The language

$STOP$	Process that does nothing.
$a \to P$	Prefixing a process by the event $a \in \Sigma$
$?x : A \to P$	Prefix choice, or 'input' (also forms based on multi-part channels which we will not generally deal with in the semantics, as they can be re-written).
$P \mathrel{\Box} Q$	External choice based on initial events.
$P \sqcap Q$	Binary nondeterministic choice.
$\bigsqcap S$	Nondeterministic choice over a nonempty set S of processes.
$P \rhd Q$	Sliding choice (equivalent to $(P \sqcap STOP) \mathrel{\Box} Q$)
$P \underset{X}{\parallel} Q$	Parallel composition interfaced on X
$P \setminus X$	Hides $X \subseteq \Sigma$
$P[\![R]\!]$	Applies the renaming relation R
$SKIP$	Terminating process (signalled by ✓)
$P;\ Q$	Sequential composition.

Plus recursion: single, mutual, infinite mutual.

Process terms

This language contains the infinitary constructs:

$\bigsqcap S$ (S infinite)

$?x : A \to P$ (A infinite, and P varying with x)

infinite mutual recursion

and therefore has infinite process terms in it.

The set of process terms is the smallest fixed point of the syntax: the *well-founded* process terms. Thus we can do structural induction.

In order for this fixed point to exist we need a pre-defined cardinal bound on the size of S and set of allowable mutual recursion structures.

Semantic approaches to CSP

Operational semantics: a formalised implementation strategy for a language. We will interpret closed program terms as a *labelled transition system*.

A *denotational semantics* maps a language into some abstract model which attempts to capture the "essence" of each program. The denotational models of CSP are based on well-chosen sets of behaviours (e.g., traces, failures/divergences).

An *algebraic semantics* seeks to characterise equivalence between processes by setting out a complete set of laws and associated rules.

While any one of these flavours of semantics is enough to describe CSP, it is far preferable to understand something of them all and how they complement each other. We will give formal expositions of the three semantic styles, and show how they relate to each other.

Concentration on essentials

While machine-readable CSP contains a rich functional language for creating the data items used for process parameters and state, we will ignore the semantic details of this in order to concentrate on the communicating behaviour.

We will always simply assume that all these values have been computed. Provided (as we do) we assume that all communicable types are finitary and processes are never involved in parameters or communications, this create few difficulties.

Transition systems

A *labelled transition system* (LTS) is a set of nodes and, for each event a in some set, a relation $\xrightarrow{a}$ between nodes.

Think of it as a directed graph with a label on each edge.

Most LTS's have a distinguished node n_0 which is the one we are assumed to start from.

You move from node to node, with the labels on the edges being the actions seen when each step is made. The set of outward-pointing labels from a node P is written P^0.

The labels are the only things that can be observed during a run.

Actions

For CSP we usually take the set of possible labels to be $\Sigma^{\checkmark,\tau} = \Sigma \cup \{\checkmark, \tau\}$.

Actions in Σ are visible to the external environment, and can only happen with its co-operation.

The special action τ cannot be seen from the outside and happens automatically.

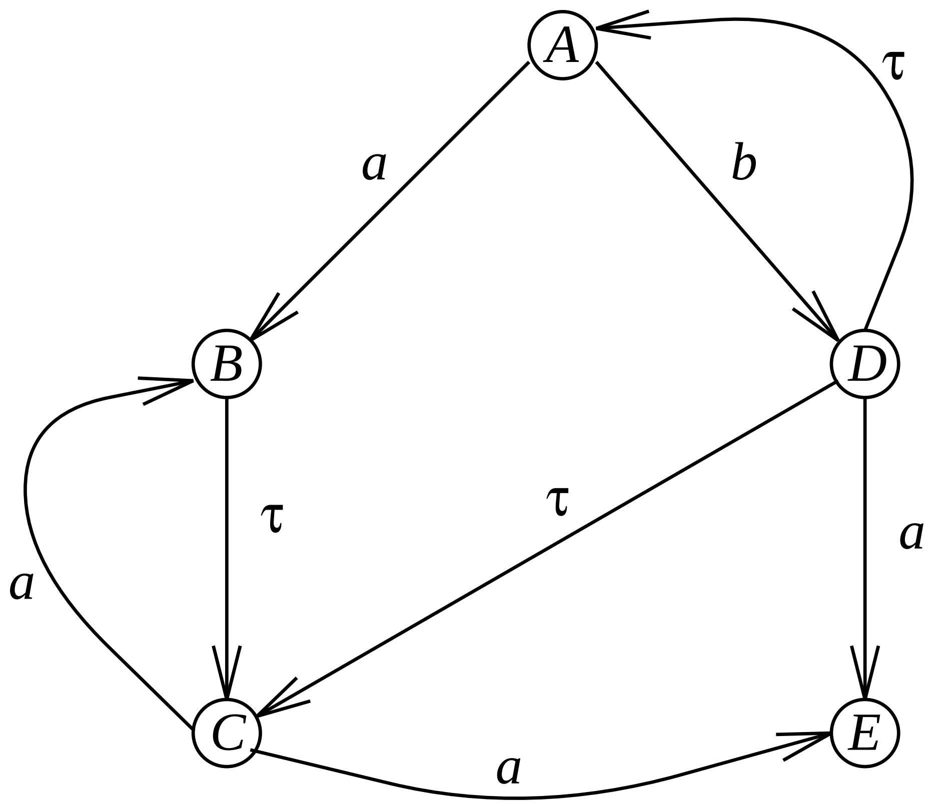
If in a state with no actions outside Σ (a *stable* state) we might have to wait there for ever.

