.

Concurrency 2

Functions vs Processes

$\Longrightarrow$ Interaction

Jean-Jacques Lévy

jeanjacqueslevy.net/dea

Concurrency $\Rightarrow$ Non-determinism

Suppose x is a global variable. At beginning, $x = 0$

Consider

$P = [x := x + 1; x := x + 1 \mid\mid x := 2 * x]$

after P, then x may have several values ($x \in \{2, 3, 4\}$)

Hence P is not a function from memory states to memory states.

In concurrent programming, execution is not deterministic since it is upto an external agent (the scheduler).

Let $\Sigma = \text{Variables} \mapsto \text{Values}$ be the set of memory states.
Let $[\![P]\!]$ be the meaning of P.

A concurrent program is not a (partial) function from memory states to memory states. $[\![P]\!] \notin \Sigma \mapsto \Sigma$.

A concurrent program is a relation on memory states. $[\![P]\!] \in \Sigma \mapsto 2^{\Sigma}$.

Concurrency $\Rightarrow$ Interaction

Consider

$P = [x := 1]$
$Q = [x := 0; x := x + 1]$

P and Q are same functions on memory states : $\sigma \mapsto \sigma[1/x]$

However

after $P \mid\mid P$, then $x \in \{1\}$
after $P \mid\mid Q$, then $x \in \{1, 2\}$

A semantic (meaning) is compositional iff $[\![P]\!] = [\![Q]\!]$ implies $[\![\ C[P]\ ]\!] = [\![\ C[Q]\ ]\!]$ for any context $C[\ ]$.

In previous example, in any compositional semantics, $[\![P]\!] \neq [\![Q]\!]$.

Conclusion

P and Q are not equivalent processes.

Concurrency $\Leftrightarrow$ Termination

Concurrent processes are often non terminating.

An operating system never terminates ; same for the software of a vending machine, or a traffic-light controler, or a human, etc.

A process P is a set of pairs (f_i, P_i), atomic action and a derivative process. It starts by performing f_i and then becomes process P_i.

Atomic steps usually terminate.

Roughly speaking, let $\mathcal{P}$ be the set of processes. Then $\mathcal{P} = 2^{(\Sigma \mapsto \Sigma) \times \mathcal{P}}$

Is this equation meaningful ? Answer : Scott's domains, denotational semantics. Remarkable and difficult theory of Plotkin (Scott's powerdomains 1976).

We try the simpler theory of labeled transition systems.

Labeled Transition Systems

A LTS is triple $(\mathcal{P}, \mathcal{A}ct, \mathcal{T})$ where

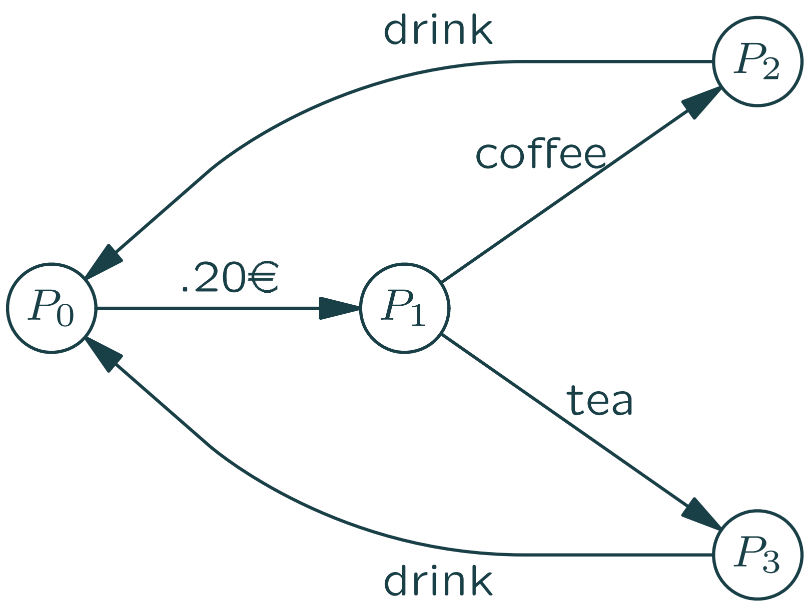
- $\mathcal{P}$ is the set of processes
- $\mathcal{A}ct$ is the set of actions
- $\mathcal{T} \subseteq \mathcal{P} \times \mathcal{A}ct \times \mathcal{P}$ is the transition relation

Let write $P \xrightarrow{\mu} Q$ for $(P, \mu, Q) \in \mathcal{T}$.
Read P interacts with environment with action μ, then becomes Q.

