Formal Techniques for Software Engineering: Process Algebra Operators

Rocco De Nicola

IMT Institute for Advanced Studies, Lucca
rocco.denicola@imtlucca.it

June 2013

Lesson 7

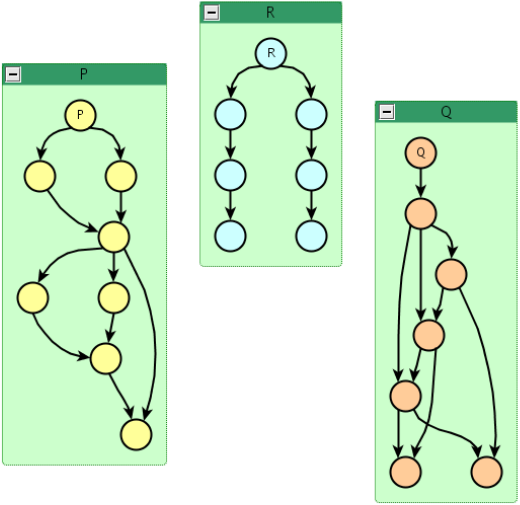
Operators for Processes Modelling

Processes are composed via a number of basic operators

1. Basic Processes
2. Action Prefixing
3. Sequentialization
4. Choice

5. Parallel Composition & Interaction
6. Abstraction (Interaction delimiters)
7. Infinite Behaviours

The General Idea



Some Key Questions

1. Why is structure important?
2. Which ops to write processes?
3. Which ops to compose them in parallel?
4. How to define ops?
5. How to evaluate convenience / expressiveness?

Semantics

Given a *process signature* (i.e. the syntax for writing process expressions), different approaches are available to endow processes with meaning:

1. *Operational semantics*: processes are seen as some sort of abstract machines defined by structural induction;
   traditionally, Milner's **CCS** emphasized the use of this method.
2. *Denotational semantics*: processes are mapped to mathematical objects of a suitable domain by some interpretation function $[\![\cdot]\!]$;
   traditionally, Hoare's **CSP** emphasized the use of this method.
3. *Algebraic semantics*: processes are equated by stating a set of laws (axioms), whence the name *process algebra*;
   traditionally, Bergstra-Klop's **ACP** emphasized the use of this method.

Operational Semantics

To each process expression, an LTS is associated by relying on structural induction to define the meaning of each operator.

Inference Systems

An inference system is a set of inference rule of the form:

$$\frac{p_1, \cdots, p_n}{q}$$

Transition Rules

For each operator op, we have a number of rules alike the one below, where $\{i_1, \cdots, i_m\} \subseteq \{1, \cdots, n\}$ and $E'_i = E_i$ when $i \notin \{i_1, \cdots, i_m\}$.

$$\frac{E_{i_1} \xrightarrow{\alpha_1} E'_{i_1} \quad \cdots \quad E_{i_m} \xrightarrow{\alpha_m} E'_{i_m}}{op(E_1, \cdots, E_n) \xrightarrow{\alpha} C[E'_1, \cdots, E'_n]}$$

The Elegance of Operational Semantics

Abstract machine

Few SOS rules define all the automata that can ever be specified with the chosen operators (processes as states, transitions as transitions).
The set of rules is fixed once and for all. Given any process, the rules are used to derive its transitions.

Structural induction

The LTS of complex systems are defined in terms of the behavior of their components.

A remark

The LTS is the least one satisfying the inference rules.

Rule induction

A property is true for the whole LTS if whenever it holds for the premises of each rule, it holds also for the conclusion.

The Elegance of Operational Semantics

Abstract machine

Few SOS rules define all the automata that can ever be specified with the chosen operators (processes as states, transitions as transitions).
The set of rules is fixed once and for all. Given any process, the rules are used to derive its transitions.

Structural induction

The LTS of complex systems are defined in terms of the behavior of their components.

A remark

The LTS is the least one satisfying the inference rules.

Rule induction

A property is true for the whole LTS if whenever it holds for the premises of each rule, it holds also for the conclusion.

The Elegance of Operational Semantics

Abstract machine

Few SOS rules define all the automata that can ever be specified with the chosen operators (processes as states, transitions as transitions).
The set of rules is fixed once and for all. Given any process, the rules are used to derive its transitions.

Structural induction

The LTS of complex systems are defined in terms of the behavior of their components.

A remark

The LTS is the least one satisfying the inference rules.

Rule induction

A property is true for the whole LTS if whenever it holds for the premises of each rule, it holds also for the conclusion.

The Elegance of Operational Semantics

Abstract machine

Few SOS rules define all the automata that can ever be specified with the chosen operators (processes as states, transitions as transitions).
The set of rules is fixed once and for all. Given any process, the rules are used to derive its transitions.

Structural induction

The LTS of complex systems are defined in terms of the behavior of their components.

A remark

The LTS is the least one satisfying the inference rules.

Rule induction

A property is true for the whole LTS if whenever it holds for the premises of each rule, it holds also for the conclusion.

Atomic Actions

An elementary action of a system represents the atomic (non-interruptible) abstract step of a computation that is performed by a system to move from one state to the other.

Actions represent various activities of concurrent systems:

1. Sending a message
2. Receiving a message
3. Updating values
4. Synchronizing with other processes
5. ...

We have two main types of atomic actions:

- Visible Actions
-

Atomic Actions

An elementary action of a system represents the atomic (non-interruptible) abstract step of a computation that is performed by a system to move from one state to the other.

Actions represent various activities of concurrent systems:

1. Sending a message
2. Receiving a message
3. Updating values
4. Synchronizing with other processes
5. ...

We have two main types of atomic actions:

- Visible Actions
- Internal Actions

Playful digression

10 kinds of persons

There are only 10 kinds of persons in the whole world

- Those who understand binary notation
- Those who don't

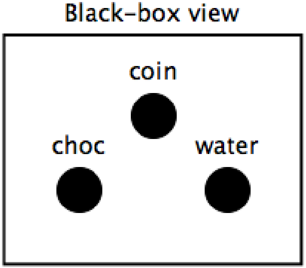
Playful digression

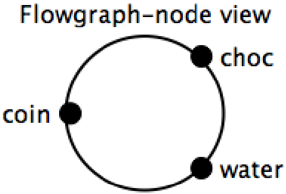
10 kinds of persons

There are only 10 kinds of persons in the whole world

- Those who understand binary notation
- Those who don't

Graphical notation: black-boxes and flowgraphs





Basic Processes

Inactive Process

Is usually denoted by

- *nil*
- 0
- *stop*

The semantics of this process is characterized by the fact that there is no rule to define its transition: it has no transition.

A broken vending machine

$$nil$$

Does not accept coins and does not give any drink.

Basic Processes

Inactive Process

Is usually denoted by

- *nil*
- 0
- *stop*

The semantics of this process is characterized by the fact that there is no rule to define its transition: it has no transition.

A broken vending machine

$$nil$$

Does not accept coins and does not give any drink.

Basic Processes ctd

Termination

Termination is sometimes denoted by

- *skip*
- 1
- *exit*
- $\surd$

that can only perform the special action $\surd$ ("tick") to indicate termination and become *nil*

$$\frac{}{exit \xrightarrow{\surd} stop}$$

A gentle broken vending machine

$$exit$$

Does not accept coins, does not gives drinks but says that everything is ok.

Basic Processes ctd

Termination

Termination is sometimes denoted by

- *skip*
- 1
- *exit*
- $\surd$

that can only perform the special action $\surd$ ("tick") to indicate termination and become *nil*

$$\frac{}{exit \xrightarrow{\surd} stop}$$

A gentle broken vending machine

$$exit$$

Does not accept coins, does not gives drinks but says that everything is ok.

