Handout

for
A Short Course
on
Mobile Processes

University of Gabon
February 1993

K. Handa

Notes:

This is a post-lecture handout for a short course on mobile processes given at University of Mobile Processes on February 1993. 5 lectures each 3 hours as well as 2 seminars, 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 Usoro Usowane 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 Calculus in Languages (only \( \pi \)-calculus is treated in detail.)
- Introduction to Asynchronous Calculus of Communicating Systems
- Types for Mobile Processes
- Syntax on Processes.

A Short Course on Mobile Processes

Plan of Lecture
2/12

- General Introduction
  Computation by interaction: context of \( \pi \)-calculus
- Syntax
  Syntax of Core Calculus
  Extensions and Variants.
- Structural Rules
  Definition
  Examples
  Discussion on Structural Rules
- Reduction
  Definition
  Examples
  Scope Extension
  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 "Funchosas Poas"
- Embedding Calculi and Languages
- Functions as Processes
  Lazy λ-calculus
  Encoding
  Basic Syntactic Properties
  Observability and Full abstraction
  Types for λ-like Agents (*)
- Parallel Imperative Language
  Variables
  Control Constructs
  Procedures (*)
  Operational Correspondence
  Observability
  Turing Machine in TL (*)
  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 Repetition

(*) Optional.
Introsuction to Universality Theorem.

Generalised Concurrent Combinators
- Definition
- Examples

Abstract Notion of Embeddings
- Definition
- Basic Properties

Examples of Embeddings (*)
- Computability
- Synchronous in Asynchronous
- Cases for summations.

Universality Theorem.
- Basic constructions
- Proof outline
- Discussions and Comparisons.

Remaining Issues.

(?) Optional.

Types for Mobile Processes

- Backgrounds
  - Types for functions
  - Behavioural 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

- Refinement of Sorting (1)
  - Linearity
  - Replication (Server-Client Types)
  - I/O Types
- Refinement of Sorting (2)
  - Type System for Linearity
  - Inversion Behviourol Equality
- Beyond Sorting
Lecture I

Basics of Mobile Processes

Abstraction as Interaction

- Decomposition of the "whole" into components
- Their interaction

- Examples
  - Physics (particles)
  - Ecology
  - Object model (Internet)
  - Agent model (AIS)
  - Biology
Computation as Interaction.

Context of π-calculus (1)

- Denotational Semantics (Scott-Simuly 80)
  - Compositional approach to semantics, influenced by λ-calculus.
  - Mathematical foundations.

- Difficulty in treating Interfering Concurrently:
  \[ x := 1 \parallel x := 2 ; x := 2x \]

- CCS (Harper 80)
  - Return to syntax.
  - Calculus purely based on "interaction" rather than "function".

\[
\alpha P + \Omega | \tau . R + S \rightarrow \pi R \text{ (Cooperation)}
\]

- Basic theory of "behavioral equivalences".
  - When can we say two processes are (essentially) the same?
Context of $\pi$-calculus ($\pi$)

- Issues in CCS
  - Not extendable to programming languages
  - Encoding is hard.
  - Synchronization is doubtful.
- $\pi$-calculus ($\pi$-calculus; Milner, Park, Wadsworth)
  - Another syntax for interaction,
    with a new equation:
    \[
    \text{Communication} = (\text{Synchronisation}) \text{ New Syntax}
    \]
    Thus we get:
    \[
    \lambda X.P (\lambda X.Q \rightarrow \text{Act}P) \mid Q
    \]
  - Solving the issues of CCS.
    - Extensible.
    - Encoding of data.
    - Synchronization.
  - Links to other concurrency models.
    - We still do not have Scott models.

$\pi$-Calculus

- Syntax (terms): $P, Q, \ldots$
- Structural rules: $\Rightarrow$
- Reduction

- Labelled Transition: $P \Rightarrow (\exists Q \rightarrow)$

- Dissimilarities $\sim, \not\sim$

Basic constructs:
\[
\begin{align*}
& a(z). P \rightarrow (a z, P) \\
& a: \{O, P\} \times \{O, P\} \\
& \text{let } Z(x) = P \text{ in } Q
\end{align*}
\]
Terms

- Throughout the lecture, we fix the set of names $N$, ranging over by $a, b, c, \ldots$ or $x, y, z, \ldots$.

- Terms:
  \[ P ::= \text{ax}.P \mid \text{ab}.P \mid P \upharpoonright Q \]
  \[ (\forall a) P \mid \emptyset \mid !\text{ax}.P \]

- Binders:
  \[ \text{ax}.P \quad (\forall a) P \quad !\text{ax}.P \]

Structural Equality

- $\equiv$ is the smallest congruence relation closed under:
  - $P \equiv Q \Rightarrow P \equiv Q$
  - $P \upharpoonright \emptyset = \emptyset (P \equiv Q \Rightarrow P \equiv Q (\emptyset R))$
  - $(\forall a) P \equiv (\forall a) P \equiv (\forall a) P \equiv (\forall a) P$
  - $(\forall a) P \upharpoonright Q \equiv (\forall a) (P \upharpoonright \emptyset) (c \land \text{env}(Q))$
  - $(\forall a) P \upharpoonright Q$

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

- The reduction relation $\rightarrow$ is the smallest relation generated by:

- (COM) $\alpha x. P | ab. Q \rightarrow P[x:=Q] Q$

- (REP) $! \alpha x. P | ab. Q \rightarrow P[x:=Q] \! \alpha x. P$

- (PAR) $P \rightarrow Q \Rightarrow P \upharpoonright R \rightarrow Q \upharpoonright R$

- (RES) $P \rightarrow Q \Rightarrow \omega \rightarrow P \rightarrow \omega \rightarrow Q$

- (STR) $P \Rightarrow P \rightarrow Q \ \omega \Rightarrow Q \Rightarrow P \rightarrow Q$.

