Formal Specification of Distributed Systems: Communicating Sequential Processes (CSP)

Nicola Dragoni
Embedded Systems Engineering
DTU Informatics

1 Processes
Forget (for a While) about Distributed Systems...
Forget (for a While) about Distributed Systems...

- Objects **act** and **interact** with us and with each other in accordance with some **characteristic pattern of behaviour**.
Forget (for a While) about Distributed Systems...

- Objects act and interact with us and with each other in accordance with some characteristic pattern of behaviour.

- To describe their patterns of behaviour, first decide what kinds of event or action will be of interest.
Forget (for a While) about Distributed Systems...

- Objects act and interact with us and with each other in accordance with some characteristic pattern of behaviour.

- To describe their patterns of behaviour, first decide what kinds of event or action will be of interest.

- Then, choose a different name for each kind.
Example: Vending Machine

- In the case of a **simple vending machine** there are two kinds of **event**:
  - **coin**: the insertion of a coin in the slot of a vending machine
  - **choc**: the extraction of a chocolate from the dispenser of the machine
In the case of a simple vending machine, there are two kinds of event:

- **coin**: the insertion of a coin in the slot of a vending machine
- **choc**: the extraction of a chocolate from the dispenser of the machine

In the case of a more complex vending machine, there may be a greater variety of events:

- **in1p**: the insertion of one penny
- **in2p**: the insertion of a two penny coin
- **small**: the extraction of a small biscuit or cookie
- **large**: the extraction of a large biscuit or cookie
- **out1p**: the extraction of one penny in change
Alphabet

- The set of names of events which are considered relevant for a particular description of an object is called its alphabet.

- The alphabet is a permanent predefined property of an object: it is logically impossible for an object to engage in an event outside its alphabet.

  - for example, a machine designed to sell chocolates could not suddenly deliver a toy battleship.

- There is no need to make a distinction between events which are initiated by the object (perhaps choc) and those which are initiated by some agent outside the object (for example, coin).
Let us now begin to use the word process to stand for the behaviour pattern of an object, insofar as it can be described in terms of the limited set of events selected as its alphabet.

Conventions:
1. Words in lower-case letters denote distinct events (e.g., coin, choc, in2p, out1p) and so also do the letters, a, b, c, d, e
2. Words in upper-case letters denote specific defined processes (e.g., VMS: the simple vending machine, VMC: the complex vending machine)
3. The letters x, y, z are variables denoting events
4. The letters A, B, C stand for sets of events
5. The letters X, Y are variables denoting processes
6. The alphabet of process P is denoted $\alpha_P$ (e.g., $\alpha_{VMS} = \{\text{coin, choc}\}$, $\alpha_{VMC} = \{\text{in1p, in2p, small, large, out1p}\}$)
Broken Object

• The process with alphabet A which never actually engages in any of the events of A is called STOP\(_A\).

• This describes the behaviour of a broken object: although it is equipped with the physical capabilities to engage in the events of A, it never exercises those capabilities.

• Objects with different alphabets are distinguished, even if they never do anything.
Prefix

• Let $x$ be an event and let $P$ be a process. Then

$$ (x \rightarrow P) $$

(pronounced “$x$ then $P$”)

describes an object which first engages in the event $x$ and then behaves exactly as described by $P$.

• A process description which begins with a prefix is said to be guarded.

• The above process is defined to have the same alphabet as $P$, so this notation must not be used unless $x$ is in that alphabet; more formally:

$$ \alpha(x \rightarrow P) = \alpha P $$

(provided $x \in \alpha P$)
Examples

1. A simple vending machine which consumes one coin before breaking:

\[ \text{(coin} \rightarrow \text{STOP}_{\alpha_{\text{VMS}}}) \]
1. A simple vending machine which consumes one coin before breaking:

\[(\text{coin} \rightarrow \text{STOP}_{\alpha\text{VMS}})\]

2. A simple vending machine that successfully serves two customers before breaking:

\[(\text{coin} \rightarrow (\text{choc} \rightarrow (\text{coin} \rightarrow (\text{choc} \rightarrow \text{STOP}_{\alpha\text{VMS}}))))\]

Initially, this machine will accept insertion of a coin in its slot, but will not allow a chocolate to be extracted.

