$, $\llbracket x = N \rrbracket$ are always pointed.

(ii) $\llbracket M \rrbracket u \to^n P$ and $P \to Q$ implies $P \xrightarrow{\beta} Q$.

Basic Syntactic Properties (2)

Lemma

If c occurs only as displayed below (not bound within C):

$$(\nu c)\left(C[\bar{c}b_1][\bar{c}b_2]\cdots[\bar{c}b_n] \mid\ !cx.P\right)$$

$$\approx_{\Downarrow} C[P\{b_1/x\}][P\{b_2/x\}]\cdots[P\{b_n/x\}]$$

where $P \approx_{\Downarrow} Q \overset{\mathrm{def}}{\iff} P \approx Q \wedge (P\Uparrow \iff Q\Uparrow)$

$\forall n \in \omega.\ Q \to^{n}$

Remark: If c also occurs in P as negative subjects, the right hand side becomes:

$$C[(\nu c)(P\{b_1/x\} \mid\ !cx.P)]\cdots[(\nu c)(P\{b_n/x\} \mid\ !cx.P)]$$

We can easily see the above lemma is a special case of this latter one.

Basic Syntactic Properties (3)

Proof of Lemma (outline):

Let $\mathcal{R}$ be the relation between two processes as given above, taking them modulo $\equiv$. That $\mathcal{R}$ is a weak bisimulation is easy to prove.

To show $P\mathcal{R}Q$ implies $(P\Uparrow \iff Q\Uparrow)$, we say the number of negative occurrences of c on the left hand side process, its <u>index</u>. Then we show:

<u>Claim</u> Let $P\mathcal{R}Q$ and $P \to P'$ with P's index n. Then either (1) P' has $n \le i \le 2n$ index and $\exists Q'.\ Q \to Q'$ and $P'\mathcal{R}Q'$, or (2) P' has one less index than P, and $P'\mathcal{R}Q$. In particular if index is 0 only (2) holds. //

This is easy syntactic reasoning. Then we can also show $Q \to Q' \Rightarrow P \to^{1,2} P' \wedge P'\mathcal{R}Q'$ (which is easier),

Basic Syntactic Properties (4)

Now the latter shows $Q\Uparrow \Rightarrow P\Uparrow$. On the other hand, suppose $P\Uparrow$ and let P have the index, say, n. We show $\forall m \in \omega$. $Q \to^m Q'$ s.t. $P' \mathcal{R} Q'$ where $P'\Uparrow$ by induction on m. For $m=0$ this is obvious. If this holds for $m=k$, take $Q \to^k Q'$ with $P' \mathcal{R} Q'$ and $P'\Uparrow$. Then P' has an ω-infinite path of reduction

$$P' \to P_1' \to P_2' \to \cdots\cdots$$

But if P' has an index i, it cannot be the case the initial $i+1$ is all the case (2) of the claim, so at some P_j' we have $Q' \to Q''$ and $P_j' \mathcal{R} Q''$, as required. $\square$

Note the index may increase because the negative c may occur inside some replication. //

Basic Syntactic Properties (5)

Prop

(1) $M \to M' \Rightarrow \overset{\exists P.}{} [\![M]\!]_u \to P \approx_{\Downarrow} [\![M']\!]_u$

(2) $[\![M]\!]_u \to P \Rightarrow \overset{\exists M'.}{} M \to M' \wedge P \approx_{\Downarrow} [\![M']\!]_u$.

Proof:

(1) In $[\![M]\!]_u$ only (app) can take place, i.e. we have $M = ((\lambda x.M_0)M_1 \dots)M_n)$ and

$$[\![M]\!]_u \xrightarrow{\beta}{}^{+} (\nu x)([\![M_0 M_2 \dots M_n]\!]_u \mid [\![x = M_1]\!])$$

$$\approx_{\Downarrow} [\![M_0[M_1/x]\, M_2 \dots M_n]\!]_u$$

(safely assuming x can be free only in M_0). variable convention

But $\xrightarrow{\beta} \subseteq \approx_{\Downarrow}$ hence done.

(2) Similar to (1). $\square$

Basic Syntactic Properties (8)

Note! $\xrightarrow{\beta} \subseteq \simeq_{\Downarrow}$ is proved as:

(1) If $P \xrightarrow{\beta} Q$ and $Q\Uparrow$ obviously $P\Uparrow$.

(2) If $P \xrightarrow{\beta} Q$ and $P\Uparrow$, then let the infinite reduction path of P

$$P \to P' \to P'' \cdots\cdots$$

If $P' \equiv Q$ we are done. If not, by the confluence property we noticed before, there is R s.t.

$$\begin{tikzcd} & P \arrow[dl] \arrow[dr, "\beta"] & \\ P' \arrow[dr, "\beta"'] & & Q \arrow[dl] \\ & R & \end{tikzcd}$$

Now we can repeat the same argument between P' and R. In this way for all $n \in \omega$ we have $Q\Uparrow$, as required. //

Basic Syntactic Properties (9)

<u>Proposition</u> If M is closed,

$$[\![M]\!]_u \Downarrow P \Rightarrow P \xrightarrow{\overline{u}b} \text{ for some } b.$$

Proof: Because, then, P has a form:

$$(\nu \tilde{y})([\![\lambda x.N]\!]_u \mid [\![y_1 = M_1]\!] \mid \cdots \mid [\![y_n = M_n]\!])$$

for some $n \geq 0$. (because (app) and (Fetch) are the only possibilities. $\Box$.

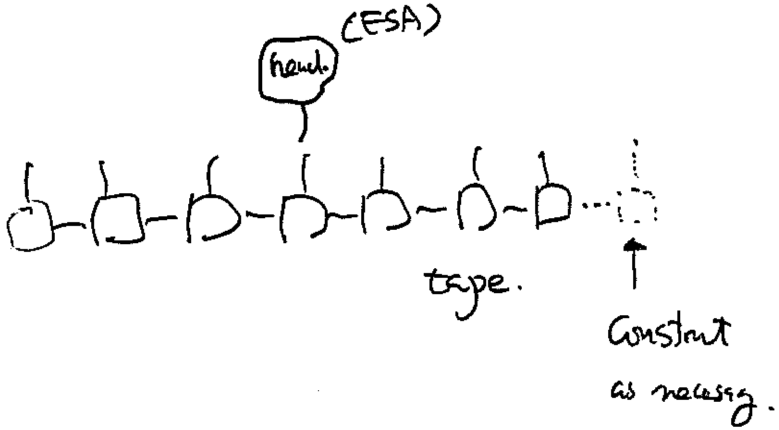
<u>Corollary</u> (of two preceding propositions)

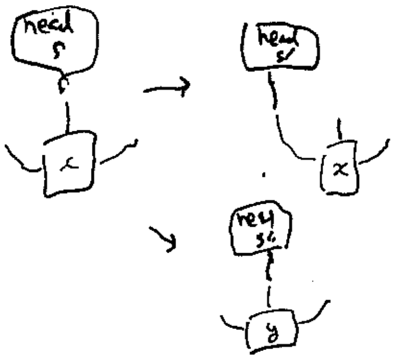
$$M\Downarrow \Leftrightarrow [\![M]\!]_u \Downarrow \Longrightarrow [\![M]\!]_u \overset{\overline{u}b}{\Longrightarrow}.$$

Remark: Easily $[\![M]\!]_u \Downarrow \Leftrightarrow [\![M]\!]_u \overset{\overline{u}b}{\Longrightarrow}$ too. //

Lecture III

The Asynchronous Calculus and Combinators.