LTS and Bisimulations.

- Labels:

  $$L ::= \overline{ab} | \overline{a(b)} | \overline{c(b)} | \omega$$

  (optional)

- Given $\rightarrow$,

  **Def**: $\equiv \rightarrow P \times P$ is a strong bisimulation when $POQ$ and $P \rightarrow P'$ s.t. $\text{Fn}(Q) \cap \text{Bv}(Q) = \emptyset$ we have, for some $Q'$, $Q \rightarrow Q'$ and $P \rightarrow Q'$.

  Similarly, exchanging $P$ and $Q$.

  $$\Rightarrow \equiv \begin{cases} \overline{a} & \text{if } \ell = 2 \\ \overline{a} & \text{if } \ell \neq 2. \end{cases}$$
Labeled Transition Relation.

\[ \text{COMMON.} \]
\[ v \text{ (IN)} \quad ax. P \xrightarrow{a} P_{a \in x} \]
\[ v \text{ (OUT)} \quad ax. P \xrightarrow{a} P \]
\[ \text{COMMON} \]
\[ \beta \frac{P \xrightarrow{\beta} P'}{a \beta \rightarrow \alpha} \]
\[ v \text{ (OUT)} \quad P \xrightarrow{\beta} P_{\beta \in x} \]
\[ v \text{ (PAIR)} \quad P \xrightarrow{\beta} P \xrightarrow{\beta} P \]
\[ v \text{ (R (S))} \quad P \xrightarrow{\beta} P \xrightarrow{\beta} P \]
\[ v \text{ (REP)} \quad \text{rep. } P \leftrightarrow \text{rep. } P_{\beta \in x} \]

\[ \text{CONSERVATIVE} \]
\[ \text{COMMON} \]
\[ P \xrightarrow{\beta} P' \quad \alpha \beta \rightarrow \alpha' \Rightarrow P \xrightarrow{\beta} P' \]
\[ \text{(B.COM)} \]
\[ P \xrightarrow{\alpha} P' \quad \alpha \beta \rightarrow \alpha' \Rightarrow P \xrightarrow{\beta} P' \]

Prop.

(1) \[ P \xrightarrow{\beta} P' \Rightarrow P \xrightarrow{\beta} P' \]

(2) \[ P \xrightarrow{\alpha} Q \Rightarrow P \xrightarrow{\beta} Q \]

(3) They give the same bisimilarity.

\[ P \text{rep. } P \xrightarrow{\beta} P' \Rightarrow P \text{rep. } P \xrightarrow{\beta} P' \]

\[ * b \text{ rep. } P \text{ (P).} \]
\[ \text{[symmetric]} \]
\[(BZN) \quad a \cdot P \xrightarrow{c(a)} P \]
\[(BN) \quad P \xrightarrow{c(a)} P', \quad Q \xrightarrow{c(a)} Q' \Rightarrow P|Q \xrightarrow{c(a)} (P|Q') \]

\[ \text{Prop} \]
\[ \text{Again we have the same bisimulations.} \]

\[ \because \text{Because of the side condition for bound actions.} \]

\[ \text{Bisimulations for:} \]
\[ * \quad P \oplus Q \sim Q \oplus P \]
\[ \{ (P \oplus Q, Q \oplus P) \} \cup \{ (P \oplus Q, Q \oplus P) \} \]
\[ \cup \equiv \]

\[ * \quad P \oplus P \sim P \]
\[ \{ (P \oplus P, P) \} \cup \equiv \cup \equiv \]

\[ * \quad P \oplus Q \Rightarrow P \sim Q \]
\[ \{ (P, Q) | P \oplus Q \} \cup \equiv \]

\[ C_{[\equiv]} (P, Q) \]
\[ \overset{\text{P}}{\rightarrow} Q \neq Q \]
\[ Q \neq Q \]
Replication Law.

Prop: If c occurs only as negative subjects in R₁, R₂ and P₁:

\[(\varrho c \cdot P \mid R₁ \mid R₂) \equiv Q₁ \rightarrow Q₁\]

\[\sim (\varrho c) (\varrho c \cdot P \mid R₁) \mid (\varrho c) (\varrho c \cdot P \mid R₂) \equiv Q₂\]

Proof: Show the following is a bisimulation:

\[Q = \{ (σ₁, σ₂) \mid σ₁ \equiv Q₁, σ₂ \equiv Q₂, Q₁, Q₂ as above \}\]

taking care of the invariance: c never becomes \textit{posterior} nor become \textit{positive}.
Counterexample,

\( \Rightarrow \text{If } c \text{ occurs positively:} \)

\[ (rc)(!x.e \mid \bar{e} \mid cx.fx) \Rightarrow \bar{e}v \]

\[ \chi (rc)(!x.e \mid \bar{e}) \mid \]

\[ (rc)(!a.e \mid cx.fx) \]