But after the first coin is inserted, the coin slot closes until a chocolate is extracted.

This machine will not accept two coins in a row, nor will it give out two consecutive chocolates.
Some Remarks

• In future, we shall omit brackets in the case of linear sequences of events, like in the previous example 2., on the convention that $\rightarrow$ is right associative.

• Note that the $\rightarrow$ operator always takes a process on the right and a single event on the left.

• If $P$ and $Q$ are processes, it is SYNTACTICALLY INCORRECT to write $P \rightarrow Q$.

• Similarly, if $x$ and $y$ are events, $x \rightarrow y$ is SYNTACTICALLY INCORRECT.

• Such a process could be correctly described: $x \rightarrow (y \rightarrow \text{STOP})$

• Thus we carefully distinguish the concept of an event from that of a process which engages in events.
Recursion

• The prefix notation can be used to describe the entire behaviour of a process that eventually stops.

• But it would be extremely tedious to write out the full behaviour of a vending machine for its maximum design life; so we need a method of describing repetitive behaviours by much shorter notations.
Recursion

• The prefix notation can be used to describe the entire behaviour of a process that eventually stops.

• But it would be extremely tedious to write out the full behaviour of a vending machine for its maximum design life; so we need a method of describing repetitive behaviours by much shorter notations.

• Consider a clock which never does anything but tick:

\[ \alpha \text{CLOCK} = \{\text{tick}\} \]

• The following equation can be used to specify the behaviour of the clock:

\[ \text{CLOCK} = (\text{tick} \rightarrow \text{CLOCK}) \]
[Recursion] The CLOCK Equation

\[
\text{CLOCK} = (\text{tick} \rightarrow \text{CLOCK})
\]

- The equation for the clock has some obvious consequences, which are derived by simply substituting equals for equals

\[
\begin{align*}
\text{CLOCK} &= (\text{tick} \rightarrow \text{CLOCK}) \\
&= (\text{tick} \rightarrow (\text{tick} \rightarrow \text{CLOCK})) \\
&= (\text{tick} \rightarrow (\text{tick} \rightarrow (\text{tick} \rightarrow \text{CLOCK})))
\end{align*}
\]

- The equation can be unfolded as many times as required, and the possibility of further unfolding will still be preserved. The potentially unbounded behaviour of the CLOCK has been effectively defined as

\[
\text{tick} \rightarrow \text{tick} \rightarrow \text{tick} \rightarrow \text{tick} \rightarrow \ldots
\]
[Recursion] Examples

1. A perpetual clock: \[\text{CLOCK} = \text{tick} \rightarrow \text{CLOCK}\]
[Recursion] Examples

1. A perpetual clock: \[\text{CLOCK} = \text{tick} \rightarrow \text{CLOCK}\]

2. A simple vending machine which serves as many chocs as required:
   \[\text{VMS} = (\text{coin} \rightarrow (\text{choc} \rightarrow \text{VMS}))\]
[Recursion] Examples

1. A perpetual clock: \[ \text{CLOCK} = \text{tick} \rightarrow \text{CLOCK} \]

2. A simple vending machine which serves as many chocs as required:

\[ \text{VMS} = (\text{coin} \rightarrow (\text{choc} \rightarrow \text{VMS})) \]

3. A machine that gives change for 5p repeatedly:

\[ \alpha \text{CH5A} = \{\text{in5p, out2p, out1p}\} \]
\[ \text{CH5A} = (\text{in5p} \rightarrow \text{out2p} \rightarrow \text{out1p} \rightarrow \text{out2p} \rightarrow \text{CH5A}) \]
[Recursion] Examples