When in an *unstable* state ($\tau \in P^0$) some action must occur within a short time. (The event that occurs may be visible or τ.)

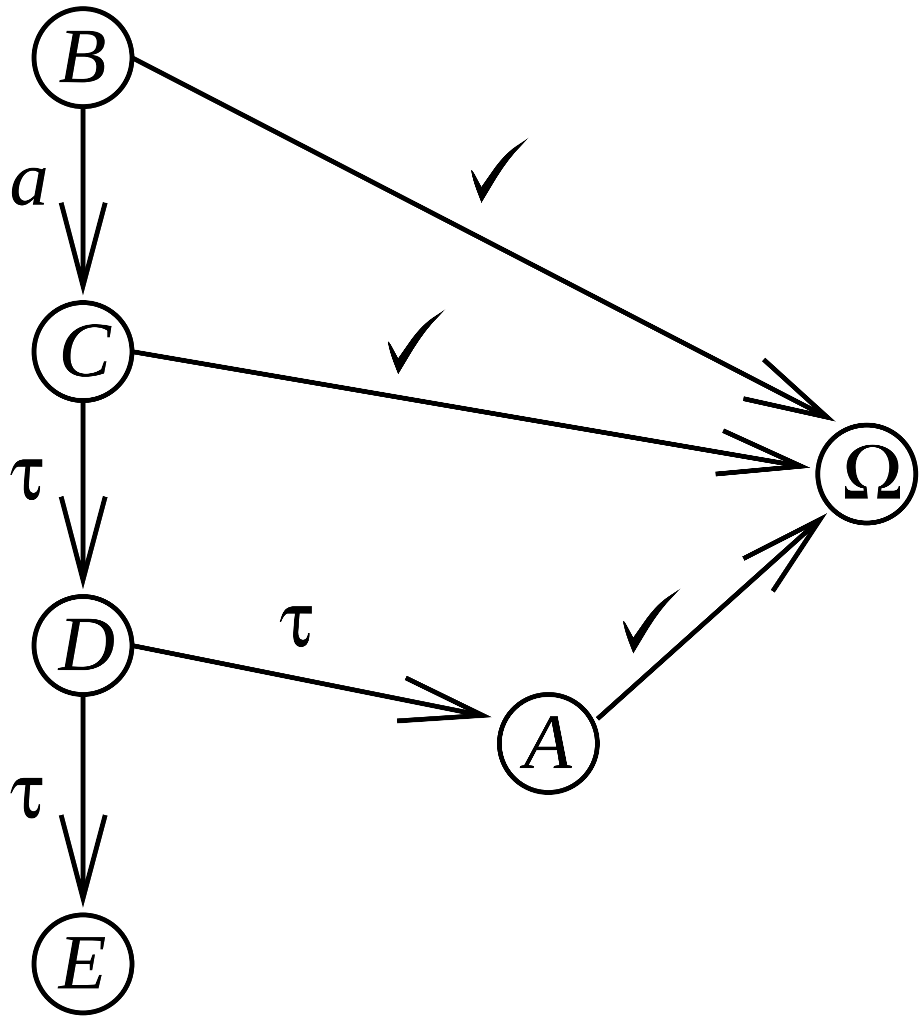
$\checkmark$ is a special signal representing the successful termination. It is observable by the environment but not controllable by it. It is also, conventionally, the final event (so the following state should not have any).