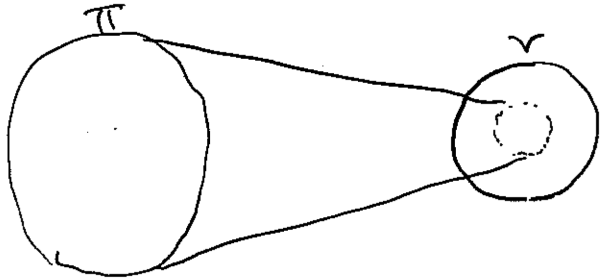
Core Calculus (1)

· Asynchronous calculus ([HT91, Boudol92 ...])

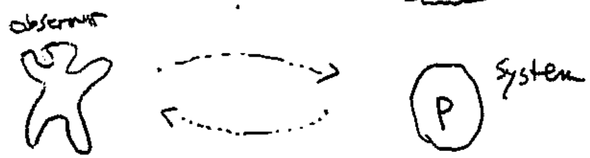
$$P ::= ax.P \mid \bar{a}b \mid P|Q \mid (\nu a)P \mid 0 \mid\ !ax.P$$

$$\begin{cases} ax.P \mid \bar{a}b \rightarrow P[b/x] \\ !ax.P \mid \bar{a}b \rightarrow\ !ax.P \mid P[b/x] \end{cases}$$

· Embedding of synchronous calculus.



· Asynchronous Observability. $\boxed{I(a) = !ax.\bar{a}x \approx 0.}$



Core Calculus (2)

(IN) $\quad 0 \xrightarrow{ab} \bar{a}b$ (sender sends a "message").

Proof of $I'(a) \approx_a 0$.

$$\mathcal{R} = \{(I'(a) \mid \Pi \bar{e_i}b_i,\ 0 \mid \Pi \bar{e_i}b_i)\}$$

(modulo $\equiv$).

$$\begin{tikzcd}
I'(a)|R \arrow[loop above, "\tau"] \arrow[r, dashed, no head] \arrow[d, "ef"'] \arrow[dd, bend left, "\bar{e_i}f_i"] & 0|R \arrow[loop above, "\epsilon"] \arrow[d, "ef"'] \arrow[dd, bend left, "\bar{e_i}f_i"] \\
I'(a)|R|\bar{e}f & 0|R|\bar{e}f \\
I'(a)|R' & 0|R' \quad //
\end{tikzcd}$$

Remark: Obviously $I'(a) \approx_s 0$.

Combinators for Concurrency. (1)

* Can we reduce the syntax further?

$$P ::= \underline{ax.P} \mid \underline{\bar{a}b} \mid \underline{P|Q} \mid \underline{(\nu x)P} \mid \underline{0} \mid \underline{!ax.P}$$

Or <u>can we do π-calculus without</u> <u>name passing</u>?

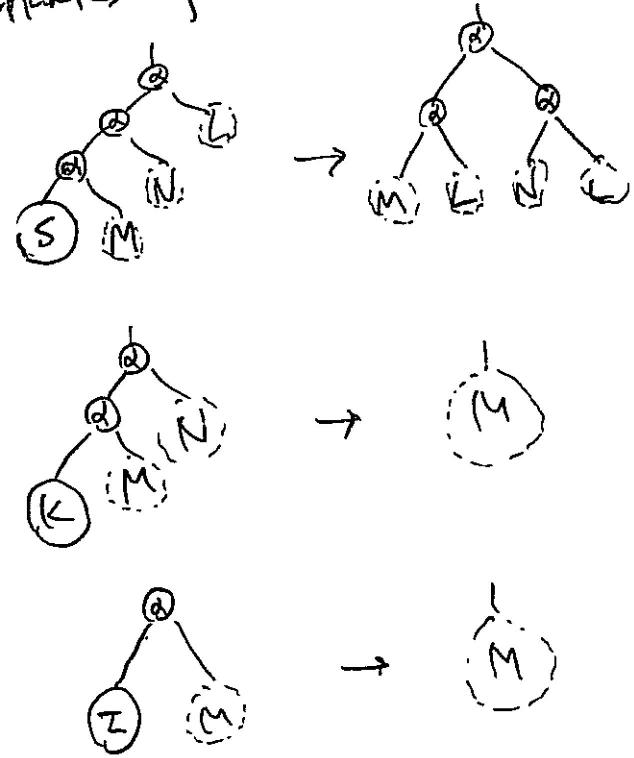
* But we CAN do λ-calculus without term passing.....

$$\begin{cases} \lambda^* x.MN = S(\lambda^* x.M)(\lambda^* x.N) \\ \lambda^* x.M = KM \qquad x \notin FV(M) \\ \lambda^* x.x = I \quad (= SKK) \end{cases}$$

Thm: $(\lambda^* x.M)N \to^* M[N/x]$.

Combinators …

· Dynamics of SKI-combinators:



· Universality: combinatory completeness

$$\forall x_1 \ldots x_n.\ \forall F(x_1 \ldots x_n).\ \exists M.$$

$$M x_1 \ldots x_n \to^* F(x_1 \ldots x_n).$$

(any functional constant is definable.)

($F(x_1 \ldots x_n)$ is a polynomial over $x_1 \ldots x_n$)

Combinators for Concurrency (3).