1. A perpetual clock: \[
\text{CLOCK} = \text{tick} \rightarrow \text{CLOCK}
\]

2. A simple vending machine which serves as many chocs as required:

\[
\text{VMS} = (\text{coin} \rightarrow (\text{choc} \rightarrow \text{VMS}))
\]

3. A machine that gives change for 5p repeatedly:

\[
\alpha \text{CH5A} = \{\text{in}5\text{p}, \text{out}2\text{p}, \text{out}1\text{p}\}
\]
\[
\text{CH5A} = (\text{in}5\text{p} \rightarrow \text{out}2\text{p} \rightarrow \text{out}1\text{p} \rightarrow \text{out}2\text{p} \rightarrow \text{CH5A})
\]

4. A different change-giving machine with the same alphabet:

\[
\text{CH5B} = (\text{in}5\text{p} \rightarrow \text{out}1\text{p} \rightarrow \text{out}1\text{p} \rightarrow \text{out}1\text{p} \rightarrow \text{out}2\text{p} \rightarrow \text{CH5B})
\]
External Choice

- By means of prefixing and recursion it is possible to describe objects with a single possible stream of behaviour.

- However, many objects allow their behaviour to be influenced by interaction with the environment within which they are placed.

  - For example, a vending machine may offer a choice of slots for inserting a 2p or a 1p coin;

  - it is the customer that decides between these two events.
[External Choice] Bar | Operator

- If \( x \) and \( y \) are distinct events

\[(x \rightarrow P \ | \ y \rightarrow Q)\]

(pronounced “\( x \) then \( P \) choice \( y \) then \( Q \)”)

describes an object which initially engages in either of the events \( x \) or \( y \).

After the first event has occurred, the subsequent behaviour of the object is described by \( P \) if the first event was \( x \), or by \( Q \) if the first event was \( y \).

- Since \( x \) and \( y \) are different events, the choice between \( P \) and \( Q \) is determined by the first event (\( x \) or \( y \)) that actually occurs.
[External Choice] Bar | Operator

• If \( x \) and \( y \) are distinct events

\[
(x \rightarrow P \mid y \rightarrow Q) \quad \text{(pronounced “x then P choice y then Q”)}
\]

describes an object which initially engages in either of the events \( x \) or \( y \).

After the first event has occurred, the subsequent behaviour of the object is described by \( P \) if the first event was \( x \), or by \( Q \) if the first event was \( y \).

• Since \( x \) and \( y \) are different events, the choice between \( P \) and \( Q \) is determined by the first event \( (x \text{ or } y) \) that actually occurs.

• As before, we insist on constancy of alphabets:

\[
\alpha(x \rightarrow P \mid y \rightarrow Q) = \alpha P \quad \text{provided } \{x, y\} \subseteq \alpha P \text{ and } \alpha P = \alpha Q
\]
[External Choice] Examples

1. A machine which offers a choice of two combinations of change for 5p:
[External Choice] Examples

1. A machine which offers a choice of two combinations of change for 5p:

\[
\text{CH5C} = \text{in5p} \rightarrow (\text{out1p} \rightarrow \text{out1p} \rightarrow \text{out1p} \rightarrow \text{out2p} \rightarrow \text{CH5C}) \\
| \text{out2p} \rightarrow \text{out1p} \rightarrow \text{out2p} \rightarrow \text{CH5C})
\]
[External Choice] Examples

1. A machine which offers a choice of two combinations of change for 5p:

$$CH5C = \text{in}5p \rightarrow (\text{out}1p \rightarrow \text{out}1p \rightarrow \text{out}1p \rightarrow \text{out}2p \rightarrow CH5C$$

$$| \text{out}2p \rightarrow \text{out}1p \rightarrow \text{out}2p \rightarrow CH5C)$$