Nondeterminism arises through the occurrence of τ actions, through ambiguous branching on Σ actions, and from any state where $\checkmark$ and Σ actions are both possible.

Example LTS without ✓

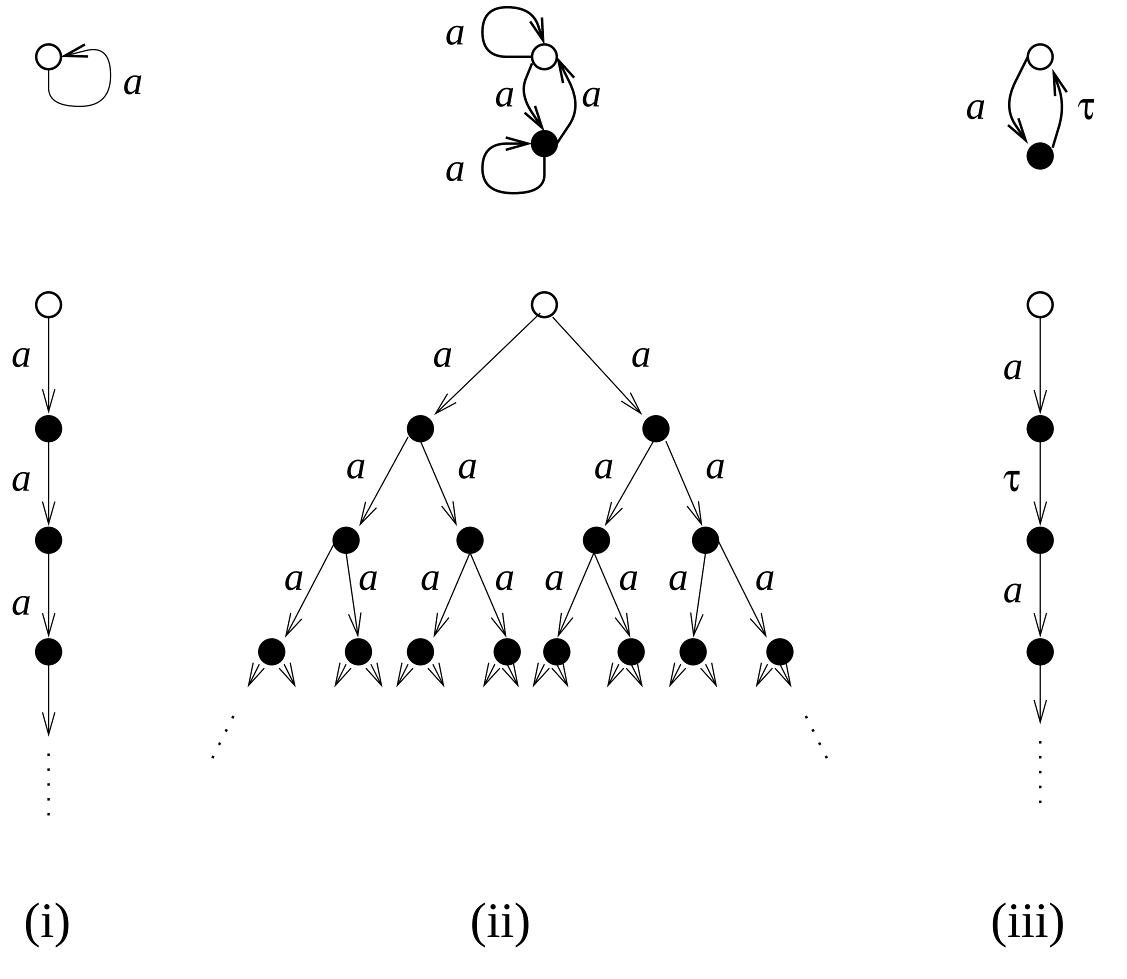


Example LTS with ✓



Synchronisation trees

There are potentially many LTS's displaying essentially the same behaviour (i.e., they are not nearly *abstract* enough for deciding process equivalence. For example, any LTS can be unwound into a *synchronisation tree*



All these processes look the same to an observer!

Bisimulation

Some process algebras (notably CCS) take LTS's to be their main semantic basis, and require direct methods for deciding when two have equivalent behaviour. (For example, modal logics of experimentation on them.)

In a sense, we will define our equivalence relations later, but for now we define the most basic equivalence relation over LTS's.