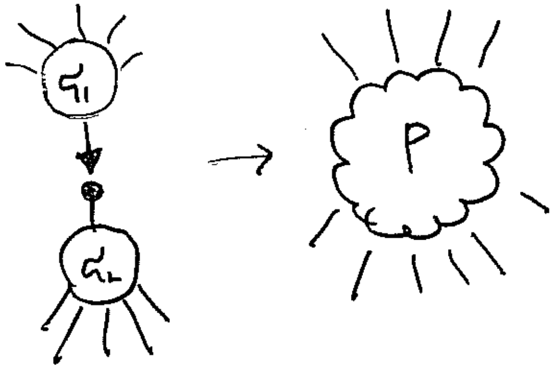
- We obtain:

$$P ::= c^{\pm}(\tilde{x}) \mid P|Q \mid (\nu x)P \mid 0 \mid !P$$

and further:

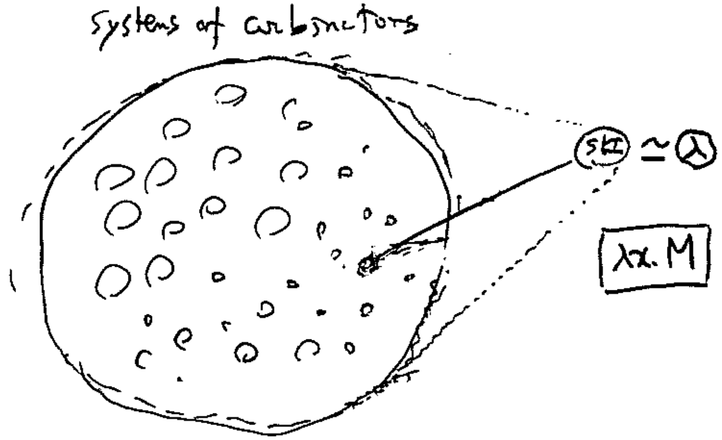
$$P ::= c^{\pm}(\tilde{x}) \mid P|Q \mid (\nu x)P \mid 0$$

- with dynamics:

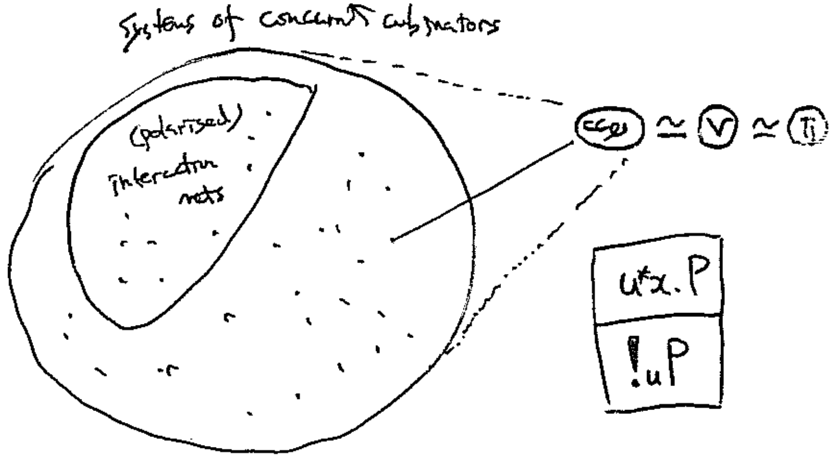


and universality: any (dyadic) interactive constraint behaviour is definable....

World of functions.



World of interactions.



Some Combinators.

Notation.

$M^+(ab) \stackrel{\mathrm{def}}{=} \bar{a}b$ (Message)

$FW^-(ab) \stackrel{\mathrm{def}}{=} ax.\bar{b}x$ (Forwarder)

$D^-(abc) \stackrel{\mathrm{def}}{=} ax.(\bar{b}x \mid \bar{c}x)$ (Duplicator)

$K^-(a) \stackrel{\mathrm{def}}{=} ax.0$ (Killer)

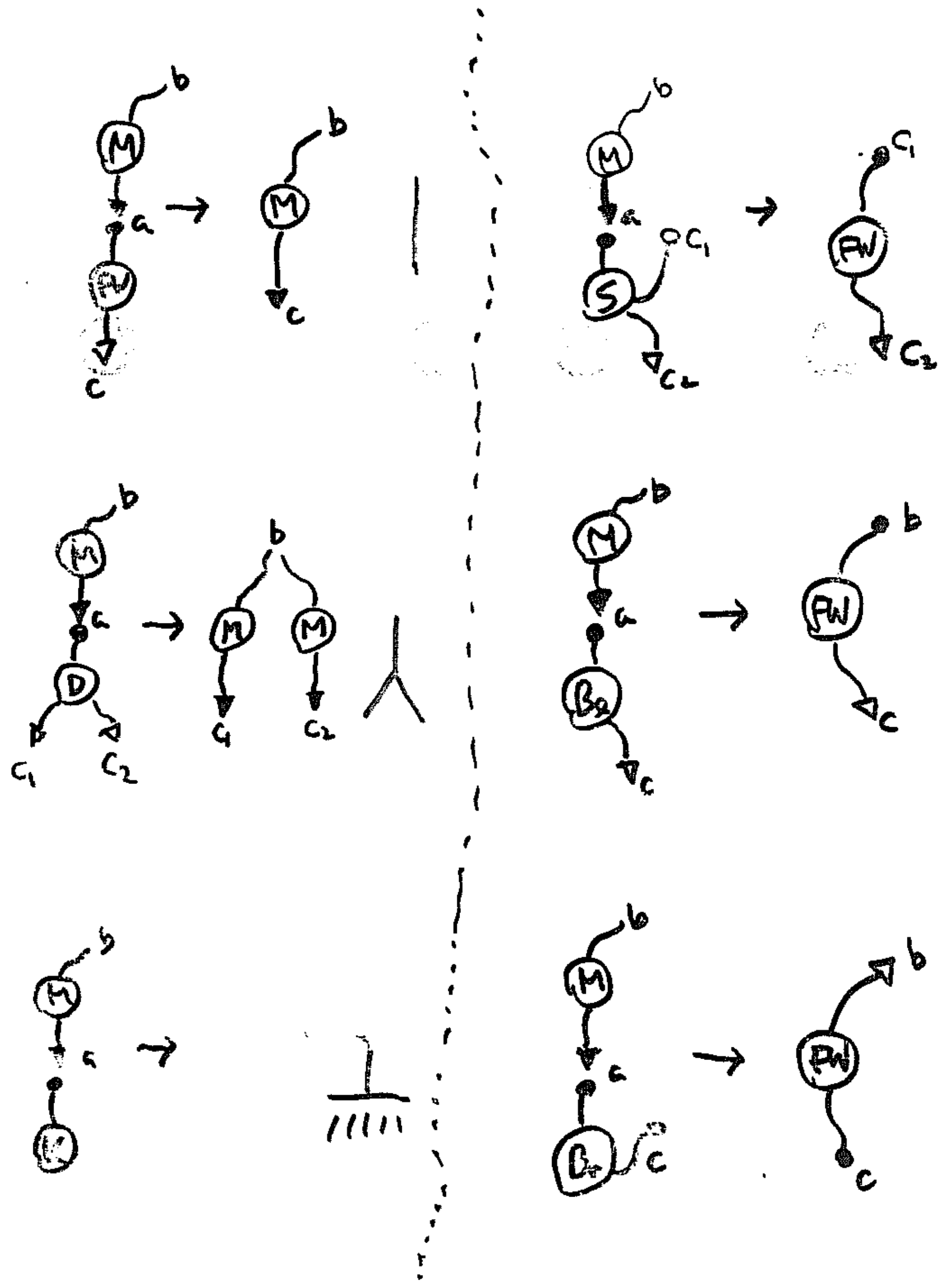
$S^-(abc) \stackrel{\mathrm{def}}{=} ax.FW(bc)$ (Synchroniser)

$B_l(ab) \stackrel{\mathrm{def}}{=} ax.FW(xb)$ (Binder-left)

$B_r(ab) \stackrel{\mathrm{def}}{=} ax.FW(bx).$ (Binder-right)

* write $C(\tilde{x})$, $C^+(\tilde{x})$, $C^-(\tilde{x})$, etc.

