Theory of concurrency

Lecture 5

Processes

Processes: **Traces**

- *A trace of the behaviour of a process* is
  - a finite sequence of symbols recording the events in which the process has engaged up to some moment in time.
- We ignore the possibility that two events occur simultaneously
  - record one of them first and then the other (the order does not matter)
- The notation:
  - $<x, y>$ consists of two events, x followed by y.
  - $<x>$ is a sequence containing only the event x.
  - $<>$ is the empty sequence containing no events.

$VMS = (coin \rightarrow (choc \rightarrow VMS))$

Processes: **Traces**

X1. A trace of the simple vending machine *VMS* at the moment it has completed service of its first two customers

<coin, choc, coin, choc>

X2. A trace of the same machine before the second customer has extracted his *choc*

<coin, choc, coin>

- Just observed alphabet events
  - no a completed transaction, no customer hunger, no machine readiness.

X3. Before a process has engaged in any events, the notebook of the observer is empty.

- The empty trace <>
- Every process has this as its shortest possible trace.

$$
\begin{aligned}
VMC = (&in2p \rightarrow (large \rightarrow VMC \mid small \rightarrow out1p \rightarrow VMC) \\
&\mid in1p \rightarrow (small \rightarrow VMC \mid in1p \rightarrow (large \rightarrow VMC \mid in1p \rightarrow STOP)))
\end{aligned}
$$

Processes: **Traces**

X4. The complex vending machine *VMC* has the following seven traces of length 2 or less

$$\langle\rangle$$

$$\langle in2p\rangle \qquad \langle in1p\rangle$$

$$\langle in2p, large\rangle \ \langle in2p, small\rangle \qquad \langle in1p, in1p\rangle \ \langle in1p, small\rangle$$

- Only one of the four traces of length two can actually occur for a given machine.

X5. A trace of *VMC* if its first customer has ignored the warning is

$$\langle in1p, in1p, in1p\rangle$$

- The traces does not record the *breakage*.
  - Breakage: no trace which extends this one, i.e.,
    - there is no event x such that a possible trace of *VMC* is $\langle in1p, in1p, in1p, x\rangle$

Processes: **Operations on traces**

- Traces play a central role in
  - recording, describing, and understanding the behaviour of processes.
- The notation:
  - s, t, u stand for traces
  - S, T, U stand for sets of traces
  - f, g, h stand for functions

Processes: Operations on traces: **Catenation**

- Catenation constructs a trace from a pair *s* and *t* by putting them in the order:
  - $s \wedge t$
  - For example

$<coin, choc>\wedge<coin, toffee> = <coin, choc, coin, toffee>$

$<in1p>\wedge<in1p> = <in1p, in1p>$

$<in1p, in1p>\wedge<> = <in1p, in1p>$

- Unit

L1. $s\wedge<> = <>\wedge s = s$

- Associativity

L2. $s \wedge (t \wedge u) = (s \wedge t) \wedge u$

- Various

L3. $s \wedge t = s \wedge u \ \equiv\ t = u$

L4. $s \wedge t = u \wedge t \ \equiv\ s = u$

L5. $s \wedge t = <> \quad \equiv\ s = <> \land t = <>$

Processes: Operations on traces: **Catenation**

- A function which maps traces to traces is *f* .

- The *strict* function maps the empty trace to the empty trace
  - $f(\langle\rangle) = \langle\rangle$

- The *distributive* function distributes through catenation
  - $f(s \wedge t) = f(s) \wedge f(t)$
  - All distributive functions are strict.
  - A distributive function is uniquely defined by its effect on singleton sequences
    - $f(\langle a\rangle) = \langle b\rangle$
    - its effect on all longer sequences can be calculated by
      - distributing the function to each element and catenating the results.

Processes: Operations on traces: **Catenation**

- $t^n = t \wedge t \wedge \ldots \wedge t \wedge t$ is n copies of t catenated with each other, $n \in \mathbb{N}$.