Q is a derivative of P if $P = P_0 \xrightarrow{\mu_1} P_1 \xrightarrow{\mu_2} P_2 \cdots \xrightarrow{\mu_n} P_n = Q$ for $n \geq 0$.

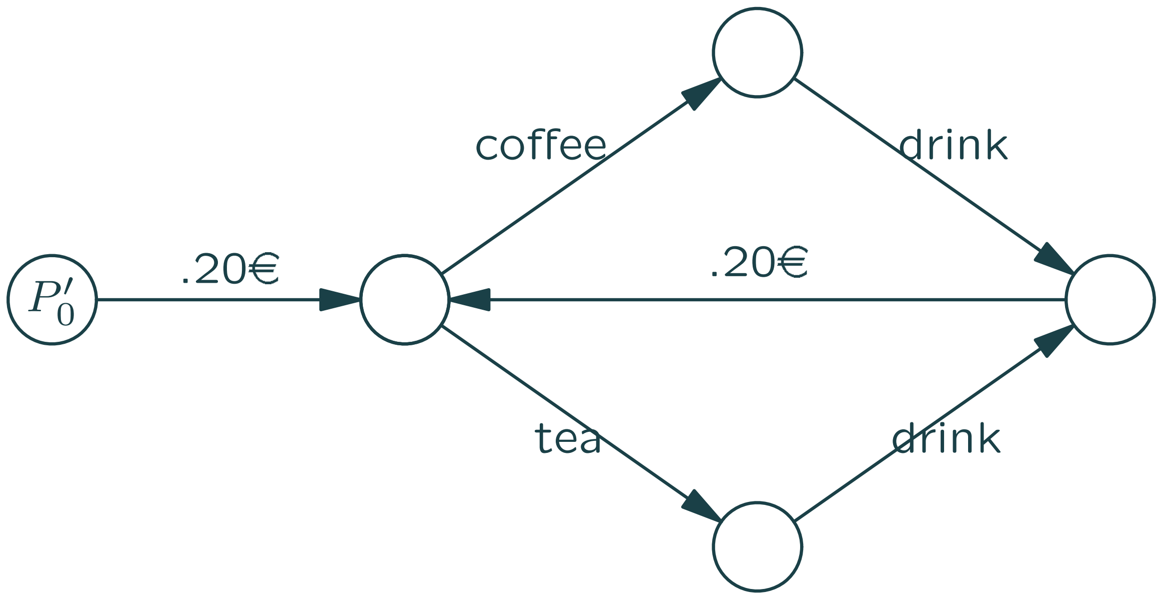
Example (1/3)

A vending machine for coffee/tea. At beginning, P_0



Example (2/3)