\( \Rightarrow \text{If } c \text{ occurs as object:} \)

\[ (rc)(!x.e \mid \bar{e} \mid \bar{c} \mid ly.yx.fx) \]

\[ \rightarrow (rc)(!x.e \mid \bar{e} \mid cx.fx) \]

Pointedness.

\[ \text{Det } a \text{ occurs active in } P \text{ when:} \]

\[ P = (rc)(P \cup R) \quad P_0 \{ \text{ax. } P \} \text{ positive} \]

\[ \text{and } a \notin \{ e \}. \]

\[ \text{Det } P \text{ is a-pointed when:} \]

1. There is a unique active occurrence of \( a \) in \( P \)
2. No other names occur active in \( P \)
3. \( P \not\vdash \)

[Diagram: a point labeled \( a \) on a circle labeled \( P \).]
Behaviour of Agents (1)

\[ a.(x_1 x_2).P \]
\[ a.(x_1 x_2).Q \]
\[ a.(x_1 x_2).P \]
\[ a.(x_1 x_2).Q \]
\[ \bar{a}.(\bar{x}_1 \bar{x}_2).Q \]
\[ \bar{a}.(\bar{x}_1 \bar{x}_2).Q \]
\[ \bar{a}.(\bar{x}_1 \bar{x}_2).Q \]
\[ \bar{a}.(\bar{x}_1 \bar{x}_2).Q \]

\[ a.(\bar{x}_1 \bar{x}_2).P \]
\[ a.(\bar{x}_1 \bar{x}_2).Q \]
\[ a.(\bar{x}_1 \bar{x}_2).P \]
\[ a.(\bar{x}_1 \bar{x}_2).Q \]
\[ \bar{a}.(x_1 x_2).Q \]
\[ \bar{a}.(x_1 x_2).Q \]
\[ \bar{a}.(x_1 x_2).Q \]
\[ \bar{a}.(x_1 x_2).Q \]

\[ P \]
\[ Q \]

\[ \bar{a} \]

\[ \bar{x}_1 \]

\[ \bar{x}_2 \]

\[ \bar{a} \]

\[ \bar{x}_1 \]

\[ \bar{x}_2 \]

\[ \bar{a} \]

\[ \bar{x}_1 \]

\[ \bar{x}_2 \]

\[ \bar{a} \]

\[ \bar{x}_1 \]

\[ \bar{x}_2 \]

\[ \bar{a} \]

\[ \bar{x}_1 \]

\[ \bar{x}_2 \]

\[ \bar{a} \]

\[ \bar{x}_1 \]

\[ \bar{x}_2 \]

\[ \bar{a} \]

\[ \bar{x}_1 \]

\[ \bar{x}_2 \]

\[ \bar{a} \]

\[ \bar{x}_1 \]

\[ \bar{x}_2 \]

\[ \bar{a} \]

\[ \bar{x}_1 \]

\[ \bar{x}_2 \]

\[ \bar{a} \]

\[ \bar{x}_1 \]

\[ \bar{x}_2 \]

\[ \bar{a} \]

\[ \bar{x}_1 \]

\[ \bar{x}_2 \]

\[ \bar{a} \]

\[ \bar{x}_1 \]

\[ \bar{x}_2 \]

\[ \bar{a} \]

\[ \bar{x}_1 \]

\[ \bar{x}_2 \]

\[ \bar{a} \]

\[ \bar{x}_1 \]

\[ \bar{x}_2 \]

\[ \bar{a} \]

\[ \bar{x}_1 \]

\[ \bar{x}_2 \]

\[ \bar{a} \]

\[ \bar{x}_1 \]

\[ \bar{x}_2 \]

\[ \bar{a} \]

\[ \bar{x}_1 \]

\[ \bar{x}_2 \]

\[ \bar{a} \]

\[ \bar{x}_1 \]

\[ \bar{x}_2 \]

\[ \bar{a} \]

\[ \bar{x}_1 \]

\[ \bar{x}_2 \]

\[ \bar{a} \]

\[ \bar{x}_1 \]

\[ \bar{x}_2 \]

\[ \bar{a} \]

\[ \bar{x}_1 \]

\[ \bar{x}_2 \]

\[ \bar{a} \]

\[ \bar{x}_1 \]

\[ \bar{x}_2 \]

\[ \bar{a} \]

\[ \bar{x}_1 \]

\[ \bar{x}_2 \]

\[ \bar{a} \]

\[ \bar{x}_1 \]

\[ \bar{x}_2 \]

\[ \bar{a} \]

\[ \bar{x}_1 \]

\[ \bar{x}_2 \]

\[ \bar{a} \]

\[ \bar{x}_1 \]

\[ \bar{x}_2 \]

\[ \bar{a} \]

\[ \bar{x}_1 \]

\[ \bar{x}_2 \]

\[ \bar{a} \]

\[ \bar{x}_1 \]

\[ \bar{x}_2 \]

\[ \bar{a} \]

\[ \bar{x}_1 \]

\[ \bar{x}_2 \]

\[ \bar{a} \]

\[ \bar{x}_1 \]

\[ \bar{x}_2 \]

\[ \bar{a} \]

\[ \bar{x}_1 \]