Basic Processes ctd

Action as basic processes

Some calculi consider actions as basic processes.

$$\frac{}{a \xrightarrow{a} stop}$$

A dishonest vending machine: Accepts a coin and stops.

$$coin$$

An alternative

When termination ticks are needed, a slightly different variant can be used:

$$\frac{}{a \xrightarrow{a} \surd}$$

Another dishonest vending machine: Accepts a coin and says it's ok.

$$coin$$

Basic Processes ctd

Action as basic processes

Some calculi consider actions as basic processes.

$$\frac{}{a \xrightarrow{a} stop}$$

A dishonest vending machine: Accepts a coin and stops.

$$coin$$

An alternative

When termination ticks are needed, a slightly different variant can be used:

$$\frac{}{a \xrightarrow{a} \surd}$$

Another dishonest vending machine: Accepts a coin and says it's ok.

$$coin$$

Action Prefixing

Prefixing

For each action μ there is a unary operator

- $\mu.\cdot$
- $\mu \rightarrow \cdot$

that builds from process E a new process $\mu.E$ that performs action μ and then behaves like E.

$$\frac{}{\mu.E \xrightarrow{\mu} E}$$

A "one shot" vending machine

$$coin \rightarrow choc \rightarrow stop$$

Accepts a coin and gives a chocolate, then stops.

Action Prefixing

Prefixing

For each action μ there is a unary operator

- $\mu.\cdot$
- $\mu \rightarrow \cdot$

that builds from process E a new process $\mu.E$ that performs action μ and then behaves like E.

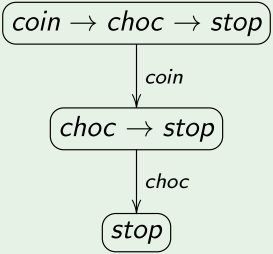
$$\frac{}{\mu.E \xrightarrow{\mu} E}$$

A "one shot" vending machine

$$coin \rightarrow choc \rightarrow stop$$

Accepts a coin and gives a chocolate, then stops.

$coin \rightarrow choc \rightarrow stop$ as LTS



Sequential Composition

Sequentialization

The binary operator for sequential composition is denoted by

- $\_ ; \_$
- $\_ \gg \_$
- $\_ \cdot \_$

If E ed F are processes, process $E; F$ executes E and then behaves like F

$$\frac{E \xrightarrow{\mu} E'}{E; F \xrightarrow{\mu} E'; F} \quad (\mu \neq \surd) \qquad\qquad \frac{E \xrightarrow{\surd} E'}{E; F \xrightarrow{\tau} F}$$

Another "one shot" vending machine

$$coin;\ choc$$

Sequential Composition

Sequentialization

The binary operator for sequential composition is denoted by

- $\_ ; \_$
- $\_ \gg \_$
- $\_ \cdot \_$

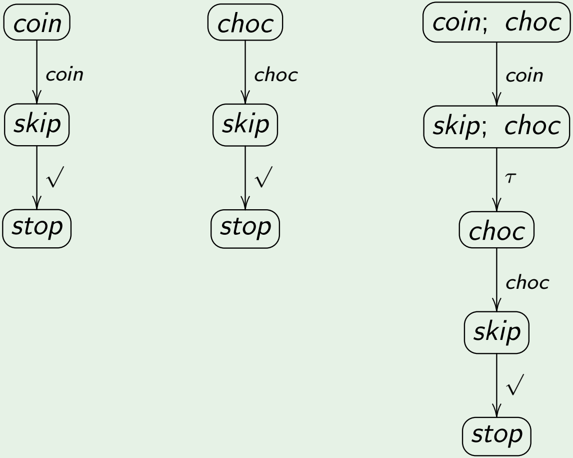
If E ed F are processes, process $E; F$ executes E and then behaves like F

$$\frac{E \xrightarrow{\mu} E'}{E; F \xrightarrow{\mu} E'; F} \quad (\mu \neq \surd) \qquad\qquad \frac{E \xrightarrow{\surd} E'}{E; F \xrightarrow{\tau} F}$$

Another "one shot" vending machine

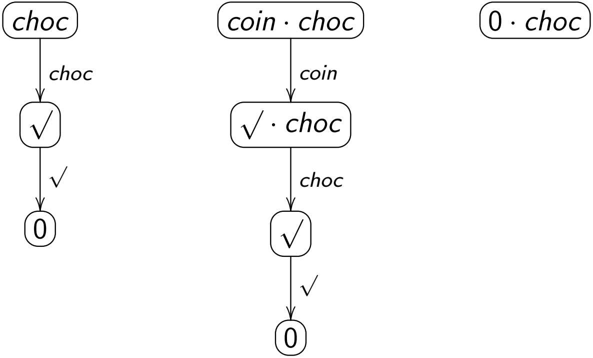
$$coin;\ choc$$

coin; *choc* as LTS



Question Time

Given the syntax $P ::= 0 \mid \surd \mid a \mid P \cdot P$ define proper SOS rules to generate an LTS such that, e.g.



Choice - 1

Nondeterministic Choice

$$\frac{E \xrightarrow{\mu} E'}{E + F \xrightarrow{\mu} E'} \qquad \frac{F \xrightarrow{\mu} F'}{E + F \xrightarrow{\mu} F'}$$

User's Choice

$$coin \rightarrow (choc \rightarrow stop \; + \; water \rightarrow stop)$$

Machine's Choice

$$coin \rightarrow choc \rightarrow stop \quad + \quad coin \rightarrow water \rightarrow stop$$

Choice - 1

Nondeterministic Choice

$$\frac{E \xrightarrow{\mu} E'}{E + F \xrightarrow{\mu} E'} \qquad \frac{F \xrightarrow{\mu} F'}{E + F \xrightarrow{\mu} F'}$$

User's Choice

$$coin \rightarrow (choc \rightarrow stop \; + \; water \rightarrow stop)$$

Machine's Choice

$$coin \rightarrow choc \rightarrow stop \quad + \quad coin \rightarrow water \rightarrow stop$$

Choice - 1

Nondeterministic Choice

$$\frac{E \xrightarrow{\mu} E'}{E + F \xrightarrow{\mu} E'} \qquad \frac{F \xrightarrow{\mu} F'}{E + F \xrightarrow{\mu} F'}$$

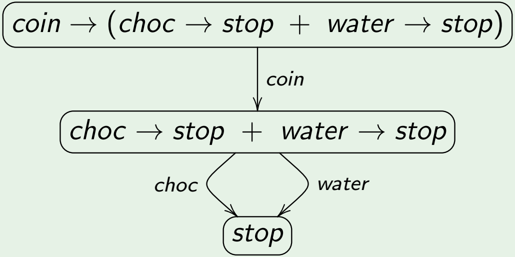
User's Choice

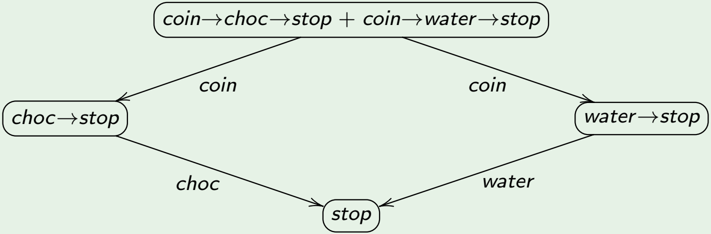
$$coin \rightarrow (choc \rightarrow stop \; + \; water \rightarrow stop)$$

Machine's Choice

$$coin \rightarrow choc \rightarrow stop \quad + \quad coin \rightarrow water \rightarrow stop$$

User's Choice vs Machine's Choice





Choice - 2

Internal Choice

$$\dfrac{}{E \oplus F \xrightarrow{\tau} E} \qquad \dfrac{}{E \oplus F \xrightarrow{\tau} F}$$