Atoms for Name Passing.



Analysis of Prefix (1).

<u>Definition</u> Given a, x, P define $a^{*}x.P$ as follows.

$P ::= a x.P \mid \bar{a}b \mid (c)P \mid \emptyset$

$\boxed{\lambda^{*}x.MN}$

INDUCTIVE CASES

(I) $a^{*}x.(P \mid Q) \stackrel{def}{=} (c_1 c_2)\big(D(a c_1 c_2) \mid c_1^{*}x.P \mid c_2^{*}x.Q\big)$

(II) $a^{*}x.(b)P \stackrel{def}{=} (b)\, a^{*}x.P$

BASE CASES.

(III) $a^{*}x.\emptyset \stackrel{def}{=} K(a).$

$\boxed{\lambda^{*}x.M}$ $x \notin FV(M).$

(IV). $a^{*}x.\, \varsigma^{+}(v\vec{w}) \stackrel{def}{=} (c)\big(S(a c v) \mid \varsigma^{+}(c\vec{w})\big)$

$a^{*}x.\, \varsigma^{-}(v\vec{w}) \stackrel{def}{=} (c)\big(S(a c v) \mid \varsigma^{-}(c\vec{w})\big)$

(V) $a^{*}x.\, M(bx) \stackrel{def}{=} FW(ab)$

$a^{*}x.\, FW(xb) \stackrel{def}{=} B_{l}(ab)$

$a^{*}x.\, FW(bx) \stackrel{def}{=} B_{r}(ab).$

$\boxed{\lambda^{*}x.x} = I$

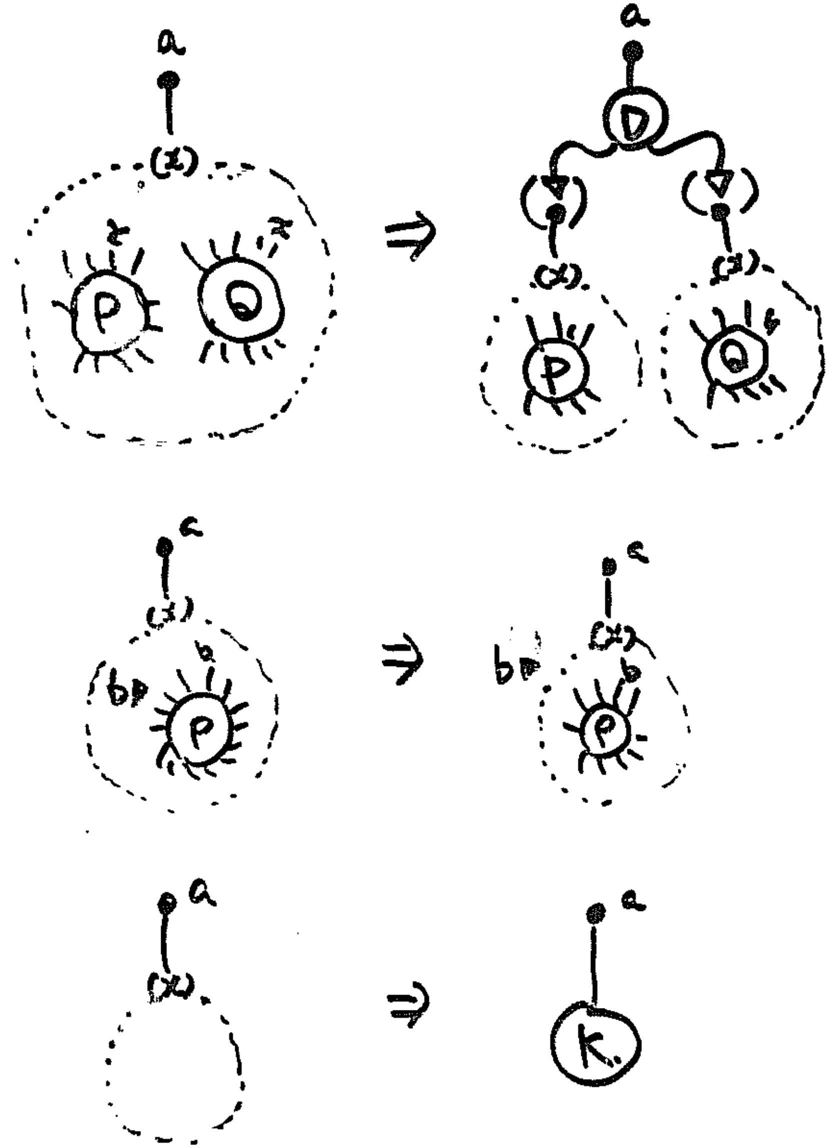
$a^{*}x.\, C(\vec{b}_1 x \vec{b}_2) \stackrel{def}{=} (c)\, a^{*}x.\big(FW(cx) \mid C(\vec{b}_1 c \vec{b}_2)\big)$

$a^{*}x.\, \varsigma^{-}(x\vec{b}) \stackrel{def}{=} (c)\, a^{*}x.\big(FW(xc) \mid \varsigma(c\vec{b})\big)$