2. A machine that serves either chocolate or toffee on each transaction:

$$VMCT = \text{coin} \rightarrow (\text{choc} \rightarrow VMCT | \text{toffee} \rightarrow VMCT)$$
1. A machine which offers a choice of two combinations of change for 5p:

\[
\text{CH5C} = \text{in5p} \rightarrow (\text{out1p} \rightarrow \text{out1p} \rightarrow \text{out1p} \rightarrow \text{out2p} \rightarrow \text{CH5C} \\
| \text{out2p} \rightarrow \text{out1p} \rightarrow \text{out2p} \rightarrow \text{CH5C})
\]

2. A machine that serves either chocolate or toffee on each transaction:

\[
\text{VMCT} = \text{coin} \rightarrow (\text{choc} \rightarrow \text{VMCT} | \text{toffee} \rightarrow \text{VMCT})
\]

3. A machine which offers a choice of coins and a choice of goods and change:

\[
\text{VMC} = (\text{in2p} \rightarrow (\text{large} \rightarrow \text{VMC} \\
| \text{small} \rightarrow \text{out1p} \rightarrow \text{VMC}) \\
| \text{in1p} \rightarrow (\text{small} \rightarrow \text{VMC} \\
| \text{in1p} \rightarrow (\text{large} \rightarrow \text{VMC} \\
| \text{in1p} \rightarrow \text{STOP})))
\]
1. A machine which offers a choice of two combinations of change for 5p:

\[
\text{CH5C} = \text{in5p} \rightarrow (\text{out1p} \rightarrow \text{out1p} \rightarrow \text{out1p} \rightarrow \text{out2p} \rightarrow \text{CH5C} \\
\text{\hspace{1cm}} | \text{out2p} \rightarrow \text{out1p} \rightarrow \text{out2p} \rightarrow \text{CH5C})
\]

2. A machine that serves either chocolate or toffee on each transaction:

\[
\text{VMCT} = \text{coin} \rightarrow (\text{choc} \rightarrow \text{VMCT} | \text{toffee} \rightarrow \text{VMCT})
\]

3. A machine which offers a choice of coins and a choice of goods and change:

\[
\text{VMC} = (\text{in2p} \rightarrow (\text{large} \rightarrow \text{VMC} \\
\text{\hspace{1cm}} | \text{small} \rightarrow \text{out1p} \rightarrow \text{VMC}) \\
\text{\hspace{1cm}} | \text{in1p} \rightarrow (\text{small} \rightarrow \text{VMC} \\
\text{\hspace{2cm}} | \text{in1p} \rightarrow (\text{large} \rightarrow \text{VMC} \\
\text{\hspace{3cm}} | \text{in1p} \rightarrow \text{STOP})))
\]

“WARNING: do not insert three pennies in a row.”
[External Choice] Extension to More Alternatives

• The definition of choice can be extended to more than two alternatives.

\[(x \rightarrow P \mid y \rightarrow Q \mid \ldots \mid z \rightarrow R)\]

• It is worth noting that \(x, y, z\) must be distinct events \((x \neq y \neq z)\).

• Note that the choice symbol \(\mid\) is not an operator on processes.

• It is SYNTACTICALLY INCORRECT to write \(P \mid Q\), for processes \(P\) and \(Q\).

• Note also that \((x \rightarrow P \mid y \rightarrow Q \mid z \rightarrow R) \neq (x \rightarrow P \mid (y \rightarrow Q \mid z \rightarrow R))\).
[External Choice] Extension to More Alternatives

- The definition of choice can be extended to more than two alternatives.

\[(x \rightarrow P \mid y \rightarrow Q \mid ... \mid z \rightarrow R)\]

- It is worth noting that \(x, y, z\) must be distinct events \((x \neq y \neq z)\).

- Note that the choice symbol \(\mid\) is **not** an operator on processes.

- It is **SYNTACTICALLY INCORRECT** to write \(P \mid Q\), for processes \(P\) and \(Q\).

- Note also that \((x \rightarrow P \mid y \rightarrow Q \mid z \rightarrow R) \neq (x \rightarrow P \mid (y \rightarrow Q \mid z \rightarrow R))\).