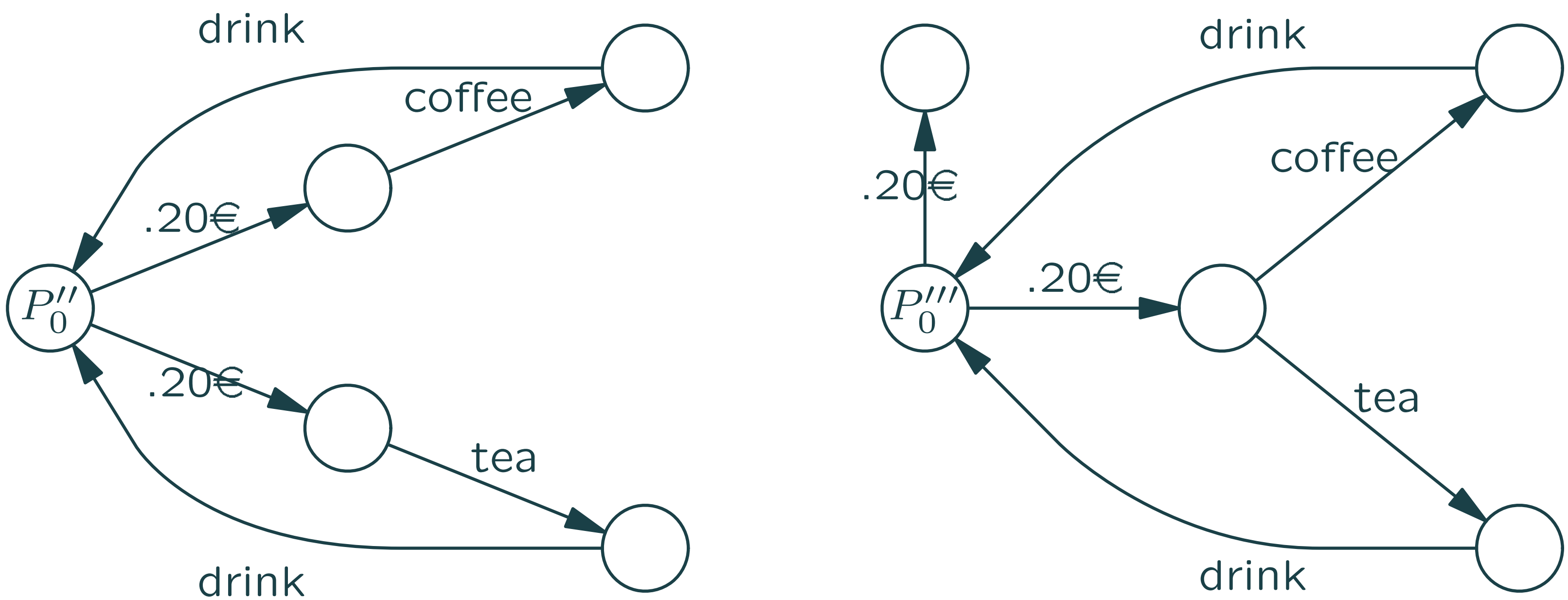
A different vending machine for coffee/tea. At beginning, P'_0



Is this LTS equivalent to previous one ?

Example (3/3)

Two new vending machines P_0'' and P_0'''



Why these LTS are not equivalent to previous ones ?

Concurrency $\Leftrightarrow$ Automata (1/2)

Let abstract $\mathcal{A}ct$ (actions) as an alphabet $\{a, b, c, \ldots\}$.
($\mathcal{A}ct$ may be infinite)

Then LTS look like automata (with possibly infinite number of states).

Consider the language of traces.

Let $P = P_0 \xrightarrow{\mu_1} P_1 \xrightarrow{\mu_2} P_2 \cdots \xrightarrow{\mu_n} P_n$ $(n \geq 0)$, then

$trace(P = P_0 \xrightarrow{\mu_1} P_1 \xrightarrow{\mu_2} P_2 \cdots \xrightarrow{\mu_n} P_n) = \mu_1 \mu_2 \cdots \mu_n$

We say that $\mu_1 \mu_2 \cdots \mu_n$ is a trace of P

Let $\mathcal{T}races(P) = \{w \mid w \text{ is a trace of } P\}$

Concurrency $\Leftrightarrow$ Automata (2/2)

In previous examples, write k for coffee, t for tea, c for $.20e$, d for drink.

$\mathcal{T}races(P_0) = prefixes((c(k+t)d)^*)$,
$\mathcal{T}races(P_0') = prefixes(c((k+t)dc)^*)$,
$\mathcal{T}races(P_0'') = prefixes((ckd+ctd)^*$,
$\mathcal{T}races(P_0''') = prefixes((c+c(k+t)dc)^*)$,

Exercice 1 Show $\mathcal{T}races(P_0) = \mathcal{T}races(P_0') = \mathcal{T}races(P_0'') = \mathcal{T}races(P_0''')$

However, P_0 and P_0' seem equivalent
but both P_0'' and P_0''' look distinct from P_0.

Why ?

After c, the set of choices are distinct in P_0 and P_0''.
Coffee button is always enabled in P_0, but not in P_0''.
Same for tea button.

In P_0''', both tea and coffee may be disabled after c.

Simulation – Bisimulation

Definition 1 Q simulates P (we write $P \lesssim Q$) if whenever $P \xrightarrow{\mu} P'$, there is Q' such that $Q \xrightarrow{\mu} Q'$ and $P' \lesssim Q'$.

Definition 2 P strongly bisimilar to Q (we write $P \sim Q$) if whenever

- $P \xrightarrow{\mu} P'$, there is Q' such that $Q \xrightarrow{\mu} Q'$ and $P' \sim Q'$.
- $Q \xrightarrow{\mu} Q'$, there is P' such that $P \xrightarrow{\mu} P'$ and $P' \sim Q'$.