Machine's Choice

$$coin \rightarrow (choc \rightarrow stop \;\oplus\; water \rightarrow stop)$$

Choice - 2

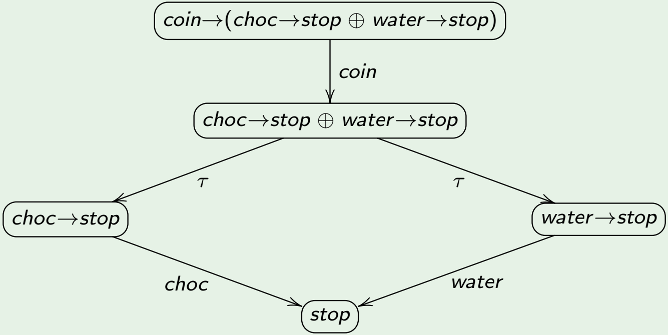
Internal Choice

$$\frac{}{E \oplus F \xrightarrow{\tau} E} \qquad \frac{}{E \oplus F \xrightarrow{\tau} F}$$

Machine's Choice

$$coin \rightarrow (choc \rightarrow stop \;\oplus\; water \rightarrow stop)$$

Machine's Choice, Again



Choice - 3

External Choice

$$\frac{E \xrightarrow{\alpha} E'}{E \,\Box\, F \xrightarrow{\alpha} E'} (\alpha \neq \tau) \qquad \frac{F \xrightarrow{\alpha} F'}{E \,\Box\, F \xrightarrow{\alpha} F'} (\alpha \neq \tau)$$

$$\frac{E \xrightarrow{\tau} E'}{E \,\Box\, F \xrightarrow{\tau} E' \,\Box\, F} \qquad \frac{F \xrightarrow{\tau} F'}{E \,\Box\, F \xrightarrow{\tau} E \,\Box\, F'}$$

Whose Choice?

$$coin \rightarrow ((choc \rightarrow stop \;\oplus\; water \rightarrow stop) \;\Box\; water \rightarrow stop)$$

Choice - 3

External Choice

$$\frac{E \xrightarrow{\alpha} E'}{E \,\Box\, F \xrightarrow{\alpha} E'} (\alpha \neq \tau) \qquad \frac{F \xrightarrow{\alpha} F'}{E \,\Box\, F \xrightarrow{\alpha} F'} (\alpha \neq \tau)$$

$$\frac{E \xrightarrow{\tau} E'}{E \,\Box\, F \xrightarrow{\tau} E' \,\Box\, F} \qquad \frac{F \xrightarrow{\tau} F'}{E \,\Box\, F \xrightarrow{\tau} E \,\Box\, F'}$$

Whose Choice?

$$coin \rightarrow \big((choc \rightarrow stop \ \oplus \ water \rightarrow stop) \ \Box \ water \rightarrow stop\big)$$

Different Transitions

External Choice

$$coin \rightarrow ((choc \rightarrow stop \;\oplus\; water \rightarrow stop) \;\Box\; water \rightarrow stop)$$
$$\xrightarrow{coin}$$
$$(choc \rightarrow stop \;\oplus\; water \rightarrow stop) \;\Box\; water \rightarrow stop$$
$$\xrightarrow{\tau}$$
$$(choc \rightarrow stop \;\Box\; water \rightarrow stop)$$

Internal Choice

$$coin \rightarrow ((choc \rightarrow stop \;\oplus\; water \rightarrow stop) \;\oplus\; water \rightarrow stop)$$
$$\xrightarrow{coin}$$
$$(choc \rightarrow stop \;\oplus\; water \rightarrow stop) \;\oplus\; water \rightarrow stop$$
$$\xrightarrow{\tau}$$
$$choc \rightarrow stop \;\oplus\; water \rightarrow stop$$
$$\xrightarrow{\tau}$$
$$choc \rightarrow stop$$

Different Transitions

External Choice

$$coin \rightarrow ((choc \rightarrow stop \ \oplus \ water \rightarrow stop) \ \Box \ water \rightarrow stop)$$

$$\xrightarrow{coin}$$

$$(choc \rightarrow stop \ \oplus \ water \rightarrow stop) \ \Box \ water \rightarrow stop$$

$$\xrightarrow{\tau}$$

$$(choc \rightarrow stop \ \Box \ water \rightarrow stop)$$

Internal Choice

$$coin \rightarrow ((choc \rightarrow stop \ \oplus \ water \rightarrow stop) \ \oplus \ water \rightarrow stop)$$

$$\xrightarrow{coin}$$

$$(choc \rightarrow stop \ \oplus \ water \rightarrow stop) \ \oplus \ water \rightarrow stop$$

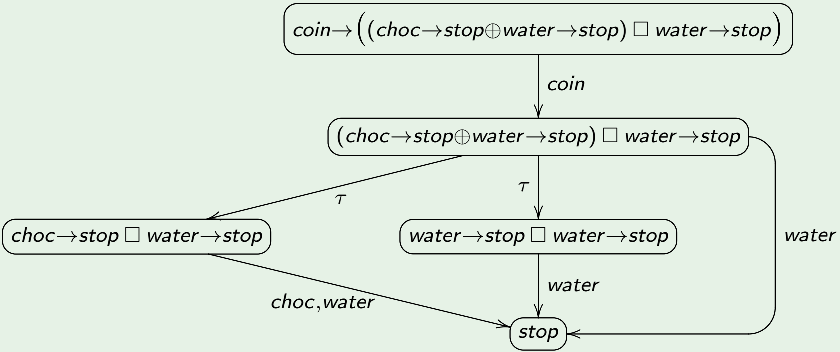
$$\xrightarrow{\tau}$$

$$choc \rightarrow stop \ \oplus \ water \rightarrow stop$$

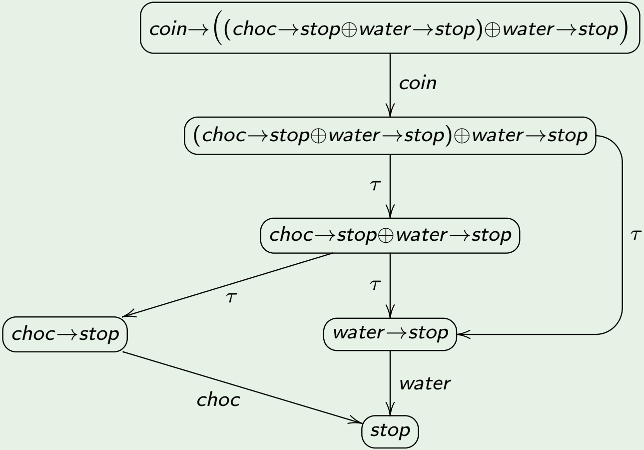
$$\xrightarrow{\tau}$$

$$choc \rightarrow stop$$

User has some choice



User has no choice



Question Time

1. Express $P \oplus Q$ in terms of $\cdot + \cdot$
2. Draw the LTS for $a + b \oplus c \,\Box\, d$ (under all possible ways to parenthesize the expression)

Parallel Composition - 1

Interleaving

$$\frac{E \xrightarrow{\mu} E'}{E \parallel F \xrightarrow{\mu} E' \parallel F} \qquad \frac{F \xrightarrow{\mu} F'}{E \parallel F \xrightarrow{\mu} E \parallel F'}$$

Anagrams as traces

Draw the LTS for

$$a \parallel e \parallel m \parallel r$$

Parallel Composition - 1

Interleaving

$$\frac{E \xrightarrow{\mu} E'}{E \parallel F \xrightarrow{\mu} E' \parallel F} \qquad \frac{F \xrightarrow{\mu} F'}{E \parallel F \xrightarrow{\mu} E \parallel F'}$$

Anagrams as traces

Draw the LTS for

$$a \parallel e \parallel m \parallel r$$

Parallel Composition - 2

We assume the existence of a co-action $\overline{\alpha}$ for each visible α, and let $\overline{\overline{\alpha}} = \alpha$

Milner's Parallel (two-party synchronization)

$$\frac{E \xrightarrow{\mu} E'}{E|F \xrightarrow{\mu} E'|F} \qquad \frac{F \xrightarrow{\mu} F'}{E|F \xrightarrow{\mu} E|F'} \qquad \frac{E \xrightarrow{\alpha} E' \qquad F \xrightarrow{\overline{\alpha}} F'}{E|F \xrightarrow{\tau} E'|F'}(\alpha \neq \tau)$$