\[ \bar{x}_2 \]

\[ \bar{a} \]

\[ \bar{x}_1 \]

\[ \bar{x}_2 \]

\[ \bar{a} \]

\[ \bar{x}_1 \]

\[ \bar{x}_2 \]

\[ \bar{a} \]

\[ \bar{x}_1 \]

\[ \bar{x}_2 \]

\[ \bar{a} \]

\[ \bar{x}_1 \]

\[ \bar{x}_2 \]

\[ \bar{a} \]

\[ \bar{x}_1 \]

\[ \bar{x}_2 \]

\[ \bar{a} \]

\[ \bar{x}_1 \]

\[ \bar{x}_2 \]

\[ \bar{a} \]

\[ \bar{x}_1 \]

\[ \bar{x}_2 \]

\[ \bar{a} \]

\[ \bar{x}_1 \]

\[ \bar{x}_2 \]

\[ \bar{a} \]

\[ \bar{x}_1 \]

\[ \bar{x}_2 \]

\[ \bar{a} \]

\[ \bar{x}_1 \]

\[ \bar{x}_2 \]

\[ \bar{a} \]

\[ \bar{x}_1 \]

\[ \bar{x}_2 \]

\[ \bar{a} \]

\[ \bar{x}_1 \]

\[ \bar{x}_2 \]
Lecture II

Embedding Calculi and Languages.

- Calculi
  - λ-Calculus [Milner 89]
  - Logic Nets [Abramsky 91]

- Programming Languages
  - Procedural Languages [Welford 83]
  - Functional Languages
  - Logic Languages

- Machines
  - Turing Machines
  - SECR and other abstract machines
  - Various automata

- Representation Term
  - Red by Race
  - Deciding Implication
Prologue to λ-Calculus Embedding.

- Functions as processes: [Miller 90] [Milner 89]

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

Reducing Higher-Order Interaction to Name Passing.

\[(\lambda x. M) \cdot N\]

becomes:

\[\lambda x. M \cdot N\]

\[P \quad Q \quad R \quad Q \quad R \quad P^\prime \quad Q^\prime \quad P^\prime \quad Q^\prime \quad P^\prime \quad Q^\prime \]
Encoding and its Behaviour (1)

\[ [\lambda x].u = \bar{x}u \]

\[ ([\lambda x].M)n = u(xz).C(M)z \]

\[ (M\bar{N})n = (\bar{x}c) (\bar{C}M)C/\bar{z}C\bar{u} \downarrow [c = N] \]

where \([x = N] \downarrow \bar{x}c.\bar{C}M\bar{D}z.\bar{A}_{x = N} \]

Notice, with No either a variable or an abstraction:

\[ [[NaNi...Nn]u] \]

\[ \downarrow \bar{c} (\bar{C}C) [N] \bar{C} (\bar{A}(\bar{A}(\bar{A}(\bar{A}(\bar{C}c, c, N)))) | \bar{A}(c, c, N)) \]

\[ \downarrow \bar{C} \bar{N} \bar{M} \bar{N} \bar{D}z. \bar{A}_{x = N} \]

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

\[ (\forall x) (\forall NaNi...Nn)u \]

\[ \downarrow \bar{c} \bar{M} \bar{D}z \]

With \( n \geq 0, m \geq 0 \) is (BIP) and (FETCH),

which again reduces to the same form.

(If terminates when \( n = 0, No \) is an abstraction.)

Encoding and its Behaviour (2)

Dynamic:

\[ \bar{x}c.\bar{C}M\bar{D}z.\bar{A}_{x = N} \]

APP \([\lambda x].M)C \bar{A}(c, c, N) \]

\[ \rightarrow \bar{c} (\bar{C}C) [N] \bar{C} \bar{A}(\bar{A}(\bar{A}(\bar{A}(\bar{C}c, c, N)))) \]

\[ \downarrow \bar{C} \bar{N} \bar{M} \bar{N} \bar{D}z. \bar{A}_{x = N} \]

(Fetch) \([\bar{x}N]u \mid x = M \]

\[ \rightarrow \bar{c} \bar{M} \bar{D}z \]

It may also terminate where \( \bar{c} \bar{M} \bar{D}z \) and \( y \notin \{x \} \bar{D}z \)
Basic Synthetic Properties (v) 

By the preceding discussion we immediately know: 

Prop 
\[ (\text{IM})_u \rightarrow p_1 \text{ and } (\text{IM})_u \rightarrow p_2 \]

implies 
\[ p_1 \vdash p_2. \]

We also note: 

Prop 
1. \( (\text{IM})_u \text{ Arg}(c_1, c_2, N), (z = N) \text{ are always }\)

pointed.

2. \( (\text{IM})_u \rightarrow p \text{ and } p \rightarrow \emptyset \) implies

\[ p \vdash\emptyset. \]
Basic Syntactic Properties (2)

Lemma

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

$$(\forall C)[C[c, b][c, b] \rightarrow c b] / !c \cdot P]$$

$\Rightarrow C[P_c][P_c] - [P_c]$$

where $P \equiv Q \iff P \equiv Q \land (P \Rightarrow Q)$$

Lemma: $\Rightarrow Q$.