L6. $t^0 = \langle\rangle$

L7. $t^{n+1} = t \wedge t^n$

- Consequences:

L8. $t^{n+1} = t^n \wedge t$

L9. $(s \wedge t)^{n+1} = s \wedge (t \wedge s)^n \wedge t$

Processes: Operations on traces: **Restriction**

- The trace t is *restricted* to symbols in the set A
  - $(t \upharpoonright A)$
  - omit all symbols outside A.
- For example
  - $\langle around, up, down, around \rangle \upharpoonright \{up, down\} = \langle up, down \rangle$
- Restriction is distributive and therefore strict

L1. $\langle\rangle \upharpoonright A = \langle\rangle$

L2. $(s \,\hat{}\, t) \upharpoonright A = (s \upharpoonright A) \,\hat{}\, (t \upharpoonright A)$

- The effect on singleton sequences

L3. $\langle x \rangle \upharpoonright A = \langle x \rangle$ if $x \in A$

L4. $\langle y \rangle \upharpoonright A = \langle\rangle$ if $y \notin A$

Processes: Operations on traces: **Restriction**

- For example, if $y \neq x$

$$
\begin{aligned}
& \langle x, y, x\rangle \restriction \{x\} \\
&= (\langle x\rangle \wedge \langle y\rangle \wedge \langle x\rangle) \restriction \{x\} \\
&= (\langle x\rangle \restriction \{x\}) \wedge (\langle y\rangle \restriction \{x\}) \wedge (\langle x\rangle \restriction \{x\}) && \text{[by L2]} \\
&= \langle x\rangle \wedge \langle\rangle \wedge \langle x\rangle && \text{[by L3 and L4]} \\
&= \langle x, x\rangle
\end{aligned}
$$

L2. $(s \wedge t) \restriction A = (s \restriction A) \wedge (t \restriction A)$
L3. $\langle x\rangle \restriction A = \langle x\rangle$ if $x \in A$
L4. $\langle y\rangle \restriction A = \langle\rangle$ if $y \notin A$

- Restriction and set operations.

L5. $s \restriction \{\,\} = \langle\rangle$

L6. $(s \restriction A) \restriction B = s \restriction (A \cap B)$

  - These laws can be proved by induction on the length of s

Processes: Operations on traces: **Head and tail**

- For nonempty sequence s
  - s_0 is its first element
  - s' is the result of removing the first element
- For example
  - $\langle x, y, x\rangle_0 = x$
  - $\langle x, y, x\rangle' = \langle y, x\rangle$
- Head and tail are undefined for the empty sequence.

L1. $(\langle x\rangle \wedge s)_0 = x$

L2. $(\langle x\rangle \wedge s)' = s$

L3. $s = (\langle s_0\rangle \wedge s')$ if $s \neq \langle\rangle$

- The law for proving whether two traces are equal

L4. $s = t \equiv (s = t = \langle\rangle \vee (s_0 = t_0 \wedge s' = t'))$.

Processes: Operations on traces: **Star**

- The set A^* is the set of all finite traces (including $\langle\rangle$) which use symbols in the set A.
- Traces from A^* restricted to A remain unchanged.
  - $A^* = \{s \mid s \upharpoonright A = s\}$
- Consequences:

L1. $\langle\rangle \in A^*$

L2. $\langle x\rangle \in A^* \equiv x \in A$

L3. $(s \frown t) \in A^* \equiv s \in A^* \wedge t \in A^*$

- Used to determine whether a trace is a member of A^*.
  - E.g., if $x \in A$ and $y \notin A$

$$
\begin{aligned}
&\langle x, y\rangle \in A^* \\
&\equiv (\langle x\rangle \frown \langle y\rangle) \in A^* \\
&\equiv (\langle x\rangle \in A^*) \wedge (\langle y\rangle \in A^*) && \text{[by L3]} \\
&\equiv true \wedge false && \text{[by L2]} \\
&\equiv false
\end{aligned}
$$

- A recursive definition of A^*:

L4. $A^* = \{t \mid t = \langle\rangle \vee (t_0 \in A \wedge t' \in A^*)\}$

Processes: Operations on traces: **Ordering**

- If s is a copy of an initial subsequence of t
  - it is possible to find extension u of s such that $s \wedge u = t$.
  - s is a *prefix* of t
- An *ordering* relation
  - $s \leq t = (\exists\, u \bullet s \wedge u = t)$

For example,
$\langle x, y\rangle \leq \langle x, y, x, w\rangle$
$\langle x, y\rangle \leq \langle z, y, x\rangle$ iff $x = z$

- The $\leq$ relation is a partial ordering, and its least element is $\langle\rangle$:

L1. $\langle\rangle \leq s$ least element

L2. $s \leq s$ reflexive

L3. $s \leq t \wedge t \leq s \Rightarrow s = t$ antisymmetric

L4. $s \leq t \wedge t \leq u \Rightarrow s \leq u$ transitive

Processes: Operations on traces: **Ordering**

- The law for computing whether $s \leq t$ or not:

L5. $(\langle x \rangle ^\wedge s) \leq t \equiv t \neq \langle\rangle \wedge x = t_0 \wedge s \leq t'$

- The prefixes of a given subsequence are totally ordered:

L6. $s \leq u \wedge t \leq u \Rightarrow s \leq t \vee t \leq s$

- *s is in t* , if s is a subsequence of t:

L7. s in $t = (\exists\, u, v \bullet t = u \,^\wedge s \,^\wedge v)$

- Various:

L8. $(\langle x \rangle \,^\wedge s)$ in $t \equiv t \neq \langle\rangle \wedge ((t_0 = x \wedge s \leq t') \vee (\langle x \rangle \,^\wedge s$ in $t'))$

Processes: Operations on traces: **Ordering**

- A function f from traces to traces is *monotonic* if it respects the ordering $\leq$:
  - $f(s) \leq f(t)$ whenever $s \leq t$
- All distributive functions are monotonic:

L9 $s \leq t \Rightarrow (s \upharpoonright A) \leq (t \upharpoonright A)$

- A *dyadic* function may be
  - monotonic in either argument, keeping the other argument constant.
- Catenation is monotonic in its second argument:

L10 $t \leq u \Rightarrow (s \wedge t) \leq (s \wedge u)$

- A function which is monotonic in all its arguments is simply monotonic.

Processes: Operations on traces: **Length**

- $\#t$ – the length of the trace t.
  - $\#\langle x, y, x\rangle = 3$

L1. $\#\langle\rangle = 0$

L2. $\#\langle x\rangle = 1$

L3. $\#(s \wedge t) = (\#s) + (\#t)$

- $\#(t \upharpoonright A)$ – the number of occurrences in t of symbols from A.

L4. $\#(t \upharpoonright (A \cup B)) = \#(t \upharpoonright A) + \#(t \upharpoonright B) - \#(t \upharpoonright (A \cap B))$

L5. $s \leq t \Rightarrow \#s \leq \#t$

L6. $\#(t^n) = n \times (\#t)$

- The number of occurrences of a symbol x in a trace s:
  - $s \downarrow x = \#(s \upharpoonright \{x\})$

Processes: **Traces of a process**

- A function *traces(P)* describes the complete set of all possible traces of a process *P*.

X1. The only trace of the behaviour of the process *STOP* is <>.

$$traces(STOP) = \{<>\}$$

X2. Two traces of the machine that ingests a coin before breaking

$$traces(coin \rightarrow STOP) = \{<>, <coin>\}$$

X3. A clock that does nothing but *tick*

$$traces(\mu X \bullet tick \rightarrow X) = \{<>, <tick>, <tick, tick>, \ldots\} = \{tick\}^*$$

- The set of traces may be infinite, but each individual trace is finite.

X4. A simple vending machine

$$traces(\mu X \bullet coin \rightarrow choc \rightarrow X) = \{s \mid \exists n \bullet s \leq <coin, choc>^n \}$$

Processes: Traces of a process: **Laws**

- Laws for calculating set of traces.
- *STOP* has only one trace:

L1. $traces(STOP) = \{t \mid t = \langle\rangle\} = \{\langle\rangle\}$

- A trace of $(c \rightarrow P)$ may be empty or not:

L2. $traces(c \rightarrow P) = \{t \mid t = \langle\rangle \vee (t_0 = c \wedge t' \in traces(P))\} =$

$$\{\langle\rangle\} \cup \{\langle c\rangle \wedge t \mid t \in traces(P)\}$$

- A trace of a choice between is a trace of one of the alternatives:

L3. $traces(c \rightarrow P \mid d \rightarrow Q) = \{t \mid t = \langle\rangle \vee (t_0 = c \wedge t' \in traces(P)) \vee (t_0 = d \wedge t' \in traces(Q))\}$

- The single general law governing choice

L4. $traces(x : B \rightarrow P(x)) = \{t \mid t = \langle\rangle \vee (t_0 \in B \wedge t' \in traces(P(t_0)))\}$

Processes: Traces of a process: **Laws**

- The set of traces of a recursively defined process.
- A recursively defined process is the solution of an equation $X = F(X)$
- Iteration of the function F by induction

$$F^0(X) = X$$

$$\begin{aligned} F^{n+1}(X) &= F(F^n(X)) \\ &= F^n(F(X)) \\ &= \underbrace{F(\ldots\ (F(F(}_{n\ times}X)))\ldots) \end{aligned}$$