User-Machine interaction

$$(coin \rightarrow (\overline{choc} \rightarrow stop \ \oplus \ \overline{water} \rightarrow stop)) \ \ | \ \ (\overline{coin} \rightarrow choc \rightarrow stop)$$

Parallel Composition - 2

We assume the existence of a co-action $\overline{\alpha}$ for each visible α, and let $\overline{\overline{\alpha}} = \alpha$

Milner's Parallel (two-party synchronization)

$$\frac{E \xrightarrow{\mu} E'}{E|F \xrightarrow{\mu} E'|F} \qquad \frac{F \xrightarrow{\mu} F'}{E|F \xrightarrow{\mu} E|F'} \qquad \frac{E \xrightarrow{\alpha} E' \qquad F \xrightarrow{\overline{\alpha}} F'}{E|F \xrightarrow{\tau} E'|F'}(\alpha \neq \tau)$$

User-Machine interaction

$$\big(coin \rightarrow (\overline{choc} \rightarrow stop \;\oplus\; \overline{water} \rightarrow stop)\big) \;\;|\;\; (\overline{coin} \rightarrow choc \rightarrow stop)$$

We can have different interactions

Appropriate Interaction

$$(coin \rightarrow (\overline{choc} \rightarrow stop \;\oplus\; \overline{water} \rightarrow stop)) \;|\; (\overline{coin} \rightarrow choc \rightarrow stop)$$

$$\xrightarrow{\tau}$$

$$(\overline{choc} \rightarrow stop \;\oplus\; \overline{water} \rightarrow stop) \;|\; (choc \rightarrow stop)$$

$$\xrightarrow{\tau}$$

$$(\overline{choc} \rightarrow stop\;) \;|\; (choc \rightarrow stop)$$

$$\xrightarrow{\tau}$$

$$stop \;|\; stop$$

Inappropriate Interaction - Coin thrown away

$$(coin \rightarrow (\overline{choc} \rightarrow stop \;\oplus\; \overline{water} \rightarrow stop)) \;|\; (\overline{coin} \rightarrow choc \rightarrow stop)$$

$$\xrightarrow{\tau}$$

$$(\overline{choc} \rightarrow stop \;\oplus\; \overline{water} \rightarrow stop) \;|\; (choc \rightarrow stop)$$

$$\xrightarrow{\tau}$$

$$(\overline{water} \rightarrow stop) \;|\; (choc \rightarrow stop)$$

We can have different interactions

Appropriate Interaction

$$\big(coin \to (\overline{choc} \to stop \;\oplus\; \overline{water} \to stop)\big) \;\big|\; (\overline{coin} \to choc \to stop)$$

$$\xrightarrow{\tau}$$

$$(\overline{choc} \to stop \;\oplus\; \overline{water} \to stop) \;\big|\; (choc \to stop)$$

$$\xrightarrow{\tau}$$

$$(\overline{choc} \to stop\ ) \;\big|\; (choc \to stop)$$

$$\xrightarrow{\tau}$$

$$stop \;\big|\; stop$$

Inappropriate Interaction - Coin thrown away

$$\big(coin \to (\overline{choc} \to stop \;\oplus\; \overline{water} \to stop)\big) \;\big|\; (\overline{coin} \to choc \to stop)$$

$$\xrightarrow{\tau}$$

$$(\overline{choc} \to stop \;\oplus\; \overline{water} \to stop) \;\big|\; (choc \to stop)$$

$$\xrightarrow{\tau}$$

$$(\overline{water} \to stop) \;\big|\; (choc \to stop)$$

Question Time

Draw the complete LTS for

$$\big(coin \rightarrow (\overline{choc} \rightarrow stop \ \oplus \ \overline{water} \rightarrow stop)\big) \ \mid \ (\overline{coin} \rightarrow choc \rightarrow stop)$$

Parallel Composition - 3

Merge Operator with Synchronization Function

$$\frac{E \xrightarrow{\mu} E'}{E \parallel F \xrightarrow{\mu} E' \parallel F} \qquad \frac{F \xrightarrow{\mu} F'}{E \parallel F \xrightarrow{\mu} E \parallel F'} \qquad \frac{E \xrightarrow{a} E' \quad F \xrightarrow{b} F'}{E \parallel F \xrightarrow{\gamma(a,b)} E' \parallel F'}$$

with $\mu \in \Lambda \cup \{\tau\}$

Another interaction

$$getCoin.(giveChoc.nil \ + \ giveWater.nil) \ \parallel \ putCoin.getChoc.nil$$

with $\gamma(getCoin, putCoin) = ok$ e $\gamma(giveChoc, getChoc) = ok.$

Parallel Composition - 3

Merge Operator with Synchronization Function

$$\frac{E \xrightarrow{\mu} E'}{E \parallel F \xrightarrow{\mu} E' \parallel F} \qquad \frac{F \xrightarrow{\mu} F'}{E \parallel F \xrightarrow{\mu} E \parallel F'} \qquad \frac{E \xrightarrow{a} E' \qquad F \xrightarrow{b} F'}{E \parallel F \xrightarrow{\gamma(a,b)} E' \parallel F'}$$

with $\mu \in \Lambda \cup \{\tau\}$

Another interaction

$$getCoin.(giveChoc.nil \; + \; giveWater.nil) \;\parallel\; putCoin.getChoc.nil$$

with $\gamma(getCoin, putCoin) = ok$ e $\gamma(giveChoc, getChoc) = ok$.

Parallel Composition - 4

Communication Merge

$$\frac{E \xrightarrow{a} E' \qquad F \xrightarrow{b} F'}{E|_c F \xrightarrow{\gamma(a,b)} E' \parallel F'}$$

Left Merge

$$\frac{E \xrightarrow{\mu} E'}{E \mathbin{⫗} F \xrightarrow{\mu} E' \parallel F}$$

Parallel Composition - 4

Communication Merge

$$\frac{E \xrightarrow{a} E' \qquad F \xrightarrow{b} F'}{E|_c F \xrightarrow{\gamma(a,b)} E' \parallel F'}$$

Left Merge

$$\frac{E \xrightarrow{\mu} E'}{E \mathbin{\lfloor\!\!\lfloor} F \xrightarrow{\mu} E' \parallel F}$$

Question Time

Compare the LTS for

$$P \parallel Q$$

with respect to the one for

$$(P \lfloor\!\lfloor Q) + (Q \lfloor\!\lfloor P) + (P|_c Q)$$

Parallel Composition - 5

Hoare's Parallel (multi-party synchronization)

$$\frac{E \xrightarrow{\mu} E'}{E \;|[L]|\; F \xrightarrow{\mu} E' \;|[L]|\; F}\ (\mu \notin L) \qquad \frac{F \xrightarrow{\mu} F'}{E \;|[L]|\; F \xrightarrow{\mu} E \;|[L]|\; F'}\ (\mu \notin L)$$

$$\frac{E \xrightarrow{a} E' \qquad F \xrightarrow{a} F'}{E \;|[L]|\; F \xrightarrow{a} E' \;|[L]|\; F'}\ (a \in L)$$

The operators $\cdot \;|[\emptyset]|\; \cdot$ and $\cdot \;|||\; \cdot$ are substantially equivalent

Parallel Composition - 5

Hoare's Parallel (multi-party synchronization)

$$\frac{E \xrightarrow{\mu} E'}{E \;\|[L]\|\; F \xrightarrow{\mu} E' \;\|[L]\|\; F}\ (\mu \notin L) \qquad \frac{F \xrightarrow{\mu} F'}{E \;\|[L]\|\; F \xrightarrow{\mu} E \;\|[L]\|\; F'}\ (\mu \notin L)$$

$$\frac{E \xrightarrow{a} E' \qquad F \xrightarrow{a} F'}{E \;\|[L]\|\; F \xrightarrow{a} E' \;\|[L]\|\; F'}\ (a \in L)$$

The operators $\cdot \;\|[\emptyset]\|\; \cdot$ and $\cdot \;\|\!|\; \cdot$ are substantially equivalent

Interaction via Synchronization Algebra

Most operators for parallel composition can be expressed in terms of suitable synchronization algebras (assume $E \xrightarrow{*} E$ for all E).