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

$C[cor][c, P] - [c, P]$$

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

Basic Syntactic Properties (2)

Proof of Lemma (outline):

Let $R$ be the relation between two processes as given above, taking them modes $\Xi$. Then $R$ is a weak bisimulation, so it is easy to prove.

To show $P \equiv Q$ implies $(P \Rightarrow Q)$, we say the number of negative occurrences of $c$ on the left hand side process, its Index. Then we show:

Claim: Let $P \equiv Q$ and $P \Rightarrow P'$ with $P'$ index $n$. Then either (1) $P'$ has $n+1$ index and $E Q \Rightarrow Q$ and $P \equiv Q$, or (2) $P'$ has one less index than $P$, and $P \equiv Q$. In particular if index is 0 only (1) holds. $

This is easy syntactic reasoning. Then we can also show $Q \Rightarrow Q \Rightarrow P \Rightarrow P'$ and $P \equiv Q$ (which is easier).
Basic Syntactic Properties (4)

Now the latter shows $Q \Rightarrow P$. On the other hand, suppose $P$ and let $P$ have the index, say, $n$. We show $M \equiv M'$. $Q \rightarrow^* Q$ s.t. $P \equiv Q$ where $P \equiv Q$ by induction on $n$. For $n=0$ this is obvious. If this holds for $n=\overline{k}$, take $Q \rightarrow^k Q$ with $P \equiv Q$ and $P \equiv Q$. Then $P$ has an co-infinite path of reduction $P \rightarrow P' \rightarrow P'' \rightarrow \cdots$. 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 $P_0$ we have $Q \rightarrow Q'$ and $P_0 \equiv Q'$, as required. □

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

Basic Syntactic Properties (5)

Prop

1. $M \rightarrow M' \Rightarrow [\lambda M_n.] \rightarrow P \approx \lambda [\lambda M_n.]$
2. $[\lambda M_n.] \rightarrow P \Rightarrow M \rightarrow M' \wedge P_{\alpha\beta} \equiv [\lambda M_n.]$

Proof:

1. In $[\lambda M_n.]$ only (app) can take place, i.e. we have $M = (\lambda x. M_0) M_{i-1} (M_n)$ and $[\lambda M_n.] \rightarrow^* ([\lambda x. M_0] \rightarrow M_{i-1} (M_n))$

$\approx \lambda [\lambda M_n.]$ $M_{i-1} (M_n)$

(safely assuring $x$ can be free only in $M_n$).

But $\approx \lambda \equiv \lambda$ hence done.

2. Similar to (1).
Basic Syntactic Properties (5)

Note: \( \frac{a}{c} \leq \frac{b}{d} \) is proved as:

1. If \( \Gamma \vdash a \) and \( \Theta \vdash b \) obviously \( \Gamma \vdash \frac{a}{c} \).
2. If \( \Gamma \vdash \frac{a}{c} \) and \( \Theta \vdash b \), then let the finite reduction path of \( P \)
   \[ P \rightarrow P' \rightarrow P'' \ldots . \]
   \( P' \equiv c \) we are done. If not, by the confluence property we reached before,
   there is \( R \) s.t.
   \[ p' \xleftarrow{R} p \]
   \[ p \xleftarrow{b} q \]

Now we can repeat the same argument between \( P' \) and \( R \), i.e. this way for all new we have \( \Theta \vdash b \), as required. //

Proposition 4: \( M \) is closed,

\[ \Gamma \Omega \cup \Gamma \vdash a_{ub} \Rightarrow \Gamma \vdash b \] for some \( b \).

Proof: Because, then, \( P \) has a form:

\[ (v_0) [(v_1 \vdash \ldots \vdash (v_n = M, D)] \ldots (v_n = M, D) \]

for some \( n \geq 0 \). (because \( c_{uvp} \) and \( c_{uvpr} \) are the only possibilities. D.

Corollary (of the preceding proposition)

\[ M \downarrow \Downarrow (\Gamma \Omega \cup \Gamma \vdash a_{ub}) \Rightarrow \Gamma \vdash b_{ub} \]

Remark: Firstly \( \Gamma \Omega \cup \Gamma \vdash a_{ub} \Rightarrow \Gamma \vdash b_{ub} \) too.
Lecture III

The Asynchronous Calculus and Combinators.
Core Calculus (1)

- Asynchronous calculus (HTRF, Boudol 93...)

\[
P ::= ax.P \mid ab \mid P \sqcup P \sqcap (ax.P) \mid \emptyset \mid !ax.P
\]

\[
\begin{cases}
ax.P \mid ab & \rightarrow P \sqcup \emptyset \\
!ax.P \mid ab & \rightarrow !ax.P \mid P \sqcup \emptyset
\end{cases}
\]

- Embedding of synchronous calculus.

\[\begin{array}{c}
\text{Asynchronous Observability.} \\
\text{observer} \leftrightarrow \text{system}
\end{array}\]

Core Calculus (2)

(2W) \[\emptyset \rightarrow \frac{ab}{ab}\] (sender sends a "message").