**syntactically correct!**
Choice of Events

• In general, if \( B \) is any set of events and \( P(x) \) is an expression defining a process for each different \( x \) in \( B \), then

\[
(x : B \rightarrow P(X)) \quad \text{“x from B then P of x”}
\]

defines a process which first offers a choice of any event \( y \) in \( B \) and then behaves like \( P(y) \).

• In this construction, \( x \) is a local variable, so

\[
(x : B \rightarrow P(X)) = (y : B \rightarrow P(y))
\]

• The set \( B \) defines the initial *menu* of the process, since it gives the set of *actions* between which a choice is to be made at the start.
Example (with Mutual Recursion)

[INFORMAL DESCRIPTION] A drinks dispenser has two buttons labelled \textit{ORANGE} and \textit{LEMON}. The actions of pressing the two buttons are \textit{setorange} and \textit{setlemon}. The actions of dispensing a drink are \textit{orange} and \textit{lemon}. The choice of drink that will next be dispensed is made by pressing the corresponding button. Before any button is pressed, no drink will be dispensed.

[FORMAL SPECIFICATION]

\alpha_{\text{DD}} = \alpha_{\text{O}} = \alpha_{\text{L}} = \{\text{setorange, setlemon, orange, lemon}\}

\text{DD} = (\text{setorange} \rightarrow \text{O} \mid \text{setlemon} \rightarrow \text{L})

\text{O} = (\text{orange} \rightarrow \text{O} \mid \text{setlemon} \rightarrow \text{L} \mid \text{setorange} \rightarrow \text{O})

\text{L} = (\text{lemon} \rightarrow \text{O} \mid \text{setlemon} \rightarrow \text{L} \mid \text{setorange} \rightarrow \text{O})
Pictures (Processes as Trees)

• It may be helpful sometimes to make a pictorial representation of a process as a tree structure, consisting of circles connected by arrows.

• In the traditional terminology of state machines, the circles represent states of the process, and the arrows represent transitions between the states.

• The single circle at the root of the tree is the starting state.

• The process moves downward along the arrows.

• Each arrow is labelled by the event which occurs on making that transition.

• The arrows leading from the same node must all have different labels.
A simple vending machine that successfully serves two customers before breaking:

\[(\text{coin} \rightarrow (\text{choc} \rightarrow (\text{coin} \rightarrow (\text{choc} \rightarrow \text{STOP}_{VMS}))))\]
Example

[Diagram]

- coin
- choc
- toffee

Edges:
- coin to choc
- choc to toffee
- toffee to coin
• A machine that serves either chocolate or toffee on each transaction:

\[ VMCT = \text{coin} \rightarrow (\text{choc} \rightarrow VMCT \mid \text{toffee} \rightarrow VMCT) \]
[Pictures] Question: What is the Difference?
Traces

• A **trace** of the behaviour of a process: *finite sequence of symbols recording the events* in which the process has engaged up to some moment in time.

• Imagine there is an **observer** with a notebook who **watches the process** and **writes down the name of each event as it occurs**.

• A trace can be denoted as a **sequence of symbols**, separated by commas and enclosed in **angular brackets**.

  `<x,y>` consists of two events, `x` followed by `y`

  `<x>` is a sequence containing only the vent `x`

  `<>` is the empty sequence containing no events
1. A simple vending machine which serves as many chocs as required:

\[ VMS = (\text{coin} \rightarrow (\text{choc} \rightarrow VMS)) \]

A trace of this machine at the moment it has completed service of its first two customers:

\[ <\text{coin}, \text{choc}, \text{coin}, \text{choc}> \]
[Traces] Examples

1. A simple vending machine which serves as many chocs as required:

   \[ \text{VMS} = (\text{coin} \rightarrow (\text{choc} \rightarrow \text{VMS})) \]

   A trace of this machine at the moment it has completed service of its first two customers:

   \[ <\text{coin, choc, coin, choc}> \]