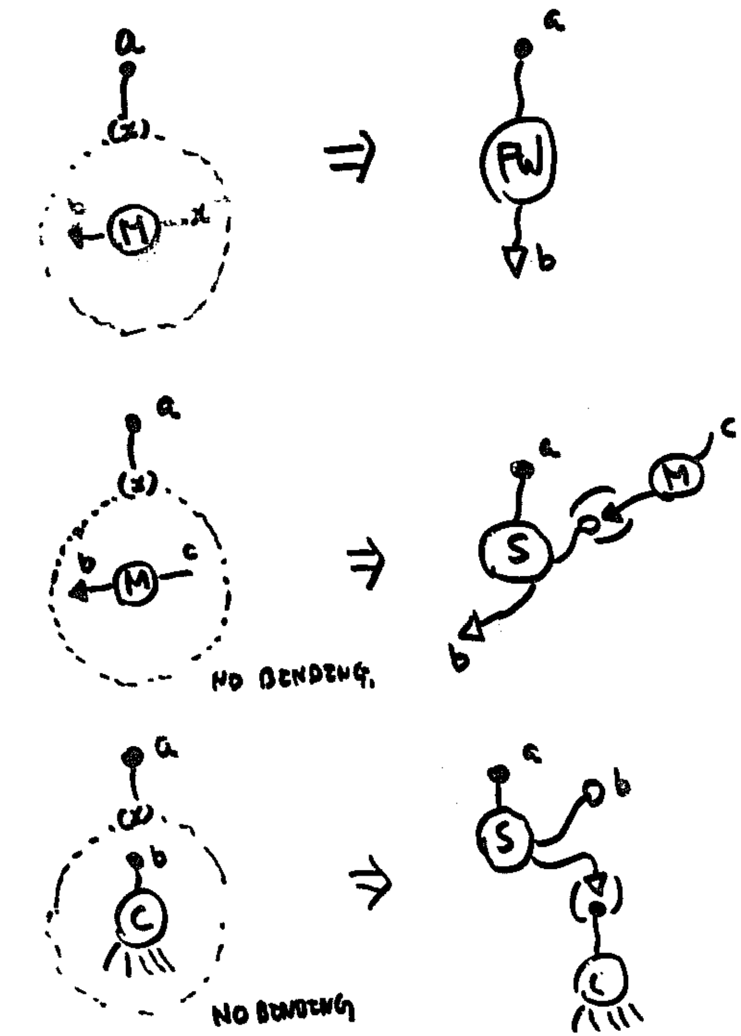
$a^{*}x.\, B_{r}(vx) \stackrel{def}{=} (c_1 c_2)\, a^{*}x.\big(D(x c_1 c_2) \mid B_{r}(c_1 c_2) \mid S(c_2 x c_1)\big)$

$a^{*}x.\, S(v x w) \stackrel{def}{=} (c_1 c_2)\, a^{*}x.\big(S(v c_1 c_2) \mid M(x c_1) \mid B_{r}(c_2 w)\big)$

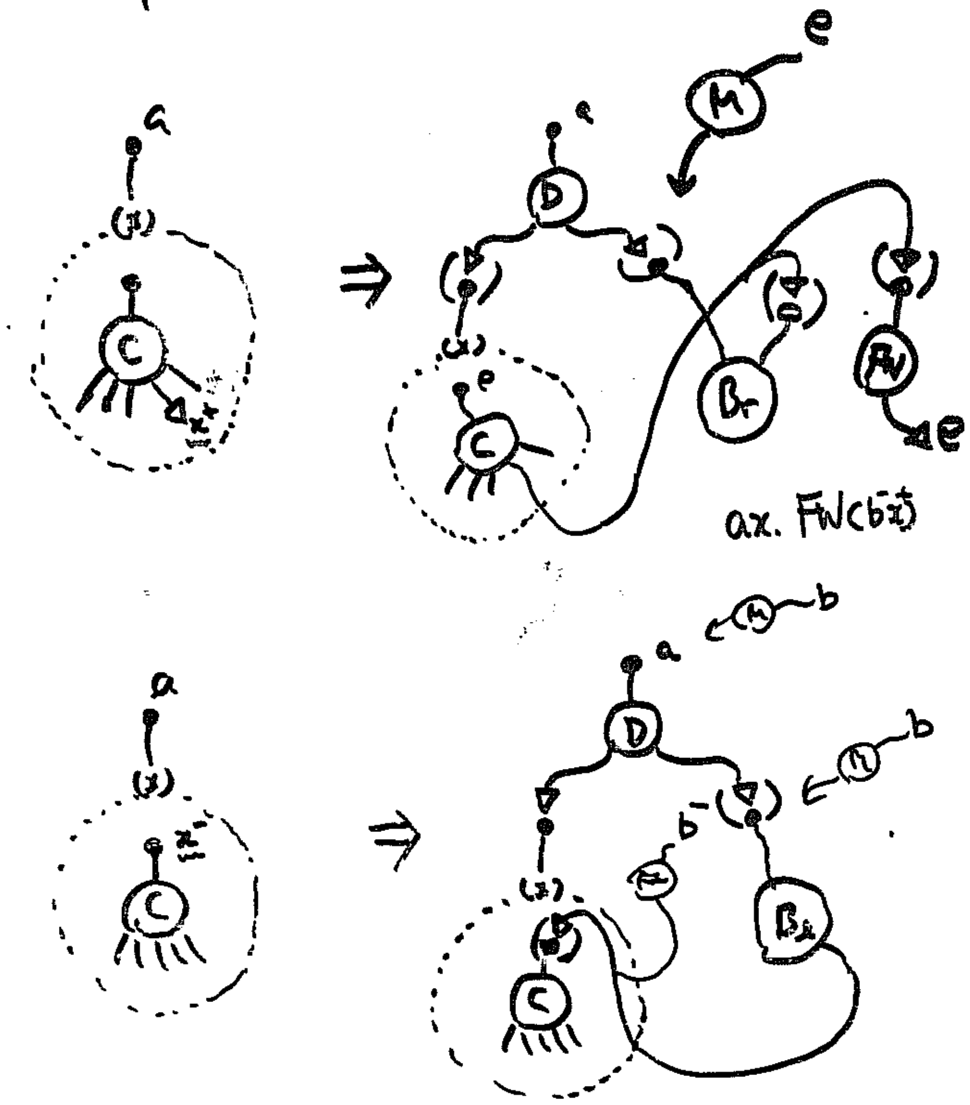
Analysis of Prefix (2).



Analysis of Prefix (3)



Analysis of Prefix (4)



Analysis of Prefix (5)

Proposition (Prefix Theorem) (H494a.).

For any a, x, and P

(i) $a^*x.P$ is a^--pointed.

(ii) $a^*x.P \mid \bar{a}b \rightarrow \approx P[b/x]$.

Proof: The only non-trivial case is (2). Suppose:

$$c_i^*x.P_i \mid \bar{c_i}b \rightarrow \approx P_i[b/x] \quad i=1,2.$$

then

$$(c_1c_2)\left(D(a,c_1,c_2) \mid c_1^*x.P_1 \mid c_2^*x.P_2\right) \mid \bar{c}b$$

$$\rightarrow \xrightarrow{\beta} \xrightarrow{\beta} \approx (P_1 \mid P_2)[b/x]$$

But $\xrightarrow{\beta}{}^* \subseteq \approx$ hence done. ∎

$a^*x.(P_1 \mid P_2)$

Analysis of Prefix (6)

Corollary (synchronous prefix "$\bar{a}$"-prefix).

For any a, x, v, P and Q, there exist terms $a^\circ x.P$ and $\bar{a}^\circ v.Q$ s.t.

(i) $a^\circ x.P$ is a^--pointed and $\bar{a}^\circ v.Q$ is $\bar{a}^-$-pointed.

(ii) $a^\circ x.P \mid \bar{a}^\circ v.Q \rightarrow \approx P[v/x] \mid Q$.

(iii) Both are polynomials from atomic agents.