Proof of \(\mathcal{I}(c) \approx \emptyset\).

\[\mathcal{O} = \{(\mathcal{I}(c), Tc, b, \emptyset | Tc, b)\}\]

(Model of \(\emptyset\)).

\[\begin{array}{c}
\mathcal{I}(c) \mid R
\end{array}\]

Remark: Obviously \(\mathcal{I}(c) \approx \emptyset\).
Combinators for Concurrency

* Can we reduce the syntax further?

\[ P ::= \alpha x.P \mid \lambda b.P(\lambda a.M) \mid a \mid \text{var} \]

Or can we do Ti-calculus without

* But we CAN do \( \lambda \)-calculus without

* term passing....

\[ \begin{cases} \alpha x. MN = S(\alpha x. M)(\alpha x. N) \\ \alpha x. M = K M \quad x \notin \text{FV}(M) \\ \alpha x. x = I \quad (= \text{SKK}) \end{cases} \]

Thm: \((\alpha x. M)N \rightarrow^* MN(x)\).

Dynamics of SK2-combinators:

\[ \begin{align*} \alpha x. M & \quad \rightarrow \quad K M \quad \text{and} \quad x \notin \text{FV}(M) \\ \alpha x. x & \quad \rightarrow \quad I \quad (= \text{SKK}) \end{align*} \]

* Universality: Conjecturing completeness.

\[ \forall x_1 \ldots x_n . \forall F \forall x_1 \ldots x_n . \exists M . \]

\[ M x_1 \ldots x_n \rightarrow^* F x_1 \ldots x_n . \quad \text{(Any functional constant is definable.)} \]

\[ (F x_1 \ldots x_n ) \text{ is a polynomial over } x_1 \ldots x_n . \]
Combinators for Concurrency (3).

- We obtain:

\[ P ::= \mathcal{C}^{\pm}(\mathcal{E}) \mid P \mathcal{I} Q \mid (\text{ms}) P (\mathcal{E}) \mid !P \]

and further:

\[ P ::= \mathcal{C}^{\pm}(\mathcal{E}) \mid P \mathcal{I} Q \mid (\text{rs}) P \mid \bot \]

with dynamics:

\[ \Rightarrow \]

and universality: Any (decidable) interactive constant behaviour is definable....
Some Combinators.

Notation.

\[ M^{+}(abc) \equiv ab \]
\[ F^{-}(abc) \equiv ax.bx \]
\[ \Delta(abc) \equiv ax.(bx|cx) \]
\[ \Lambda(abc) \equiv ax.b \]
\[ S(abc) \equiv ax.FW(cbs) \]
\[ B_L(ab) \equiv ax.FW(ab) \]
\[ B_R(ab) \equiv ax.FW(cb) \]

\* write \( C(ce), C^{+}(ce), C(ce), \ldots \)
Analysis of Prefix (a).

Definition. Given $a, x, P$ define $\alpha x. P$ as follows.

$P := \alpha x. P \mid a b / \text{cop} \mid \emptyset$

Cases:

1. $\alpha x. (P(a)) \equiv \alpha x. (\text{Drac}_c \mid D x_1 \mid P / \alpha x. \emptyset)$
2. $\alpha x. (\text{cop} a) \equiv (a) \alpha x. P$
3. $\alpha x. \emptyset \equiv \emptyset \alpha x. P$

Inductive Cases:

1. $\alpha x. \text{Fun}(x) \equiv \alpha x. (\text{Fun}(x \mid \text{Fun}(x)))$
2. $\alpha x. \text{Fun}(x) \equiv (x) \alpha x. (\text{Fun}(x) \mid \text{Fun}(x))$
3. $\alpha x. \text{Fun}(x) \equiv (x) \alpha x. (\text{Fun}(x) \mid \text{Fun}(x))$

Analysis of Prefix (c).

\[ a \Rightarrow \begin{align*}
\begin{array}{c}
P \quad Q \\
\hline
\end{array}
\end{align*}
\]
Analysis of Prefix (3)

Analysis of Prefix (4)

[Diagram of molecular structures and reactions]
Analysis of Prefix (c)

**Proposition** (Prefix Theory \(4^{th}\).

For any \(a, x,\) and \(P\)

(i) \(a^x. P\) is \(a\)-pointed.

(ii) \(a^x. P \upharpoonright \bar{a}b \rightarrow \approx P_{cbx}\).

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

\(c^x. P \upharpoonright \bar{a}b \rightarrow \approx P_{cbx} \quad i=1,2.

Then

\[ ccx, (Dcax, (c^x. P \upharpoonright c^x. P) \upharpoonright \bar{c}b) \rightarrow \rightarrow \rightarrow \approx (P \to P_{cbx} ) \]

But \(s^x \leq \approx \) hence done. \(\Box \)

**Corollary.** (Synchronous prefix \(4^{th}\).

For any \(a, x, y, P,\) and \(Q,\) there exist terms \(a^x. P\) and \(a^y. Q\) s.t.

(i) \(a^x. P\) is \(a\)-pointed and \(a^y. Q\) is \(a\)-pointed.

(ii) \((a^x. P) \upharpoonright a^y. Q \rightarrow P_{cbaQ}\).

(iii) Both are polynomials from atomic \(y\).

**Proof**

Take e.g. \(a^x. P \equiv a^x. (c \upharpoonright c^x. P)\) and \(a^y. Q \equiv (a^x. (c^x. P \upharpoonright \bar{a}y. Q)).\) Then

\(a^x. P \upharpoonright a^y. Q \rightarrow a^2 \to P_{cbxQ}\).

Runtedness is obvious. \(\Box\)
Corollary. (polyadic name yessy).