- If F is guarded, then:

L5. $traces(\mu X : A \bullet F(X)) = \bigcup_{n \geq 0} traces(F^n(STOP_A))$

Processes: Traces of a process: **Laws**

X1. RUN_A is defined as $RUN_A = (x : A \to RUN_A)$

$$\mu X : A \bullet F(X),\ F(X) = (x : A \to X)$$

L4. $A^* = \{t \mid t = \langle\rangle \vee (t_0 \in A \wedge t' \in A^*)\}$
L4. $traces(x : B \to P(x)) =$
$\{t \mid t = \langle\rangle \vee (t_0 \in B \wedge t' \in traces(P(t_0)))\}$

- Prove that $traces(RUN_A) = A^*$
- *Proof* : induction on n. $A^* = \bigcup_{n \geq 0} \{ s \mid s \in A^* \wedge \#s \leq n \}$

1. $traces(STOP_A) = \{\langle\rangle\} = \{s \mid s \in A^* \wedge \#s \leq 0\}$

2. $traces(F^{n+1}(STOP_A))$

$= traces(x : A \to F^n(STOP_A))$	[def. F, F^{n+1}]
$= \{t \mid t = \langle\rangle \vee (t_0 \in A \wedge t' \in traces(F^n(STOP_A)))\}$	[L4]
$= \{t \mid t = \langle\rangle \vee (t_0 \in A \wedge (t' \in A^* \wedge \#t' \leq n))\}$	[ind. hyp.]
$= \{t \mid (t = \langle\rangle \vee (t_0 \in A \wedge t' \in A^*)) \wedge \#t' \leq n + 1\}$	[property of #]
$= \{t \mid t \in A^* \wedge \#t \leq n + 1\}$	[T* L4]

Processes: Traces of a process: **Laws**

X2. *VMS* is defined as $VMS = (coin \rightarrow (choc \rightarrow VMS))$

- Prove that $traces(VMS) = \bigcup_{n\geq 0}\{ s \mid s \leq \langle coin, choc\rangle^n \}$
- *Proof* : induction on *n*.

$traces(F^n(VMS)) = \{t \mid t \leq \langle coin, choc\rangle^n \}$ with $F(X) = coin \rightarrow choc \rightarrow X$

1. $traces(STOP) = \{\langle\rangle\} = \{s \mid s \leq \langle coin, choc\rangle^0 \}$ [C L6]

2. $traces(coin \rightarrow choc \rightarrow F^n(STOP))$

$= \{\langle\rangle, \langle coin\rangle\} \cup \{ \langle coin, choc\rangle \frown t \mid t \in traces(F^n(STOP)) \}$ [L2 twice]

$= \{\langle\rangle, \langle coin\rangle\} \cup \{ \langle coin, choc\rangle \frown t \mid t \leq \langle coin, choc\rangle^n \}$ [ind. hyp.]

$= \{s \mid s = \langle\rangle \vee s = \langle coin\rangle \vee \exists\, t \bullet s = \langle coin, choc\rangle \frown t \wedge t \leq \langle coin, choc\rangle^n \}$

$= \{s \mid s \leq \langle coin, choc\rangle^{n+1} \}$

- The conclusion follows by L5.

L6. $t^0 = \langle\rangle$

L2. $traces(c \rightarrow P) = \{t \mid t = \langle\rangle \vee (t_0 = c \wedge t' \in traces(P)) \} = \{\langle\rangle\} \cup \{ \langle c\rangle \frown t \mid t \in traces(P) \}$

L5. $traces(\mu X : A \bullet F(X)) = \bigcup_{n\geq 0} traces(F^n(STOP_A))$

Processes: Traces of a process: **Laws**

- <> is a trace:

L6. $<> \in traces(P)$

- The prefix is a trace:

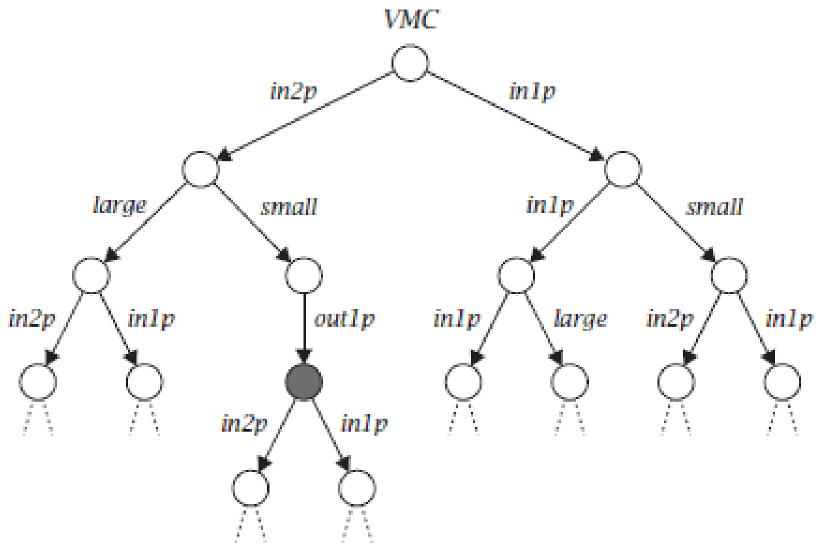
L7. $s\char`^t \in traces(P) \Rightarrow s \in traces(P)$

- Every occurring event is in the alphabet:

L8. $traces(P) \subseteq (aP)^*$

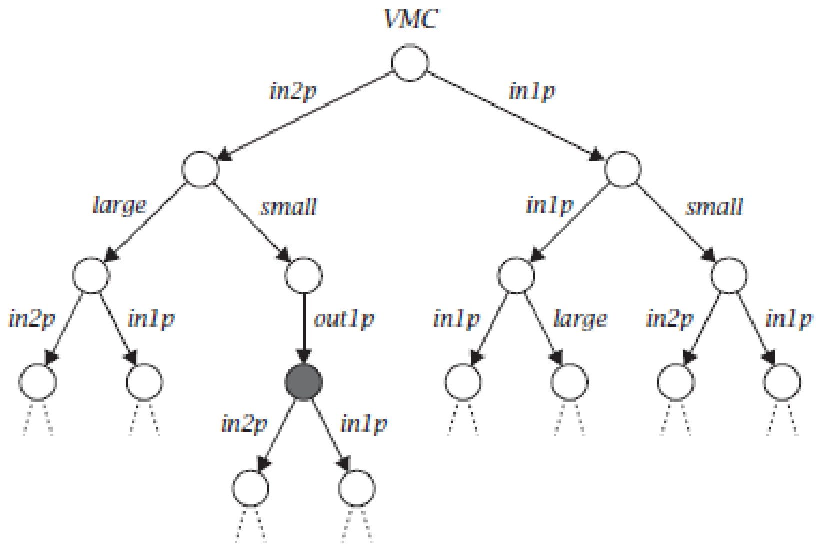
Processes: Traces of a process: **Laws**

- The trace of the behaviour of a process up to any node on the tree is
  - the sequence of labels on the path from the root of the tree to that node.
- In the tree for *VMC*
  - the trace for the path from the root to the black node is <*in2p, small, out1p*>



Processes: Traces of a process: **Laws**

- *All initial subpaths* of a path in a tree are also *paths* in the same tree.
- *The empty trace* defines the path from the root to itself.
- Any set of traces satisfying L6 and L7 constitutes a mathematical representation for
  - a tree with no duplicate labels on branches emerging from a single node.
  - *The traces* of a process are *the set of paths* from the root to some node in the tree.
  - Conversely, *each path* from the root of a tree to a node uniquely specifies *a trace*.



L6. $\langle\rangle \in traces(P)$
L7. $s\hat{}t \in traces(P) \Rightarrow s \in traces(P)$

Processes: Traces of a process: **After**

- A process which behaves the same as P after all the actions in the trace s ($s \in traces(P)$)
  - P / s (P after s)
- If s is not a trace of P, $(P\ /\ s)$ is not defined.

X1. $(VMS\ /\ \langle coin \rangle) = (choc \rightarrow VMS)$

$VMS = (coin \rightarrow (choc \rightarrow VMS))$

X2. $(VMS\ /\ \langle coin, choc \rangle) = VMS$

X3. $(VMC\ /\ \langle in1p \rangle^3) = STOP$

X4. To avoid loss arising from installation of *VMCRED*, the owner decides to eat the first chocolate himself

$VMCRED = \mu X \bullet (coin \rightarrow choc \rightarrow X \mid choc \rightarrow coin \rightarrow X)$

$VMS2 = (coin \rightarrow VMCRED)$

- $(VMCRED\ /\ \langle choc \rangle) = VMS2$.
- In a tree of P, $(P\ /\ s)$ is the *subtree* with the root at the end of the path labelled by the symbols of s.