Graphically,

Exercice 2 Give intuition for $P_0 \lesssim P_0''' \lesssim P_0$

Exercice 3 Give intuition for $P_0 \sim P_0'$, $P_0 \not\sim P_0''$, $P_0 \not\sim P_0'''$

Definition of bisimulation (1/3)

Definition 3 A bisimulation is a binary relation $\mathcal{R}$ on processes such that $P \,\mathcal{R}\, Q$ implies whenever

- $P \xrightarrow{\mu} P'$, there is Q' such that $Q \xrightarrow{\mu} Q'$ and $P' \,\mathcal{R}\, Q'$.
- $Q \xrightarrow{\mu} Q'$, there is P' such that $P \xrightarrow{\mu} P'$ and $P' \,\mathcal{R}\, Q'$.

An alternative definition for strong bisimulation is :

Definition 4 Let $\sim \; = \; \cup\{\mathcal{R} \mid \mathcal{R} \text{ is a bisimulation}\}$

Proposition 5 $\sim$ is an equivalence relation.
(reflexive, symmetric, transitive)

Exercice 4 Show above proposition.

Exercice 5 What is the least bisimulation ?

Definition of bisimulation (2/3)

First definition of bisimulation is circular. To make it clear, better is to return to standard theory on fixpoints in complete lattices.

A complete lattice $\mathcal{D}$ is any set with

- a partial ordering $\preceq$ (reflexive, transitive, antisymmetric)
- for any subset $E \subseteq D$, there is an upper bound $\cup E$ and a lower bound $\cap E$ in D.

Examples : $2^{\mathcal{P}}$ with $\subseteq$, $2^{\mathcal{P}\times\mathcal{P}}$ with $\subseteq$, etc.

f function $D \mapsto D$ is monotonic iff $x \preceq y$ implies $f(x) \preceq f(y)$.

Theorem 6 [Tarski] In a complete lattice D, any monotonic function f has a least fixpoint $\mathtt{lfp}(f)$ and greatest fixpoint $\mathtt{gfp}(f)$.

Moreover $\mathtt{lfp}(f) = \cap\{x \mid f(x) \preceq x\}$ and $\mathtt{gfp}(f) = \cup\{x \mid x \preceq f(x)\}$

Exercice 6 Prove it.

Definition of bisimulation (3/3)

Proposition 7 $\sim$ is the largest relation $\sim'$ such that $P \sim' Q$ implies whenever

- $P \xrightarrow{\mu} P'$, there is Q' such that $Q \xrightarrow{\mu} Q'$ and $P' \sim' Q'$.
- $Q \xrightarrow{\mu} Q'$, there is P' such that $P \xrightarrow{\mu} P'$ and $P' \sim' Q'$.

Proof : Consider the complete lattice of binary relations on $\mathcal{P}$ with $\subseteq$.
Take $P\ f(\mathcal{R})\ Q$ defined as whenever

- $P \xrightarrow{\mu} P'$, there is Q' such that $Q \xrightarrow{\mu} Q'$ and $P'\ \mathcal{R}\ Q'$.
- $Q \xrightarrow{\mu} Q'$, there is P' such that $P \xrightarrow{\mu} P'$ and $P'\ \mathcal{R}\ Q'$.

Then f is monotonic, since $\mathcal{R} \subseteq \mathcal{S}$ implies $f(\mathcal{R}) \subseteq f(\mathcal{S})$.

Moreover $\mathcal{R}$ is a bisimulation iff $\mathcal{R} \subseteq f(\mathcal{R})$.

Hence $\sim\ = \cup\{\mathcal{R} \mid \mathcal{R} \subseteq f(\mathcal{R})\} = \mathtt{gfp}(f)$.

Therefore $\sim\ = f(\sim)$ and $\sim$ is largest $\sim'$ such that $\sim'\ = f(\sim')$.

First definition of $\sim$ was correct (just add "largest").

Co-induction

In order to show $P \sim Q$, it is sufficient to show that $P \mathcal{R} Q$ for some bisimulation $\mathcal{R}$.

I.e. $(P \mathcal{R} Q$ for some relation $\mathcal{R}$ such that $\mathcal{R} \subseteq f(\mathcal{R}))$ $\Rightarrow P \sim Q$.