For any $a, e, v, P, Q$, there exist terms $a, e, v, P$ and $e, v, Q$ such that

(i) $a, e, v, P$ is $e$-pointed and $e, v, Q$ is $a$-pointed,

(ii) $a: (e, v, P)$ $\vdash$ $e, v, Q$ $\rightarrow$ $\approx P(e, v, P)$ $|$ $Q$,

(iii) Both are polynomials from $a$ to $x_k$.

Proof.

Using the greedy encodings

$a: (e, v, P)$ $\equiv$ $a, e, v, x_1, \ldots, x_n, P$  
$e, v, Q$ $\equiv$ $v, e, v, x_1, \ldots, x_n, Q$  

then use $\approx$ relying on pointedness. $\square$

Analysis of Prefix (b).

Corollary. (branching prefix).

For any $a, e, v, e, v, P, R, Q$, there exist terms $a: (e, v, P) \& (e, v, R)$ and $e, v, Q$ such that

(i) $a: (e, v, P) \& (e, v, R)$ is $e$-pointed and $e, v, Q$ is $a$-pointed,

(ii) $a: (e, v, P) \& (e, v, R)$ $\vdash$ $e, v, Q$ $\rightarrow$ $\approx P(e, v, P)$ $|$ $Q$

$\rightarrow$ $\approx R(e, v, R)$ $| Q$

$\approx (e, v, P) \& (e, v, R)$ $| v, m_1, (e, v, Q)$

$\rightarrow$ $\approx (e, v, P) \& (e, v, R)$ $| v, m_2, (e, v, Q)$

$\rightarrow$ $\approx P(e, v, P)$ $| Q$

(iii) Both are polynomials from $a$ to $x_k$.

Proof.

Use the previous corollary. $\square$
Analysis of Replicator a)

\[ \forall x. \exists y \in b \]

Notation:

Given \( C(c) \),

\[ \forall C(c) \equiv \forall C(c) \quad (+1) \]
\[ \# C(c) \equiv \# C(c) \quad (+2) \]
\[ \# ! C(c) \equiv \# ! C(c) \quad (+3) \]
\[ N(c) \equiv ! \exists x. C(c) \quad (+1) \]

\[ \implies \quad \text{Diagram} \]

\[ \bullet \]
\[ \text{Header} \]
\[ \text{Distributor of Names} \]
\[ \# C_1 \quad \# C_2 \quad \ldots \quad \# C_m \]

\[ c_1 \ldots c_m \triangleright (C_1(\overline{a_1}), C_2(\overline{a_2}), \ldots, C_m(\overline{a_m})) \]

* So we have 27 atoms, whose instantiations are atomic agents.
Embedding (1)

Definition.

\[(\forall x. P)^* \equiv \forall x. P^*\]
\[\exists b^* \equiv \exists b\]

(3) \(P^* \equiv \forall \exists P^*\)

(\(\Pi \Theta\))^* \equiv P^* \downarrow \Theta^*

\[\emptyset^* \equiv \emptyset\]

(1\(\forall x. P\))^* \equiv \exists x^* P^*

Embedding (2)

Lemma

1. \(P \equiv \Theta \Rightarrow \exists x. P \equiv \exists x. \Theta\)
2. \(P \equiv \Theta \Rightarrow \forall x. P \equiv \forall x. \Theta\)

Theorem

1. \(P \Rightarrow \Theta \Rightarrow P^* \Rightarrow \Theta^*\)
2. \(P^* \Rightarrow P^* \Rightarrow \exists P^*. P^* \equiv P^* \land P \Rightarrow P^*\)
3. \(P \equiv \Theta \Leftrightarrow P^* \equiv \Theta^*\)

Proof: For (3) prove \(P \equiv P^*\) using the above lemma. \(\square\)
Generators (1)

Notation. Given a set of terms \( \mathcal{P} \), we write \( \overline{\mathcal{P}} \) for the least set of terms with \( \emptyset \) such that:

\[
P \in \overline{\mathcal{P}} \Rightarrow P_0 \in \overline{\mathcal{P}} \quad (c \text{ is renaming})
\]

\[
P, Q \in \overline{\mathcal{P}} \Rightarrow PQ \in \overline{\mathcal{P}}
\]

\[
P \in \overline{\mathcal{P}} \Rightarrow cNP \in \overline{\mathcal{P}}
\]

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

\[
\forall P \in \mathcal{P}_0 \exists P' \in \overline{\mathcal{P}}. \quad P \approx P'
\]

Generators (2)

Corollary.

\( \mathcal{P}_0 \) has a finite set of generators.

Proof: Direct from (the proof of) the embedding theorem. \( \square \)
What We Have Gotten...

\[ M \cdot i = \chi M | MN | z \]

... is a BNF:

\[ P \cdot i = G_{\omega c} \mid P \mid Q \mid \omega \mid P \]

which gives a self-contained

\[ \text{onverse of concurrent conjuring.} \]

What is sleeping underneath?
Notes on Interaction Behaviour on CC.