Proof

Take e.g. $a^\circ x.P \stackrel{\text{def}}{=} a^*z.(c)(\bar{z}c \mid c^*x.P)$ and $\bar{a}^\circ v.Q \stackrel{\text{def}}{=} (c)(\bar{a}c \mid c^*w.(\bar{w}v \mid Q))$. Then

$$a^\circ x.P \mid \bar{a}^\circ v.Q \rightarrow \xrightarrow{\beta}{}^2 P[v/x] \mid Q.$$

Pointedness is obvious. ∎

Analysis of Prefix (7).

Corollary (polyadic name passing).

For any a, $\tilde{x}$, $\tilde{v}$, P, Q, there exist terms $a:(\tilde{x}).P$ and $\overline{a}:\langle\tilde{v}\rangle.Q$ such that

(i) $a:(\tilde{x}).P$ is a^--pointed and $\overline{a}:\langle\tilde{v}\rangle.Q$ is a^+-pointed,

(ii) $a:(\tilde{x}).P \mid \overline{a}:\langle\tilde{v}\rangle.Q \rightarrow\approx P[\tilde{v}/\tilde{x}] \mid Q$,

(iii) Both are polynomials from atomic gates.

Proof.

Using the greedy encodings

$a:(\tilde{x}).P \overset{\mathrm{def}}{=} a^{\circ}z.\,z^{\circ}x_1 \ldots z^{\circ}x_n.\,P$

$\overline{a}:\langle\tilde{v}\rangle.Q \overset{\mathrm{def}}{=} (c)(\overline{a}^{\circ}c.\,\overline{c}^{\circ}v_1 \ldots \overline{c}v_n.\,Q)$

then use $\xrightarrow{\beta}$ relying on pointedness. ∎

Analysis of Prefix (8).

Corollary (branching prefix).

For any a, $\tilde{x}$, $\tilde{y}$, $\tilde{v}$, P_1, P_2 and Q, there exist terms $a:[(\tilde{x}).P_1]\&[(\tilde{y}).P_2]$ and $\overline{a}:in_i[\langle\tilde{v}\rangle.Q]$

s.t.

(i) $a:[(\tilde{x}).P_1]\&[(\tilde{y}).P_2]$ is a^--pointed and $\overline{a}:in_i[\langle\tilde{v}\rangle.Q]$ is a^+-pointed.

(ii) $a:[(\tilde{x}).P_1]\&[(\tilde{y}).P_2] \mid \overline{a}:in_1[\langle\tilde{v}\rangle.Q]$

$\rightarrow\approx P_1[\tilde{v}/\tilde{x}] \mid Q$

$a:[(\tilde{x}).P_1]\&[(\tilde{y}).P_2] \mid \overline{a}:in_2[\langle\tilde{v}\rangle.Q]$

$\rightarrow\approx P_2[\tilde{v}/\tilde{x}] \mid Q$

(iii) Both are polynomials from atomic gates.

Proof.

Use the previous corollary. ∎

Analysis of Replicator (1)

$$!ax.\overline{x}(b)\overline{x}b$$

Notation.

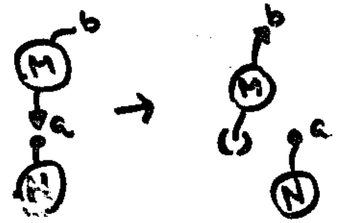
Given $G(\tilde{x})$,

$$!G^{-}(\tilde{x}) \triangleq !G(\tilde{x}) \qquad (+6)$$

$$\#G^{\pm}(e) \stackrel{\mathrm{def}}{=} !e:(\tilde{x}).G^{-}(\tilde{x}) \qquad (+7)$$

$$\#!G^{-}(e) \stackrel{\mathrm{def}}{=} !e:(\tilde{x}).!G^{-}(\tilde{x}) \qquad (+6)$$

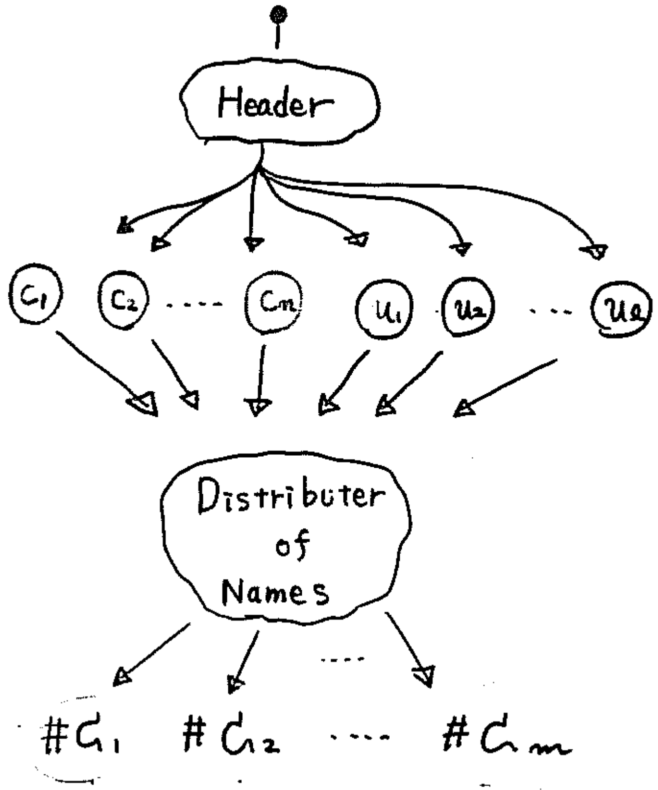
$$N(e) \stackrel{\mathrm{def}}{=} !ex.c(c)\overline{x}c \qquad (+1)$$



* So we have 27 atoms, whose instantiations are atomic agents.

Construction for the Theorem (2)

$!^{*}_{u} P$ where $P \equiv c_1 \cdots c_n \blacktriangleright (G_1(\tilde{u}_1), \ldots, G_m(\tilde{u}_m))$



$$c_1c_2\cdots c_n \blacktriangleright (G_1(\tilde{u}_1), G_2(\tilde{u}_2), \cdots G_m(\tilde{u}_m))$$

Embedding (1).