Definition

A Synchronization Algebra is a 4-tuple $\langle \Lambda, *, 0, \bullet \rangle$ where

1. Λ is a set of labels including $*$ (idle) e 0 (forbidden),
2. $\bullet$ is an associative and commutative binary operation Λ (i.e. $\bullet : \Lambda \times \Lambda \to \Lambda$) that satisfies:
   1. $a \bullet 0 = 0$ for all $a \in \Lambda$,
   2. $* \bullet * = *$,
   3. $a \bullet b = *$ implies $a = b = *$, for all $a, b \in \Lambda$.

$$\frac{E \xrightarrow{\alpha} E' \quad F \xrightarrow{\beta} F'}{E \bullet F \xrightarrow{\alpha \bullet \beta} E' \bullet F'} \quad (\alpha \bullet \beta \neq 0)$$

$\bullet$	$*$	α	0
$*$	$*$		0
α			0
0	0	0	0

Interaction via Synchronization Algebra

Most operators for parallel composition can be expressed in terms of suitable synchronization algebras (assume $E \xrightarrow{*} E$ for all E).

Definition

A Synchronization Algebra is a 4-tuple $\langle \Lambda, *, 0, \bullet \rangle$ where

1. Λ is a set of labels including $*$ (idle) e 0 (forbidden),
2. $\bullet$ is an associative and commutative binary operation Λ (i.e. $\bullet : \Lambda \times \Lambda \to \Lambda$) that satisfies:
    1. $a \bullet 0 = 0$ for all $a \in \Lambda$,
    2. $* \bullet * = *$,
    3. $a \bullet b = *$ implies $a = b = *$, for all $a, b \in \Lambda$.

$$\frac{E \xrightarrow{\alpha} E' \quad F \xrightarrow{\beta} F'}{E \bullet F \xrightarrow{\alpha \bullet \beta} E' \bullet F'} \quad (\alpha \bullet \beta \neq 0)$$

$\bullet$	$*$	α	0
$*$	$*$		0
α			0
0	0	0	0

Interaction via Synchronization Algebra

Most operators for parallel composition can be expressed in terms of suitable synchronization algebras (assume $E \xrightarrow{*} E$ for all E).

Definition

A Synchronization Algebra is a 4-tuple $\langle \Lambda, *, 0, \bullet \rangle$ where

1. Λ is a set of labels including $*$ (idle) e 0 (forbidden),
2. $\bullet$ is an associative and commutative binary operation Λ (i.e. $\bullet : \Lambda \times \Lambda \to \Lambda$) that satisfies:
    1. $a \bullet 0 = 0$ for all $a \in \Lambda$,
    2. $* \bullet * = *$,
    3. $a \bullet b = *$ implies $a = b = *$, for all $a, b \in \Lambda$.

$$\frac{E \xrightarrow{\alpha} E' \quad F \xrightarrow{\beta} F'}{E \bullet F \xrightarrow{\alpha \bullet \beta} E' \bullet F'} \quad (\alpha \bullet \beta \neq 0)$$

$\bullet$	$*$	α	0
$*$	$*$		0
α			0
0	0	0	0

Interaction via Synchronization Algebra

Most operators for parallel composition can be expressed in terms of suitable synchronization algebras (assume $E \xrightarrow{*} E$ for all E).

Definition

A Synchronization Algebra is a 4-tuple $\langle \Lambda, *, 0, \bullet \rangle$ where

1. Λ is a set of labels including $*$ (idle) e 0 (forbidden),
2. $\bullet$ is an associative and commutative binary operation Λ (i.e. $\bullet : \Lambda \times \Lambda \to \Lambda$) that satisfies:
    1. $a \bullet 0 = 0$ for all $a \in \Lambda$,
    2. $* \bullet * = *$,
    3. $a \bullet b = *$ implies $a = b = *$, for all $a, b \in \Lambda$.

$$\frac{E \xrightarrow{\alpha} E' \quad F \xrightarrow{\beta} F'}{E \bullet F \xrightarrow{\alpha \bullet \beta} E' \bullet F'} \quad (\alpha \bullet \beta \neq 0)$$

$\bullet$	$*$	α	0
$*$	$*$		0
α			0
0	0	0	0

Abstraction - 1

Restriction

$$\frac{E \xrightarrow{\alpha} E'}{E \setminus L \xrightarrow{\alpha} E' \setminus L} \quad (\alpha\,,\ \overline{\alpha} \notin L)$$

Forcing Interaction

$$(\ (coin.\overline{ok}.nil) \mid ok.(\overline{choc}.nil + \overline{water}.nil)\ ) \setminus ok \quad \mid \quad \overline{coin}.choc.nil$$
$$\xrightarrow{\tau}$$
$$(\ (\overline{ok}.nil) \mid ok.(\overline{choc}.nil + \overline{water}.nil)\ ) \setminus ok \quad \mid \quad choc.nil$$
$$\xrightarrow{\tau}$$
$$(\ nil \mid (\overline{choc}.nil + \overline{water}.nil)\ ) \setminus ok \quad \mid \quad choc.nil$$
$$\xrightarrow{\tau}$$
$$(\ nil \mid nil\ ) \setminus ok \quad \mid \quad nil$$

A malicious user executing $\overline{ok}.choc.nil$ would be stopped.

Abstraction - 1

Restriction

$$\frac{E \xrightarrow{\alpha} E'}{E \setminus L \xrightarrow{\alpha} E' \setminus L} \;\; (\alpha\,,\,\overline{\alpha} \notin L)$$

Forcing Interaction

$$\begin{array}{c}
\big(\, (coin.\overline{ok}.nil) \mid ok.(\overline{choc}.nil + \overline{water}.nil)\,\big)\setminus ok \quad \mid \quad \overline{coin}.choc.nil \\
\xrightarrow{\tau} \\
\big(\, (\overline{ok}.nil) \mid ok.(\overline{choc}.nil + \overline{water}.nil)\,\big)\setminus ok \quad \mid \quad choc.nil \\
\xrightarrow{\tau} \\
\big(\, nil \mid (\overline{choc}.nil + \overline{water}.nil)\,\big)\setminus ok \quad \mid \quad choc.nil \\
\xrightarrow{\tau} \\
\big(\, nil \mid nil\,\big)\setminus ok \quad \mid \quad nil
\end{array}$$

A malicious user executing $\overline{ok}.choc.nil$ would be stopped.

Abstraction - 2

Hiding

$$\frac{E \xrightarrow{\alpha} E'}{E/L \xrightarrow{\alpha} E'/L} \; (\alpha \notin L) \qquad\qquad \frac{E \xrightarrow{\alpha} E'}{E/L \xrightarrow{\tau} E'/L} \; (\alpha \in L)$$

Avoiding Interaction

$$\big(\; (coin.ok.nil) \;\; |[ok]| \;\; ok.(choc.nil + water.nil) \;\big) / \, ok$$

The *ok* signal is internalized thus it cannot be used by a dishonest user.