Processes: Traces of a process: **After**

- The laws for the meaning of the operator $/$.
- After doing nothing, a process remains unchanged:

L1. $P\ /\ \langle\rangle = P$

- After engaging in s^t, the behaviour of P is the same as that of $(P\ /\ s)$ after engaging in t:

L2. $P\ /\ (s\hat{}t) = (P\ /\ s)\ /\ t$

- After engaging in event c, the behaviour of a process is as defined by this initial choice:

L3. $(x : B \to P(x))\ /\ \langle c\rangle = P(c)$ **if** $c \in B$

- $/\langle c\rangle$ is the inverse of the prefixing operator $c \to$:

L3A. $(c \to P)\ /\ \langle c\rangle = P$

Processes: Traces of a process: **After**

- The traces of $(P \ / \ s)$:

L4. $traces(P \ / \ s) = \{t \mid s \frown t \in traces(P)\}$ **if** $s \in traces(P)$

- A process P never stops iff
  - $\forall s : traces(P) \bullet P \ / \ s \neq STOP$
- A process P is cyclic if in all circumstances it is possible for it to return to its initial state:
  - $\forall s : traces(P) \bullet \exists t \bullet (P \ / \ (s \frown t) = P)$

- $STOP$ is trivially cyclic.
- If a process is cyclic, then it also never stops.
  - What about inverse?

Processes: Traces of a process: **After**

X5. Are the following processes cyclic?

- RUN_A
- VMS

$$RUN_A = (x : A \to RUN_A)$$

$$VMS = (coin \to (choc \to VMS))$$

- $(choc \to VMS)$
- $(coin \to VMS)$

Processes: Traces of a process: **After**

- The use of / in a recursively defined process may *invalidate* its guards
  - multiple solutions for the recursive equations.

- For example
  - $X = (a \rightarrow (X \,/\, \langle a \rangle))$
    - is not guarded
    - has as its solution *any* process of the form $a \rightarrow P$ for any P.
      - *Proof* : $(a \rightarrow ((a \rightarrow P) \,/\, \langle a \rangle)) = (a \rightarrow P)$ by L3A.

- / operator is never used the in recursive process definitions.

L3A. $(c \rightarrow P) \,/\, \langle c \rangle = P$

Processes: More operations on traces: **Change of symbol**

- f^* maps a sequence of symbols in A^* to a sequence in B^* by
  - applying f to each element of the sequence,
    - f is a function mapping symbols from a set A to symbols in a set B.

- *double* is a function which doubles its integer argument
  - $double^* (\langle 1, 5, 3, 1\rangle) = \langle 2, 10, 6, 2\rangle$

- A starred function is distributive and strict:

L1. $f^*(\langle\rangle) = \langle\rangle$

L2. $f^*(\langle x\rangle) = f(\langle x\rangle)$

L3. $f^*(s \^{} t) = f^*(s) \^{} f^*(t)$

Processes: More operations on traces: **Change of symbol**

L1. $f^*(<>) = <>$
L2. $f^*(<x>) = f(<x>)$
L3. $f^*(s \wedge t) = f^*(s) \wedge f^*(t)$

- Consequences

L4. $f^*(s)_0 = f(s_0)$ if $s \neq <>$

L5. $\# f^*(s) = \# s$

- Not a law in general: $f^*(s \upharpoonright A) = f^*(s) \upharpoonright f(A)$, where $f(A) = \{f(x) \mid x \in A\}$.

- The counterexample:

$f(b) = f(c) = c$ where $b \neq c$

$$
\begin{aligned}
& f^*(<b> \upharpoonright \{c\}) \\
&= f^*(<>) && \text{[since } b \neq c\text{]} \\
&= <> && \text{[L1]} \\
&\neq <c> \\
&= <c> \upharpoonright \{c\} \\
&= f^*(<c>) \upharpoonright f(\{<c>\}) && \text{[since } f(c) = c\text{]}
\end{aligned}
$$

- This law is true if f is a one-one function (injection)

L6. $f^*(s \upharpoonright A) = f^*(s) \upharpoonright f(A)$ **if** f is an injection.