Definition.

$$(ax.P)^\circ \stackrel{\text{def}}{=} a^*x.\,P^\circ$$

$$\bar{a}b^\circ \stackrel{\text{def}}{=} \bar{a}b$$

$$((c)P)^\circ \stackrel{\text{def}}{=} (c)\,P^\circ$$

$$(P|Q)^\circ \stackrel{\text{def}}{=} P^\circ | Q^\circ$$

$$\emptyset^\circ \stackrel{\text{def}}{=} \emptyset$$

$$(!ax.P)^\circ \stackrel{\text{def}}{=} !a^*x.\,P^\circ.$$

Embedding (2)

Lemma

(1) $P \approx Q \Rightarrow a^*x.P \approx a^*x.Q$

(2) $P \approx Q \Rightarrow !a^*x.P \approx !a^*x.Q$

Theorem

(1) $P \to P' \Rightarrow P^\circ \to\approx P'.$

(2) $P^\circ \to P'' \Rightarrow \exists P'.\ P' \approx P^\circ \wedge P \to P'.$

(3) $P \approx Q \Leftrightarrow P^\circ \approx Q^\circ.$

Proof: For (3) prove $P \approx P^\circ$ using the above lemma. ∎

Generators. (1)

Notation Given a set of terms $\mathcal{P}_0$, we write $\overline{\mathcal{P}_0}$ for the least set of terms with $\mathbf{0}$ such that:

$$P \in \mathcal{P}_0 \Rightarrow P\sigma \in \overline{\mathcal{P}_0} \quad (\sigma \text{ is renaming}).$$

$$P, Q \in \overline{\mathcal{P}_0} \Rightarrow P \mid Q \in \overline{\mathcal{P}_0}$$

$$P \in \overline{\mathcal{P}_0} \Rightarrow (a)P \in \overline{\mathcal{P}_0}$$

Definition $\mathcal{P}_0$ is a set of generators up to $\approx$ if the following holds.

$$\forall P \in \mathcal{P}_{\pi a}\ \exists P' \in \overline{\mathcal{P}_0}.\ P \approx P'.$$

Generators (2)

Corollary.

$\mathcal{P}_{\pi a}$ has a finite set of generators.

Proof: Direct from (the proof of) the embedding theorem. ∎

Notes on Interactive Behaviour on CC (1)

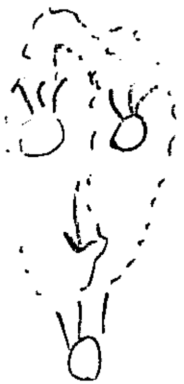
What We Have Gotten....

MN

$M ::= \lambda x M \mid MN \mid x$

$M ::= G \mid (MN)$

... is a BNF:

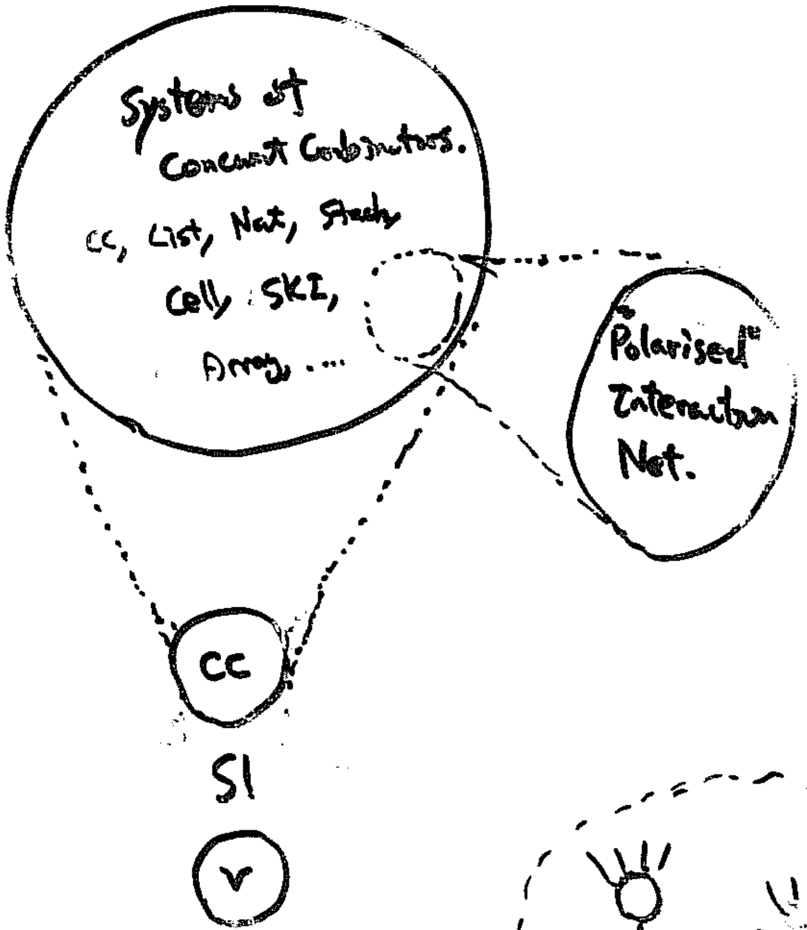


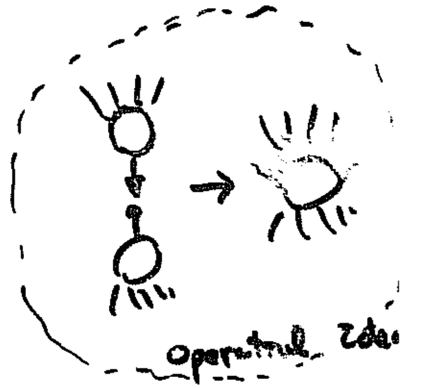
$$P ::= C^{(x)}_{(a)} \mid P|Q \mid (x)P \mid 0.$$

which gives a self-contained universe of concurrent computing.

$3\ 5 \xrightarrow{+} 8$

* What is sleeping underneath?





Notes on Interactive Behaviour on CC. (2)

From Applicative Behaviour to SKI.

$$S x_1 \ldots x_n \to F(x_1 \ldots x_n).$$

$$\Downarrow$$

$$(\lambda x_1 \ldots x_n.M) x_1 \ldots x_n \to M$$

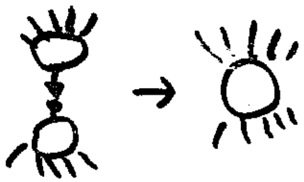
$$\Downarrow$$

$$(\lambda x.M)x \to M$$

$$\Downarrow$$

S, K, I, ..

From Interactive Behavior to CC.



$$\Downarrow$$

Branching + Replication.

$$\Downarrow$$

$$ax.P \mid \bar{a}b \to P\{b/x\}$$

$$!ax.P \mid \bar{a}b \to !ax.P \mid P\{b/x\}$$

$$\Downarrow$$

$\mathcal{M}$, D, FN, ...

Lecture IV

Types for Mobile Processes

Types for Mobile Processes (1)

- Types for functions:

$$\lambda x.x : Nat \to Nat$$

The type $Nat \to Nat$ says the following:

(1) The term is composable with

$N : Nat$ on the right

$M : (Nat \to Nat) \to \alpha$ on the left.

(2) when composed with $N : Nat$, it gives:

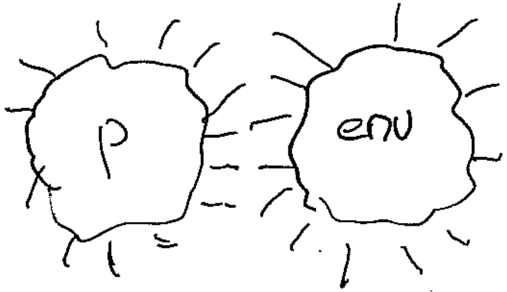
$(\lambda x.x)N : Nat$

else we gurentee NOTHING.