2. A trace of the same machine before the second customer has extracted his choc:

   \[ <\text{coin, choc, coin}> \]
[Traces] Examples

1. A simple vending machine which serves as many chocs as required:

\[ VMS = (\text{coin} \rightarrow (\text{choc} \rightarrow VMS)) \]

A trace of this machine at the moment it has completed service of its first two customers:

\[ <\text{coin, choc, coin, choc}> \]

2. A trace of the same machine before the second customer has extracted his choc:

\[ <\text{coin, choc, coin}> \]

3. Before a process has engaged in any events, the notebook of the observer is empty. This is represented by the empty trace:

\[ <> \]
[Traces] Examples

1. A simple vending machine which serves as many chocs as required:

\[ \text{VMS} = (\text{coin} \rightarrow (\text{choc} \rightarrow \text{VMS})) \]

A trace of this machine at the moment it has completed service of its first two customers:

\[ <\text{coin, choc, coin, choc}> \]

2. A trace of the same machine before the second customer has extracted his choc:

\[ <\text{coin, choc, coin}> \]

3. Before a process has engaged in any events, the notebook of the observer is empty. This is represented by the empty trace:

\[ <> \]

- Every process has this as its shortest possible trace.
Operations on Traces

- Traces play a central role in recording, describing, and understanding the behaviour of processes.

- In the following we explore some of the general properties of traces and of operations on them.

- We will use the following conventions
  - $s, t, u$ stand for traces
  - $S, T, U$ stand for sets of traces
  - $f, g, h$ stand for functions
Catenation

• By far the most important operation on traces is catenation, which constructs a trace from a pair of operands \( s \) and \( t \) by simply putting them together in this order; the result will be denoted \( s \circ t \).

\[
\begin{align*}
<\text{coin}, \text{choc}> & \circ <\text{coin}, \text{toffee}> = <\text{coin}, \text{choc}, \text{coin}, \text{toffee}> \\
<\text{in1p}> & \circ <\text{in1p}> = <\text{in1p}, \text{in1p}> \\
<\text{in1p}, \text{in1p}> & \circ <> = <\text{in1p}, \text{in1p}>
\end{align*}
\]
[Catenation] Properties

- The most important properties of catenation are that it is associative, and has <> as its unit:

  L1  \( s^{<>} = <>^s = s \)

  L2  \( s^{(t^u)} = (s^t)^u \)

  L3  \( s^t = s^u \equiv t = u \)

  L4  \( s^t = u^t \equiv s = u \)

  L5  \( s^t = <> \equiv s = <> \land t = <> \)
**Catenation** Properties

- If $n$ is a natural number, we define $t^n$ as $n$ copies of $t$ catenated with each other. It is readily defined by induction on $n$.

| L6  | $t^0 = <>$ |
| L7  | $t^{n+1} = t \cdot t^n$ |
| L8  | $t^{n+1} = t^n \cdot t$ |
| L9  | $(s \cdot t)^{n+1} = s \cdot (t \cdot s)^n \cdot t$ | HOMEWORK |
Restriction

• The expression \((t \uparrow A)\)

  denotes the trace \(t\) when restricted to symbols in the set \(A\); it is formed from \(t\) simply by omitting all symbols outside \(A\).

• Example:

  \(<\text{around,up,down,around}> \uparrow \{\text{up,down}\} = <\text{up,down}>\)
[Restriction] Properties

L1 \[<> \uparrow A = <>\] \hspace{1cm} \textit{strict}

L2 \[(s \bowtie t) \uparrow A = (s \uparrow A) \bowtie (t \uparrow A)\] \hspace{1cm} \textit{distributive}

L3 \[<x> \uparrow A = <x>\] \hspace{1cm} \text{if } x \in A

L4 \[<y> \uparrow A = <>\] \hspace{1cm} \text{if } y \notin A

L5 \[s \uparrow \{\} = <>\]

L6 \[(s \uparrow A) \uparrow B = s \uparrow (A \cap B)\]
Head and Tail

- If $s$ is a nonempty sequence, its first sequence (head) is denoted $s_0$, and the result of removing the first symbol (tail) is $s'$.