Abstraction - 3

Renaming

$$\frac{E \xrightarrow{\mu} E'}{E[f] \xrightarrow{f(\mu)} E'[f]}$$

Multilingual Interaction

An Italian user

$$\overline{soldo}.\, acqua.\, nil$$

can interact with the machine with English indication by applying:

$$(\,\overline{soldo}.\, acqua.\, nil\,)\,[coin/soldo,\ water/acqua]$$

Infinite Behaviour - 1

Recursion

$$\frac{E\{rec\,X.E/X\} \xrightarrow{\mu} E'}{rec\,X.E \xrightarrow{\mu} E'}$$

A Long Lasting Vending Machine

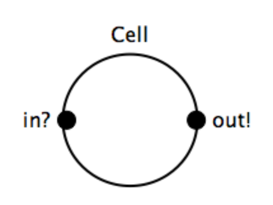
$$rec\,D.\,coin.\,(\overline{choc}.\,D \;+\; \overline{water}.\,D)$$

Infinite Behaviour - 1

Recursion

$$\frac{E\{rec\,X.E/X\} \xrightarrow{\mu} E'}{rec\,X.E \xrightarrow{\mu} E'}$$

Cell $rec\,C.\,in.\overline{out}.\,C$



$$\frac{in.\overline{out}.\,rec\,C.\,in.\overline{out}.\,C \xrightarrow{in} \overline{out}.\,rec\,C.\,in.\overline{out}.\,C}{rec\,C.\,in.\overline{out}.\,C \xrightarrow{in} \overline{out}.\,rec\,C.\,in.\overline{out}.\,C}$$

A Long Lasting Vending Machine

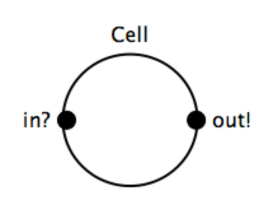
$$rec\,D.\,coin.(\overline{choc}.\,D + \overline{water}.\,D)$$

Infinite Behaviour - 1

Recursion

$$\frac{E\{rec\, X.E/X\} \xrightarrow{\mu} E'}{rec\, X.E \xrightarrow{\mu} E'}$$

Cell $rec\, C.\, in.\overline{out}.\, C$



$$\frac{(in.\overline{out}.\,C)\,\{\; rec\, C.\, in.\overline{out}.\, C \;/\; C\;\}\, in.\overline{out}.\, rec\, C.\, in.\overline{o}}{rec\, C.\, in.\overline{out}.\, C \xrightarrow{in} \overline{out}.\, rec\, C.}$$

A Long Lasting Vending Machine

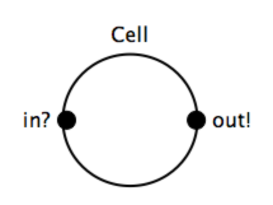
$$rec\, D.\, coin.\,(\overline{choc}.\, D \;+\; \overline{water}.\, D)$$

Infinite Behaviour - 1

Recursion

$$\frac{E\{rec\,X.E/X\} \xrightarrow{\mu} E'}{rec\,X.E \xrightarrow{\mu} E'}$$

Cell $rec\,C.\,in.\overline{out}.\,C$



$$\frac{in.\overline{out}.\,rec\,C.\,in.\overline{out}.\,C \xrightarrow{in} \overline{out}.\,rec\,C.\,in.\overline{out}.\,C}{rec\,C.\,in.\overline{out}.\,C \xrightarrow{in} \overline{out}.\,rec\,C.\,in.\overline{out}.\,C}$$

A Long Lasting Vending Machine

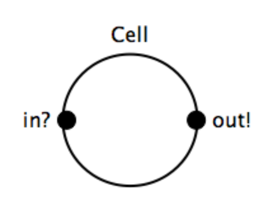
$$rec\,D.\,coin.\,(\overline{choc}.\,D \;+\; \overline{water}.\,D)$$

Infinite Behaviour - 1

Recursion

$$\frac{E\{rec\,X.E/X\} \xrightarrow{\mu} E'}{rec\,X.E \xrightarrow{\mu} E'}$$

Cell $rec\,C.\,in.\overline{out}.\,C$



$$\frac{in.\overline{out}.\,rec\,C.\,in.\overline{out}.\,C \xrightarrow{in} \overline{out}.\,rec\,C.\,in.\overline{out}.\,C}{rec\,C.\,in.\overline{out}.\,C \xrightarrow{in} \overline{out}.\,rec\,C.\,in.\overline{out}.\,C}$$

A Long Lasting Vending Machine

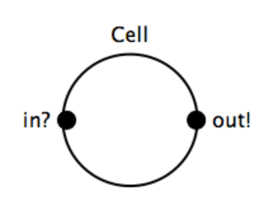
$$rec\,D.\,coin.\,(\overline{choc}.\,D + \overline{water}.\,D)$$

Infinite Behaviour - 1

Recursion

$$\frac{E\{rec\, X.E/X\} \xrightarrow{\mu} E'}{rec\, X.E \xrightarrow{\mu} E'}$$

Cell $rec\, C.\, in.\overline{out}.\, C$



$$\frac{in.\overline{out}.\, rec\, C.\, in.\overline{out}.\, C \xrightarrow{in} \overline{out}.\, rec\, C.\, in.\overline{out}.\, C}{rec\, C.\, in.\overline{out}.\, C \xrightarrow{in} \overline{out}.\, rec\, C.\, in.\overline{out}.\, C}$$

A Long Lasting Vending Machine

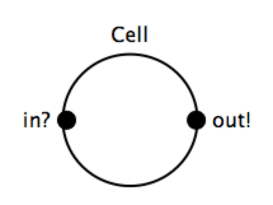
$$rec\, D.\, coin.(\overline{choc}.\, D \;+\; \overline{water}.\, D)$$

Infinite Behaviour - 1

Recursion

$$\frac{E\{rec\,X.E/X\} \xrightarrow{\mu} E'}{rec\,X.E \xrightarrow{\mu} E'}$$

Cell $rec\,C.\,in.\overline{out}.\,C$



$$\frac{in.\overline{out}.\,rec\,C.\,in.\overline{out}.\,C \xrightarrow{in} \overline{out}.\,rec\,C.\,in.\overline{out}.\,C}{rec\,C.\,in.\overline{out}.\,C \xrightarrow{in} \overline{out}.\,rec\,C.\,in.\overline{out}.\,C}$$

A Long Lasting Vending Machine

$$rec\,D.\,coin.\,(\overline{choc}.\,D \;+\; \overline{water}.\,D)$$

Infinite Behaviour - 1

Recursion

$$\frac{E\{rec\, X.E/X\} \xrightarrow{\mu} E'}{rec\, X.E \xrightarrow{\mu} E'}$$

Long Lasting Vending Machine

$$rec\, D.\, coin.\, (\overline{choc}.\, D \;+\; \overline{water}.\, D)$$

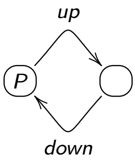
$$rec\, D.\, coin.\, (\overline{choc}.\, D \;+\; \overline{water}.\, D) \quad \Big\} \quad \xrightarrow{coin}$$

$$\left.\begin{array}{l} \overline{choc}.\, rec\, D.\, coin.\, (\overline{choc}.\, D \;+\; \overline{water}.\, D) \\ + \\ \overline{water}.\, rec\, D.\, coin.\, (\overline{choc}.\, D \;+\; \overline{water}.\, D) \end{array}\right\} \quad \xrightarrow{\overline{choc}}$$

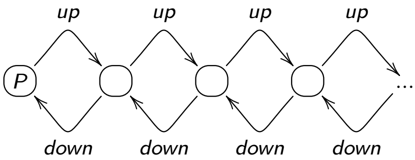
$$rec\, D.\, coin.\, (\overline{choc}.\, D \;+\; \overline{water}.\, D) \quad \Big\} \quad \xrightarrow{coin} \quad \ldots$$

Question Time

1. Define a process P such that its LTS resembles the one below



2. Define a process P such that its LTS resembles the one below



Infinite Behaviour - 2

The notation $rec\,X.E$ for recursion makes the process expressions more difficult to parse and less pleasant to read.

Recursion via constant declaration

A suitable alternative is to allow for the (recursive) definition of some fixed set of constants, that can then be used as some sort of procedure calls inside processes.