(3) It is semantically a function from Nat to Nat.

Types for Mobile Processes (2)

- What are types for processes?



> Given a specification of environments, what would be the behaviour of the process?

ex. $a(x).\overline{b}[x]$

when composed with $\overline{a}[3]$, it will emit $\overline{b}[3]$ .....

Types for Mobile Processes (3)

- Current status:

  - Sorting [Milner 92] ⇐
    - single notion of types.
    - structure of "name carrying".
    - Type inference [Vasco & Honda 93] and refinement [Pierce & Sangiorgi 93, ⇐ Vasco 94, Honda 96, ...]

  - Types with dynamic structure.

    $(\nu c)\, az.(\overline{z}c \mid cx_1.(\overline{z}c \mid cx_2.P))$ [Yoshida 96]

  - Semantics. ⇐

> - We still understand LITTLE
> - But connections to various areas (LL, game semantics) are emerging.
> - What is the "spectrum of types" in this setting?

Sorting (1)

(1) Polyadic π-calculus.

$$P ::= a(x_1 .. x_n).P \mid \bar{a}[v_1 .. v_n].P \mid P|Q \mid (\nu x)P \mid \emptyset \mid !P$$

with the same rules for $\equiv$ except:

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

reduction:

$$a(x_1 .. x_n).P \mid \bar{a}[v_1 .. v_n].Q \to P\{\tilde{v}/\tilde{x}\} \mid Q.$$

(ii) Sorting: $\langle \{S_i^{A_i}\}, \vdash \rangle$

$\bar{a}[bc].\bar{e}[bc] \Rightarrow \{S_1^{ae}, S_2^{b}, S_3^{c}\}$ $S_1 \mapsto S_2 S_3$

$a(xz).\bar{b}[x] \Rightarrow \{S_1^{a}, S_2^{\phi}, S_3^{\phi}, S_4^{b}\}$ $S_1 \mapsto S_2 S_3$, $S_4 \mapsto S_2$

$\bar{a}[a] \Rightarrow \{S^{a}\}$ $S \mapsto S$

Sorting (2)

(3) "Type"-presentation.

$\{S_1^{ae}, S_2^{b}, S_3^{c}\}$ $S_1 \mapsto S_2 S_3$ $\quad \boxed{\bar{a}[bc]\ \bar{e}[bc]}$

$\Updownarrow$

$a : (\alpha_1 \alpha_2)$, $e : (\alpha_1 \alpha_2)$, $b : \alpha_1$, $c : \alpha_2$

$\{S_1^{a}, S_2^{\phi}, S_3^{b}\}$ $S_1 \mapsto S_2$, $S_2 \mapsto S_3 S_3$ $\quad \boxed{a(x).\bar{x}[bb].\emptyset}$

$\Updownarrow$

$a : ((\alpha\alpha))$, $b : \alpha$

Types: $\alpha ::= X \mid (\alpha_1 .. \alpha_n) \mid \mu X.\alpha.$

Typing: $a_1 : \alpha_1, a_2 : \alpha_2, .. a_n : \alpha_n$ $\quad \Gamma, \Delta, \Xi, \cdots$

Equality: $\approx$.

$$\left.\begin{array}{c} (\alpha_1 \ldots \alpha_n) \xrightarrow{i} \alpha_i \\ X \xrightarrow{X} \mathbb{1} \\ \dfrac{\alpha[\mu X.\alpha / X] \xrightarrow{l} \beta}{\mu X.\alpha \xrightarrow{l} \beta} \end{array}\right\}$$

$\alpha \approx \beta$ when they are bisimilar w.r.t. $\xrightarrow{l}$.

Sorting (3)

(4) Type Inference System $\Gamma \vdash P$.

$$\frac{\Gamma \vdash P}{\Gamma/x \vdash a(x_1..x_n).P} \quad \begin{pmatrix}\Gamma(a) = (\alpha_1..\alpha_n) \\ \Gamma(x_i) = \alpha_i\end{pmatrix}$$

$$\frac{\Gamma \vdash P}{\Gamma \vdash \bar{a}[x_1..x_n].P} \quad (\text{same})$$

$$\frac{\Gamma \vdash P \quad \Gamma \vdash Q}{\Gamma \vdash P \mid Q} \qquad \frac{\Gamma \vdash P}{\Gamma/a \vdash (\nu a)P}$$

$$\frac{\Gamma \vdash P}{\Gamma \vdash !P} \qquad \emptyset \vdash 0$$

$$\frac{\Gamma \vdash P}{\Gamma, a{:}\alpha \vdash P} \ \alpha \in FN(\Gamma) \qquad \frac{\Gamma, a{:}\alpha, b{:}\beta, \Delta \vdash P}{\Gamma, b{:}\beta, a{:}\alpha, \Delta \vdash P}$$

Sorting (4)

(5) Basic Syntactic Properties.

Prop.

(1) $\Gamma \vdash P \Rightarrow FN(P) \subseteq FN(\Gamma)$.

(2) $\Gamma, a{:}\alpha \vdash P \wedge a \notin FN(P) \Rightarrow \Gamma \vdash P$.

(3) $\Gamma \vdash P \wedge P_0$ is a subterm of P
$\Rightarrow \exists \Delta.\ \Delta \vdash P_0$.

(4) $\Gamma \vdash P \wedge P \equiv Q \Rightarrow \Gamma \vdash Q$.

(5) $\Gamma \vdash a(x_1..x_n).P \mid \bar{a}[v_1..v_m].Q$
$\Rightarrow n = m$.

(6). (Subject Reduction)
$\Gamma \vdash P \wedge P \rightarrow P' \Rightarrow \Gamma \vdash P'$.

Sorting (5)

<u>Def</u>

$$P \in Err \overset{def}{\iff} \exists P'.\ P \to^* P' \equiv (\nu \tilde{b})(a(x_1..x_n).P \mid \bar{a}[v_1..v_m].Q \mid R) \quad n \neq m.$$

<u>Theorem</u>

$$\Gamma \vdash P \Rightarrow P \notin Err.$$

Ex. λ-encoding.

$$[\![x]\!]_u = \bar{x}u \qquad [\![\lambda x.M]\!]_u = u(xz).[\![M]\!]_z$$

$$[\![MN]\!]_u = (\nu c_1 c_2)([\![M]\!]_{c_1} \mid \bar{c}_1[c_2 u].!c_2(z).[\![N]\!]_z)$$

If M has free variables $x_1,..x_n$, we have:

$$[\![M]\!]_u = x_1{:}Var, ..\ x_n{:}Var,\ u{:}Body$$

where $Var \overset{def}{=} (Body)$ and $Body \overset{def}{=} \mu X.((X)X)$.

Lecture V

Symmetries in Processes.

(from slides for a seminar).