- Examples: $<x,y,x>_0 = x$  
  $<x,y,x>' = <y,x>$

- Both operations are undefined for the empty sequence.

L1 $(<x> \widehat{s})_0 = x$

L2 $(<x> \widehat{s})' = s$

L3 $s = (<s_0> \widehat{s'})$ if $s \not\in <>$

L4 $s = t \equiv (s = t = <> \lor (s_0 = t_0 \land s' = t'))$
The set $A^*$ is the set of all finite traces (including <> ) which are formed from symbols in the set $A$.

When such traces are restricted to $A$, they remain unchanged. This fact permits a simple definition:

$$A^* = \{ s \mid s \upharpoonright A = s \}$$

**L1  <> \in A^*  

**L2  <x> \in A^* \equiv x \in A  

**L3  (s \bowtie t) \in A^* \equiv s \in A^* \land t \in A^*
Ordering

• If $s$ is a copy of an initial subsequence of $t$, it is possible to find some extension $u$ of $s$ such that $s \hat{\circ} u = t$.

• Ordering relation ($\leq$) : $s \leq t = (\exists u : s \hat{\circ} u = t)$

• We say that $s$ is a prefix of $t$.

• Examples:

  $<x,y> \leq <x,y,x,w>$

  $<x,y> \leq <z,y,x> \equiv x = z$
[Ordering] Properties

L1 $<> \leq s$ least element

L2 $s \leq s$ reflexive

L3 $s \leq t \land t \leq s \Rightarrow s = t$ antisymmetric

L4 $s \leq t \land t \leq u \Rightarrow s \leq u$ transitive

L5 $(<> s) \leq t \equiv t \neq <> \land x = t_0 \land s \leq t'$

L6 $s \leq u \land t \leq u \Rightarrow s \leq t \lor t \leq s$
[Ordering] Properties

L7  \[ s \text{ in } t = (\exists u, v : u \preceq s \preceq v) \]

L8  \[ (<x> \preceq s) \text{ in } t \equiv t \neq <> \land ((t_0 = x \land s \leq t') \lor ((<x> \preceq s) \text{ in } t')) \]

L9  \[ s \leq t \Rightarrow (s \uparrow A) \leq (t \uparrow A) \]

L10 \[ t \leq u \Rightarrow (s \uparrow t) \leq (s \uparrow u) \]
Length

- The length of the trace $t$ is denoted $\#t$. For instance: $\#<x,y,x> = 3$

| L1 | $\#<> = 0$ |
| L2 | $\#<x> = 1$ |
| L3 | $\#(s \sim t) = (\#s) + (\#t)$ |
| L4 | $\#(t \uparrow (A \cup B)) = \#(t \uparrow A) + \#(t \uparrow B) - \#(t \uparrow (A \cap B))$ |
| L5 | $s \leq t \Rightarrow \#s \leq \#t$ |
| L6 | $\#(t^n) = n \times (\#t)$ |

NB The number of occurrences of a symbol $x$ in a trace $s$ is defined:

$s \downarrow x = \#(s \uparrow \{x\})$
Traces of a Process

• Before a process starts, it is not known which of its possible traces will ACTUALLY be recorded: the choice will depend on environmental factors beyond the control of the process.

• However the COMPLETE set of all possible traces of a process P can be known in advance, and we define a function traces(P) to yield that set.