Let $\Gamma = \left\{X_1 \triangleq E_1, X_2 \triangleq E_2, \ldots, X_n \triangleq E_n\right\}$ be the set of definitions, then

$$\frac{X \triangleq E \in \Gamma \quad E \xrightarrow{\mu} E'}{X \xrightarrow{\mu} E'}$$

Long Lasting Vending Machine

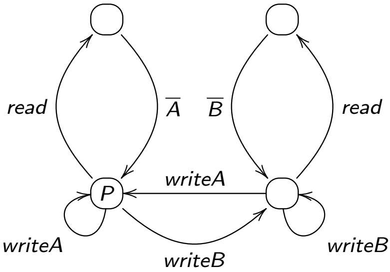
$$D \triangleq coin.\,(\overline{choc}.\,D \ + \ \overline{water}.\,D)$$

From Constant Declarations to Recursive Processes

$X_1 \triangleq E_1$	$rec\, X_1.\, E_1$
$X_1 \triangleq E_1$	$rec\, X_1.\, (E_1\{\ rec\, X_2.E_2\ /\ X_2\ \})$
$X_2 \triangleq E_2$	$rec\, X_2.\, (E_2\{\ rec\, X_1.E_1\ /\ X_1\ \})$
$X_1 \triangleq E_1$	$rec\, X_1.\, ((E_1\{\ rec\, X_2.E_2\ /\ X_2\ \})\{\ rec\, X_3.(E_3\{\ rec\, X_2.E_2\ /\ X_2\ \})\ /\ X_3\ \})$
$X_2 \triangleq E_2$	$rec\, X_2.\, ((E_2\{\ rec\, X_3.E_3\ /\ X_3\ \})\{\ rec\, X_1.(E_1\{\ rec\, X_3.E_3\ /\ X_3\ \})\ /\ X_1\ \})$
$X_3 \triangleq E_3$	$rec\, X_3.\, ((E_3\{\ rec\, X_1.E_1\ /\ X_1\ \})\{\ rec\, X_2.(E_2\{\ rec\, X_1.E_1\ /\ X_1\ \})\ /\ X_2\ \})$
...	...

Question Time

1. Redefine the previous two up-down processes using suitable constant declarations instead of the recursion construct.
2. Define a process P such that its LTS resembles the one below



Infinite Behaviour - 3

Replication

$$\frac{E | \,!\, E \xrightarrow{\mu} E'}{!\, E \xrightarrow{\mu} E'}$$

Chocolate ad libitum

$$\begin{array}{ll}
!\, coin.\, \overline{choc}.\, nil & \xrightarrow{coin} \\
\overline{choc}.\, nil \;\; | \;\; !\, coin.\, \overline{choc}.\, nil & \xrightarrow{coin} \\
\overline{choc}.\, nil \;\; | \;\; \overline{choc}.\, nil \;\; | \;\; !\, coin.\, \overline{choc}.\, nil & \xrightarrow{\overline{choc}} \\
nil \;\; | \;\; \overline{choc}.\, nil \;\; | \;\; !\, coin.\, \overline{choc}.\, nil & \xrightarrow{\overline{choc}} \\
nil \;\; | \;\; nil \;\; | \;\; !\, coin.\, \overline{choc}.\, nil &
\end{array}$$

The replication operator can be defined by the following equation $!\, E \triangleq E|!\, E$ that can be expressed in terms of rec as follows: $recX.(E|X)$

Infinite Behaviour - 4

Iteration

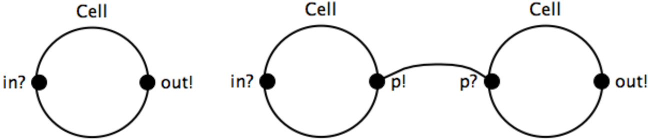
$$\frac{}{E^* \xrightarrow{\epsilon} \surd} \qquad \text{and} \qquad \frac{E \xrightarrow{\mu} E'}{E^* \xrightarrow{\mu} E'; E^*}$$

This iteration operator is the classical one of regular expressions.

Question Time

1. Draw the LTS for $rec\, X.\, (a.X + b.X + c.0)$
2. Draw the LTS for $rec\, X.\, (a.X \oplus b.X \oplus c.0)$
3. Draw the LTS for $rec\, X.\, ((a.X + b.X)|c.0)$
4. Suppose P can issue $\overline{a}$ but not a.
   Intuitively, what is the "meaning" of $(P|a.Q)\backslash a$?
   Intuitively, what is the "meaning" of $(P|!a.Q)\backslash a$?

Linking



$C \triangleq in.\overline{out}.\, C \qquad C_i \triangleq C[p/out] \qquad C_o \triangleq C[p/in]$

$$C \frown C = (\ C[p/out] \mid C[p/in]\ )\backslash p = (\ C_i \mid C_o\ )\backslash p$$

Question Time

Draw the LTS for $C \frown C$

Question Time

Let the operation $\frown$ be defined as

$$X \frown Y = (X[\,p, q, r\,/\,i, d, z\,] | Y[\,p, q, r\,/\,inc, dec, zero]) \backslash \{p, q, r\}$$

Assume the following constants declarations are present:

$$\begin{array}{rcl} Z & \triangleq & \overline{zero}.Z + inc.(I \frown Z) \\ I & \triangleq & \overline{dec}.D + inc.\bar{i}.I \\ D & \triangleq & d.I + z.Z \end{array}$$

Draw, at least in part, the LTS for Z.

Interaction with Value Passing

Single Evolutions

$$\frac{}{a(x).E \xrightarrow{a(v)} E\{v/x\}} \; (v \text{ is a value}) \qquad \frac{}{\overline{a}\, e.E \xrightarrow{\overline{a}\ val(e)} E}$$

Interaction

$$\frac{E \xrightarrow{\overline{a}\ v} E' \quad F \xrightarrow{a(v)} F'}{E|F \xrightarrow{\tau} E'|F'} \qquad \frac{E \xrightarrow{a(v)} E' \quad F \xrightarrow{\overline{a}\ v} F'}{E|F \xrightarrow{\tau} E'|F'}$$

Interaction with Value Passing

Single Evolutions

$$\frac{}{a(x).E \xrightarrow{a(v)} E\{v/x\}} \ (v \ \text{ is a value}) \qquad \frac{}{\overline{a}\, e.E \xrightarrow{\overline{a}\ val(e)} E}$$

Interaction

$$\frac{E \xrightarrow{\overline{a}\ v} E' \quad F \xrightarrow{a(v)} F'}{E|F \xrightarrow{\tau} E'|F'} \qquad \frac{E \xrightarrow{a(v)} E' \quad F \xrightarrow{\overline{a}\ v} F'}{E|F \xrightarrow{\tau} E'|F'}$$

Conditional Execution

$$\frac{val(e) = true \quad E \xrightarrow{\mu} E'}{if\ e\ then\ E\ else\ F \xrightarrow{\mu} E'} \qquad \frac{val(e) = false \quad F \xrightarrow{\mu} F'}{if\ e\ then\ E\ else\ F \xrightarrow{\mu} F'}$$

Let us consider a vending machine that accept 20 cents coins (or higher) and offers a chocolate:

$$coin(x).\ if\ x \geq 20\ then\ choc.nil\ else\ nil$$

The user interacts with the machine as follows:

$$coin(x).\ if\ x \geq 20\ then\ \overline{choc}.nil\ else\ nil \mid \overline{coin}\,40.choc.nil$$
$$\xrightarrow{\tau}$$
$$if\ 40 \geq 20\ then\ \overline{choc}.nil\ else\ nil \mid choc.nil$$
$$\xrightarrow{\tau}$$
$$nil \mid nil$$

Conditional Execution

$$\frac{val(e) = true \quad E \xrightarrow{\mu} E'}{if\ e\ then\ E\ else\ F \xrightarrow{\mu} E'} \qquad \frac{val(e) = false \quad F \xrightarrow{\mu} F'}{if\ e\ then\ E\ else\ F \xrightarrow{\mu} F'}$$

Let us consider a vending machine that accept 20 cents coins (or higher) and offers a chocolate:

$$coin(x).\ if\ x \geq 20\ then\ choc.nil\ else\ nil$$

The user interacts with the machine as follows:

$$coin(x).\ if\ x \geq 20\ then\ \overline{choc}.nil\ else\ nil \ \mid\ \overline{coin}\,40.choc.nil$$
$$\xrightarrow{\tau}$$
$$if\ 40 \geq 20\ then\ \overline{choc}.nil\ else\ nil \ \mid\ choc.nil$$
$$\xrightarrow{\tau}$$
$$nil \ \mid\ nil$$