From Applicative Behaviour to SK2.

\[ S \; x_1 \ldots x_n \to F(x_1 \ldots x_n) \]
\[ \downarrow \]
\[ (\lambda x_1 \ldots x_n. M) \; x_1 \ldots x_n \to M \]
\[ \downarrow \]
\[ (\lambda x. M) \; x \to M \]
\[ \downarrow \]
\[ S, K \to \Sigma \]

From Interactive Behaviour to CC.

\[ \text{Branching + Replication.} \]
\[ \downarrow \]
\[ \alpha \cdot P \; x \to P \; \! x \cdot \alpha \]
\[ \downarrow \]
\[ \downarrow \]
\[ \mu \]
\[ \Sigma \]
\[ \text{P} \]
\[ \ldots \]
Lecture IV

Types for Mobile Processes

Types for Mobile Processes (a)

- Types for functions:

\[ \lambda x. x : \text{Nat} \to \text{Nat} \]

The type \( \text{Nat} \to \text{Nat} \) says the following:

1. The term is composable with \( N : \text{Nat} \) on the right.
2. When composed with \( N : \text{Nat} \), it gives:
   \[ (\lambda x. x) N : \text{Nat} \]
3. It is semantically a function from \( \text{Nat} \) to \( \text{Nat} \).
Types for Mobile Processes (4)

- What are types for processes?

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

\[
\text{ex. } a(x), b[x]
\]

when composed with \(a[x]\), it will emit \(b[x]\).

Types for Mobile Processes (5)

- Current status:
  
  - Sorting [Milner 82]
    
    - Single notion of types.
    
    - Structure of "move carrying".

  - Type Inference [Wadler & Honda 93]
    
    - Type inference (Honda 93, Wadler 94, Honda 96). ...

  - Types with dynamic structure.
    
    \[
    (\mathcal{E}C)(\mathcal{E}\mathcal{E}(\tau \cdot C.X.(\mathcal{E}(C.X.P)))) \quad (\text{Yoshida} 95)
    \]

  - Semantics. \(\subseteq\)

- We still understand LITTLE.
- But connectives to various cases (\(\subseteq\) and semantics) are emerging.
- What is the "spectrum of types" in this setting?
Sorting (2)

(3) "Type"-presentation.

\[
\{ s_1^a, s_2^b, s_3^c \} \quad s_1 \rightarrow s_2 s_3
\]

\[
\{ s_1^a, s_2^b, s_3^b \} \quad s_1 \rightarrow s_2, \quad s_2 \rightarrow s_3 s_4
\]

\[
\begin{align*}
& a : (c(d_2)), \\& e : (d(id_1)), \\& b : d_1, \\& c : d_2
\end{align*}
\]

\[
\begin{align*}
& a : (C(d_3)), \\& b = d
\end{align*}
\]

Types: \( d ::= x \mid (d_1 \ldots d_n) \mid \text{ex. } a \).

Typing: \( a_i : d_i, a_1 : d_1 \ldots a_n : d_n \quad P, A, B, \ldots \)

Equality: \( (d_1 \ldots d_n) \overset{\sim}{\rightarrow} d_i \);
\[
\begin{align*}
& x \overset{\sim}{\rightarrow} 1 \quad \text{ex. } a \\
& x \overset{\text{ex. } a}{\rightarrow} \text{B}
\end{align*}
\]

\( a \approx 0 \) when they are bisimilar w.r.t. \( \approx \).
(4) Type Inference System $\Gamma \vdash P$.

\[
\Gamma \vdash P \\
\Gamma \vdash a : a_1 : ... : a_n. P \\
\Gamma \vdash (a_1 : ... : a_n) P
\]

(5) Basic Syntactic Properties.

Proof:
1. $\Gamma \vdash P \Rightarrow \text{FN}(\Gamma) \subseteq \text{FN}(P)$.
2. $\Gamma, a : a \vdash P \wedge \text{FN}(P) \Rightarrow \Gamma \vdash P$.
3. $\Gamma \vdash P \land P_0$ is a subterm of $P$ \\
\quad $\Rightarrow \exists \theta. \Delta \vdash P_0$.
4. $\Gamma \vdash P \land P \equiv Q \Rightarrow \Gamma \vdash Q$.
5. $\Gamma \vdash a : a_1 : ... : a_n. P \mid \text{CN-vm} 2. Q$ \\
\quad $\Rightarrow m = a_m$.
6. (Subject Reduction) \\
\[
\Gamma \vdash P \land P \Rightarrow P' \Rightarrow \Gamma \vdash P'.
\]

(6) (Subject Reduction)
Sority (5)

\[ D_f \in \text{Err} \quad \text{iff} \quad P \neq P' \equiv (\forall x)(a(x,x), P, \pi(x,x,0, R)) \quad n \neq a_n. \]

**Theorem**

\[ P \vdash P \Rightarrow P \in \text{Err}. \]

Ex. λ-encoding.

\[ \lambda x. \lambda y. x = x \quad (\lambda x M) \quad \lambda x. (\lambda M D) \]

\[ (\lambda M N) a = ((\lambda c. a)(\lambda M D C) \mid \lambda c. c a) \mid \lambda c. c a) (\lambda M D) \]

If \( M \) has free variables \( x_1, \ldots, x_n \), we have:

\[ \llbracket M \rrbracket a = x_1 : \text{Var}, \ldots, x_n : \text{Var}, u : \text{Body} \]

where \( \text{Var} : \text{Body} \) and \( \text{Body} : \text{ax.} \cdot (x) x \).

Lecture V

Symmetries in Processes.

(from slides for c seminar)