• Examples:

  \[
  \text{traces(STOP)} = \{<>\}
  \]

  \[
  \text{traces(coin \xrightarrow{} STOP)} = \{<> , <\text{coin}>\}
  \]

  \[
  \text{traces(CLOCK = tick \xrightarrow{} CLOCK)} = \{<> , <\text{tick}> , <\text{tick,tick}> , <\text{tick,tick,tick}> , ...\} = \{\text{tick}\}^*
  \]
Traces of a Process] Some Laws

L1 \( \text{traces}(\text{STOP}) = \{t \mid t = <>\} = \{<>\} \)

L2 \( \text{traces}(c \rightarrow P) = \{<>\} \cup \{<c> \sim t \mid t \in \text{traces}(P)\} \)

L3 \( \text{traces}(c \rightarrow P \mid d \rightarrow Q) = \{<>\} \cup \{<c> \sim t \mid t \in \text{traces}(P)\} \cup \{<d> \sim t \mid t \in \text{traces}(Q)\} \)

L4 \( <> \in \text{traces}(P) \)

L5 \( s \sim t \in \text{traces}(P) \Rightarrow s \in \text{traces}(P) \)

L6 \( \text{traces}(P) \subseteq \alpha(P)^* \)
There is a close relationship between the traces of a process and the picture of its behaviour drawn as a tree:

_for any node on the tree, the trace of the behaviour of a process up to the time when it reaches that node is just the sequence of labels encountered on the path leading from the root of the tree to that node._

Example: in the tree for VMC shown in the Figure, the trace corresponding to the path from the root to the black node is `<in2p,small,out1p>`
Specifications

• A specification of a product is a description of the way it is intended to behave.

• This description is a predicate containing free variables, each of which stands for some observable aspect of the behaviour of the product.

• In the case of a process, the most obviously relevant observation of its behaviour is the trace of events that occur up to a given moment in time.

• We will use the variable $tr$ to stand for an arbitrary trace of the process being specified.
1. The owner of a vending machine does not wish to make a loss by installing it. He therefore specifies that the number of chocolates dispensed must never exceed the number of coins inserted:

\[
\text{NOLOSS} = (\text{tr} \downarrow \text{choc}) \leq (\text{tr} \downarrow \text{coin})
\]
1. The owner of a vending machine does not wish to make a loss by installing it. He therefore specifies that the number of chocolates dispensed must never exceed the number of coins inserted:

\[
\text{NOLOSS} = (\text{tr} \downarrow \text{choc}) \leq (\text{tr} \downarrow \text{coin})
\]

2. The customer of a vending machine wants to ensure that it will not absorb further coins until it has dispensed the chocolate already paid for:

\[
\text{FAIR1} = ((\text{tr} \downarrow \text{coin}) \leq (\text{tr} \downarrow \text{choc}) + 1)
\]
1. The owner of a vending machine does not wish to make a loss by installing it. He therefore specifies that the number of chocolates dispensed must never exceed the number of coins inserted:

\[
\text{NOLOSS} = (\text{tr } \downarrow \text{ choc}) \leq (\text{tr } \downarrow \text{ coin})
\]

2. The customer of a vending machine wants to ensure that it will not absorb further coins until it has dispensed the chocolate already paid for:

\[
\text{FAIR1} = ((\text{tr } \downarrow \text{ coin}) \leq (\text{tr } \downarrow \text{ choc}) + 1)
\]

3. The manufacturer of a simple vending machine must meet the requirements both of its owner and its customer:

\[
\text{VMPEC} = \text{NOLOSS} \land \text{FAIR1} = (0 \leq ((\text{tr } \downarrow \text{ coin}) = (\text{tr } \downarrow \text{ choc}))) \leq 1
\]
4. The specification of a correction to the complex vending machine forbids it to accept three pennies in a row:

\[
\text{VMCFIX} = (\neg <\text{in1p}>^3 \text{ in tr})
\]
4. The specification of a correction to the complex vending machine forbids it to accept three pennies in a row:

\[ \text{VMCFIX} = (\neg \langle \text{in1p} \rangle^3 \text{ in tr}) \]

5. The specification of a mended machine:

\[ \text{MENDVMC} = (\text{tr} \in \text{traces}(\text{VMC}) \land \text{VMCFIX}) \]