Pipelining

Pipeline

The binary operator for pipeline is denoted by

- $\cdot > \cdot$

If E ed F are processes, process $E > F$ spawns a copy of F everytime E succeeds.

$$\frac{E \xrightarrow{\mu} E'}{E > F \xrightarrow{\mu} E' > F} \quad (\mu \neq \surd) \qquad\qquad \frac{E \xrightarrow{\surd} E'}{E > F \xrightarrow{\tau} (E' > F)|F}$$

A "three shot" vending machine

$$(coin|coin|coin) > \; choc$$

Pipelining

Pipeline

The binary operator for pipeline is denoted by

- $\cdot > \cdot$

If E ed F are processes, process $E > F$ spawns a copy of F everytime E succeeds.

$$\frac{E \xrightarrow{\mu} E'}{E > F \xrightarrow{\mu} E' > F} \quad (\mu \neq \surd) \qquad\qquad \frac{E \xrightarrow{\surd} E'}{E > F \xrightarrow{\tau} (E' > F)|F}$$

A "three shot" vending machine

$$(coin|coin|coin) > \ choc$$

Interruption - 1

Disabling Operator

The disabling binary operator

- $[>$

permits to interrupt some actions when specific events happen.

$$\frac{E \xrightarrow{\mu} E'}{E \; [> F \xrightarrow{\mu} E' \; [> F} \quad (\mu \neq \surd) \qquad \frac{E \xrightarrow{\surd} E'}{E \; [> F \xrightarrow{\tau} E'} \qquad \frac{F \xrightarrow{\mu} F'}{E \; [> F \xrightarrow{\mu} F'}$$

A cheating customer

$$(coin \rightarrow choc \rightarrow stop) \; [> \; (bang \rightarrow choc \rightarrow stop)$$

This describes a vending machine that when "banged" gives away a chocolate without getting the coin

Interruption - 1

Disabling Operator

The disabling binary operator

- $[>$

permits to interrupt some actions when specific events happen.

$$\frac{E \xrightarrow{\mu} E'}{E \,[>\, F \xrightarrow{\mu} E' \,[>\, F} \; (\mu \neq \surd) \qquad \frac{E \xrightarrow{\surd} E'}{E \,[>\, F \xrightarrow{\tau} E'} \qquad \frac{F \xrightarrow{\mu} F'}{E \,[>\, F \xrightarrow{\mu} F'}$$

A cheating customer

$$(coin \rightarrow choc \rightarrow stop) \;[>\; (bang \rightarrow choc \rightarrow stop)$$

This describes a vending machine that when "banged" gives away a chocolate without getting the coin

Interruption - 2

Try Catch ("non classical")

The try catch operators

- $\texttt{try}\ \_\ \texttt{catch}(A)\ \_$

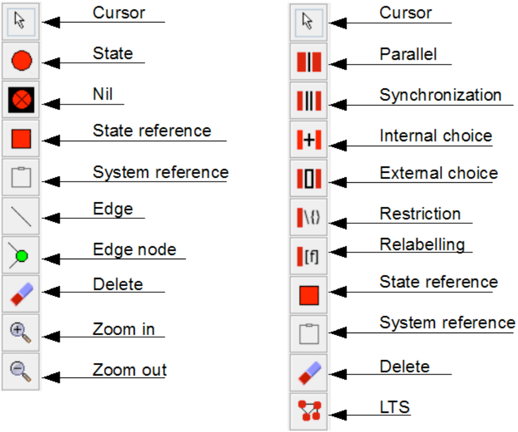
permits to handle specific events (maybe less natural to use in PA).

$$\frac{E \xrightarrow{\mu} E'}{\texttt{try}\ E\ \texttt{catch}(A)\ F \xrightarrow{\mu} \texttt{try}\ E'\ \texttt{catch}(A)\ F} \quad (\mu \notin A)$$

$$\frac{E \xrightarrow{\mu} E'}{\texttt{try}\ E\ \texttt{catch}(A)\ F \xrightarrow{\tau} F} \quad (\mu \in A)$$

$$\frac{E \xrightarrow{\surd} E'}{\texttt{try}\ E\ \texttt{catch}(A)\ F \xrightarrow{\surd} E'}$$

Tapas Basics

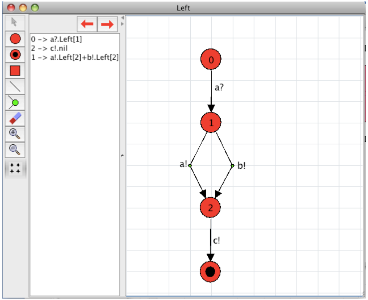


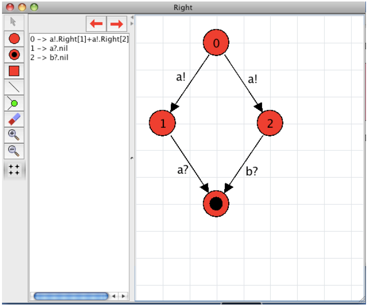
Tapas Syntax

PROC_DEC	::=	$\texttt{X}_1 = \sum_{i\in I_1} \mathsf{ACT}_1^i \,.\, \mathsf{PROC}_1^i$ ... $\texttt{X}_n = \sum_{j\in I_n} \mathsf{ACT}_n^j \,.\, \mathsf{PROC}_n^j$	*(Process dec.)*
ACT	::=	`tau` | `c!` | `c?`	*(Action)*
PROC	::=	`nil` | `P[X]` | `S`	*(Process)*
SYS_DEC	::=	C | C_1 (+) C_2 | C_1 [] C_2 | $C_1 \parallel C_2$	*(System dec.)*
C	::=	PROC	*(Component)*
	|	`sync on` CS `in` $C_1 \parallel C_2$ `end`	
	|	`rename` [F] `in` SYS_DEC `end`	
	|	`restrict` CS `in` SYS_DEC `end`	
CS	::=	$*$ | $\{c_1, \ldots, c_n\}$	*(Channel set)*
F	::=	c/c' | F, F	*(Renaming fun.)*

Tapas Exercise

1. Use Tapas to draw the processes below.

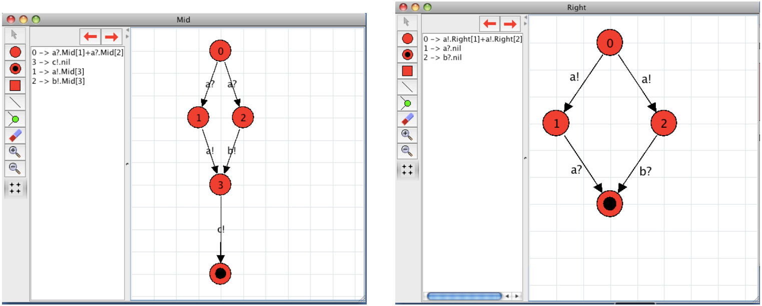




2. Build a system by composing the two processes and restricting ports *a* and *b*.
3. Use Tapas to see if it is guaranteed that *c* is eventually issued.

Tapas Exercise

1. Use Tapas to draw the processes below.



2. Build a system by composing the two processes and restricting ports *a* and *b*.
3. Use Tapas to see if it is guaranteed that *c* is eventually issued.

Practice with Tapas

1. Model a one-position buffer using Tapas: it has input ports *put*, *get*, and output ports *empty*, full, *error*. An error must be issued if an attempt is made to *get* from an empty buffer or to *put* on a full buffer.
2. Model a producer for the above buffer: it has input ports *empty*, *full* and output port *put*. It must check that the buffer is empty before issuing a *put*.
3. Model a consumer for the above buffer: which ports are needed? Which check is recommended?
4. Build a system with a buffer, a producer and a consumer running in parallel, with all ports restricted except *error*. Generate the corresponding LTS and check that error is never issued.
5. Introduce another consumer in the above scenario. Can the system issue some error?

A non-trivial exercise

Define a process P such that its LTS resembles the one below... up-to the execution of some internal actions