Exercice 7 Show $P_0 \sim P_0'$, $P_0 \not\sim P_0''$, $P_0 \not\sim P_0'''$ in vending machines.

Exercice 8 Give an alternative definition for $\lesssim$.

Exercice 9 Show $P_0 \lesssim P_0''' \lesssim P_0$.

Co-continuity (1/2)

Let D be a complete lattice. Then

f function $D \mapsto D$ is co-continuous iff $f(\cap S) = \cap f(S)$ for any descending chain $S = \{d_1, d_2, \ldots d_n \ldots\}$ where $d_1 \succeq d_2 \succeq \cdots \succeq d_n \succeq \cdots$

Theorem 8 [Kleene] $\mathtt{gfp}(f) = \cap\{f^n(\top) \mid n \geq 0\}$ where $\top$ is maximum element of D.

Consider lattice of binary relations $2^{\mathcal{P}\times\mathcal{P}}$ with $\subseteq$.

Let the graph of derivatives of P be finitely branching, i.e. $\{Q \mid P \xrightarrow{\mu} Q\}$ is finite for any P.

Take $P\ f(\mathcal{R})\ Q$ defined as whenever

- $P \xrightarrow{\mu} P'$, there is Q' such that $Q \xrightarrow{\mu} Q'$ and $P'\ \mathcal{R}\ Q'$.
- $Q \xrightarrow{\mu} Q'$, there is P' such that $P \xrightarrow{\mu} P'$ and $P'\ \mathcal{R}\ Q'$.

Then f is co-continuous.

If the graph of derivatives is finitely branching, then
$\sim\ =\ \cap\{f^n(D) \mid n \geq 0\}$

Co-continuity (2/2)

Exercice 10 Suppose P has a finite graph of derivatives. Give an algorithm for computing its minimal graph of derivatives, i.e. a graph where distinct states are not bisimilar.
$O(n \log n)$ algorithm by Paige and Tarjan,
(analogous of Hopcroft/Ullman algorithm for computing minimal finite automata).

Exercice 11 Suppose P and Q have finite graphs of derivatives. Give an algorithm for testing $P \sim Q$.

Exercices

Definition 9 $\mathcal{R}$ is a bisimulation up-to $\sim$ if $P\,\mathcal{R}\,Q$ implies whenever

- $P \xrightarrow{\mu} P'$, there is Q' such that $Q \xrightarrow{\mu} Q'$ and $P' \sim\mathcal{R}\sim Q'$.
- $Q \xrightarrow{\mu} Q'$, there is P' such that $P \xrightarrow{\mu} P'$ and $P' \sim\mathcal{R}\sim Q'$.

Exercice 12 Let $\mathcal{R}$ is a bisimulation up-to $\sim$. Show $\mathcal{R} \subseteq \sim$.
(by firstly showing that $\sim\mathcal{R}\sim$ is a bisimulation).

Let $\mu^+ \in \mathcal{A}ct^+$ (not empty words of actions)

Write $P \xrightarrow{\mu^+} Q$ if $P = P_0 \xrightarrow{\mu_1} P_1 \xrightarrow{\mu_2} P_2 \cdots \xrightarrow{\mu_n} P_n = Q$ and $\mu = \mu_1\mu_2\cdots\mu_n$ $(n > 0)$.

Exercice 13 Show that following definition of strong bisimulation is equivalent to previous one.

Definition 10 $\mathcal{R}$ is a (strong) bisimulation if $P\,\mathcal{R}\,Q$ implies whenever

- $P \xrightarrow{\mu^+} P'$, there is Q' such that $Q \xrightarrow{\mu^+} Q'$ and $P' \sim\mathcal{R}\sim Q'$.
- $Q \xrightarrow{\mu^+} Q'$, there is P' such that $P \xrightarrow{\mu^+} P'$ and $P' \sim\mathcal{R}\sim Q'$.

History

David Park invented bisimulation as maximal fixpoints. (1975)

Robin Milner wrote a full book on them for CCS. (1979)

Samson Abramsky added them in the lazy lambda calculus. (1984) See also phD of his student Luke Ong.

Davide Sangiorgi did the theory of bisimulation in the pi-calculus. (1990)

Marcelo Fiore et al put them in data types. (1992)

Many people speak now of bisimulations, as a generic names for equivalences on infinite computations.

For instance, Dave Sands and others use them for equivalence of Bohm trees in the lambda-calculus (which I never understood!!).