Processes: More operations on traces: **Interleaving**

- A sequence s is an *interleaving* of two sequences t and u if
  - it can be split into a series of subsequences
    - with alternate subsequences extracted from t and u.

- s is an interleaving of t and u:

$s = \langle 1, 6, 3, 1, 5, 4, 2, 7 \rangle$ $t = \langle 1, 6, 5, 2, 7 \rangle$ and $u = \langle 3, 1, 4 \rangle$

- A recursive definition of interleaving:

L1. $\langle\rangle$ *interleaves* $(t, u) \equiv (t = \langle\rangle \wedge u = \langle\rangle)$

L2. s *interleaves* $(t, u) \equiv s$ *interleaves* (u, t)

L3. $(\langle x \rangle ^\frown s)$ *interleaves* $(t, u) \equiv$

$$(t \neq \langle\rangle \wedge t_0 = x \wedge s \text{ interleaves } (t', u)) \vee (u \neq \langle\rangle \wedge u_0 = x \wedge s \text{ interleaves } (t, u'))$$

Processes: More operations on traces: **Composition**

- ✓ is a symbol for successful termination of the process.
  - can appear only at the end of a trace.
- *(s ; t)* is the composition of *s* and *t*.
  - *t* is a trace which start when *s* has successfully terminated.
- If ✓ does not occur in *s*, then *t* cannot start:

L1. $s ; t = s$ **if** $\neg(<✓> \text{ in } s)$

- If ✓ does occur at the end of *s*, it is removed and *t* is appended to the result:

L2. $(s \wedge <✓>) ; t = s \wedge t$ **if** $\neg(<✓> \text{ in } s)$

- All symbols after the first occurrence of ✓ are irrelevant and should be discarded:

L2A. $(s \wedge <✓> \wedge <u>) ; t = s \wedge t$ **if** $\neg(<✓> \text{ in } s)$

Processes: More operations on traces: **Composition**

- Associativity:

L3. $s\ ;\ (t\ ;\ u) = (s\ ;\ t)\ ;\ u$

- Monotonicity in first and second argument:

L4A. $s \le t \Rightarrow ((u\ ;\ s) \le (u\ ;\ t))$

L4B. $s \le t \Rightarrow ((s\ ;\ u) \le (t\ ;\ u))$

- Strictness in first argument:

L5. $\boldsymbol{\langle\rangle}\ ;\ t = \langle\rangle$

- ✓ is the left unit:

L6. $\langle ✓ \rangle\ ;\ t = t$

- ✓ is the right unit, if ✓ never occurs except at the end of a trace:

L7. $s\ ;\ \langle ✓ \rangle = s$ **if** $\neg(\langle ✓ \rangle \text{ in } (s))$

Processes: More operations on traces: **Subscription**

- $s[i]$ denotes the i^{th} element of the sequence s $(0 \leq i \leq \#s)$:

L1. $s[0] = s_0 \wedge s[i + 1] = s'[i]$ **if** $s \neq \langle\rangle$

L2. $(f^*(s))[i] = f(s[i])$ **for** $i < \#s$

Processes: More operations on traces: **Reversal**

- $\underline{s}$ is formed by taking elements of sequence s in reverse order.
  - $\underline{\langle 3, 5, 37 \rangle} = \langle 37, 5, 3 \rangle$
- Some laws for reversal:

L1. $\underline{\langle\rangle} = \langle\rangle$

L2. $\underline{\langle x \rangle} = \langle x \rangle$

L3. $\underline{s \hat{} t} = \underline{t} \hat{} \underline{s}$

L4. $\underline{\underline{s}} = s$

L5. $\underline{s}[i] = s[\#s - i - 1]$ **for** $i \le \#s$

Processes: More operations on traces: **Selection**

- If s is not a sequence of pairs, $s{\downarrow}a$ denotes the number of occurrences of a in s.

- For sequence of pairs s, $s{\downarrow}x$ is the result of
  - selecting from s all pairs with first element x and
  - replacing each pair by its second element.
  - $s = \langle a.7, b.9, a.8, c.0 \rangle$
    - $s{\downarrow}a = \langle 7, 8 \rangle$ and $s{\downarrow}d = \langle \rangle$

L1. $\langle\rangle{\downarrow}x = \langle\rangle$

L2. $(\langle y.z \rangle \wedge t) \downarrow x = t{\downarrow}x$ if $y \neq x$

L3. $(\langle x.z \rangle \wedge t) \downarrow x = \langle z \rangle \wedge (t{\downarrow}x)$