A relation R on the nodes $\hat{S}$ of an LTS S is a *strong bisimulation* if:

$$\begin{array}{l}\forall\, n_1, n_2, m_1 \in \hat{S}. \forall\, x \in \Sigma^{\checkmark,\tau}. \\ n_1 R\, n_2 \wedge n_1 \xrightarrow{x} m_1 \Rightarrow \exists\, m_2 \in \hat{S}. n_2 \xrightarrow{x} m_2 \wedge m_1 R\, m_2\end{array}$$

$$\begin{array}{l}\forall\, n_1, n_2, m_2 \in \hat{S}. \forall\, x \in \Sigma^{\checkmark,\tau}. \\ n_1 R\, n_2 \wedge n_2 \xrightarrow{x} m_2 \Rightarrow \exists\, m_1 \in \hat{S}. n_1 \xrightarrow{x} m_1 \wedge m_1\, R\, m_2\end{array}$$

This says that, if $m\,R\,n$ we cannot tell m and n apart by experimentation proceeding round S on the assumption that all we can see about a node is its set of initial actions.

It treats all actions (including τ and $\checkmark$) as equals.

Maximal bisimulation

The function from relations to relations: $(n,m) \in \Psi(R)$ if and only if $n = m$ or

$$\begin{array}{l}\forall\, n' \in \hat{S}.\, \forall\, x \in \Sigma^{\checkmark,\tau}. \\ n \xrightarrow{x} n' \Rightarrow \exists\, m' \in \hat{V}.m \xrightarrow{x} m' \wedge n' R\, m' \qquad \text{and}\end{array}$$

$$\begin{array}{l}\forall\, m' \in \hat{S}.\, \forall\, x \in \Sigma^{\checkmark,\tau}. \\ m \xrightarrow{x} m' \Rightarrow \exists\, n' \in \hat{V}.n \xrightarrow{x} n' \wedge n' R\, m'\end{array}$$

is monotone:

$$R \subseteq R' \Rightarrow \Psi(R) \subseteq \Psi(R')$$

And R is a strong bisimulation if and only if

$$R \subseteq \Psi(R)$$

i.e., R is a *pre-fixed* point of Ψ.

The greatest fixed point of a monotone function f on a complete lattice (such as the relations) is

$$\bigsqcup \{x \mid x \leq f(x)\}$$

Therefore there is always a *maximal bisimulation* which is a fixed point of Ψ and is the union of all bisimulations.

Bisimilarity equivalence

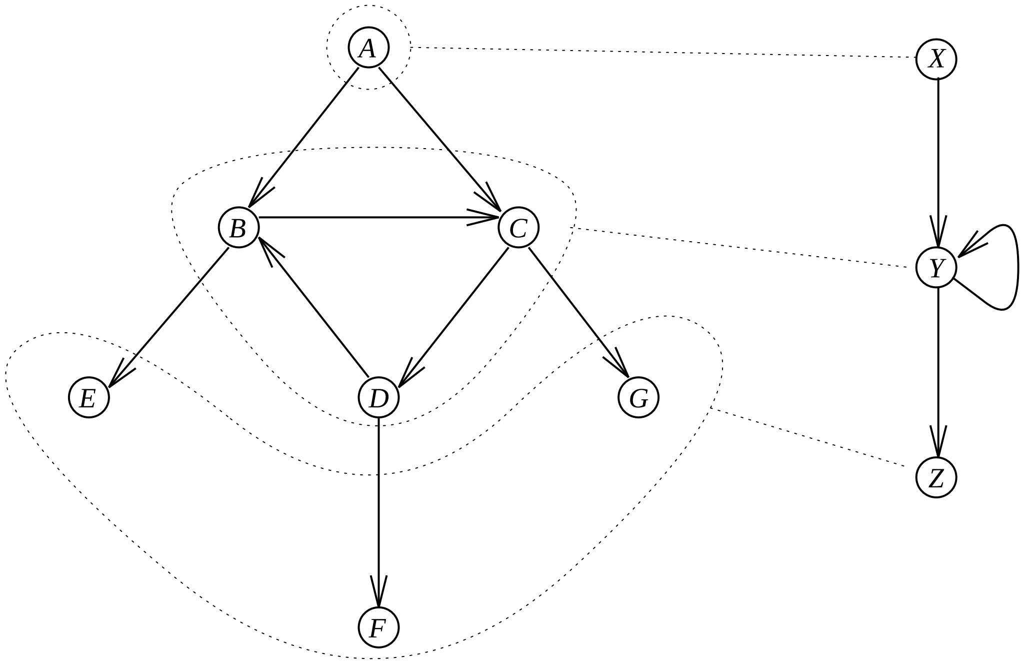
Since "$=$" is a bisimulation,

if R is one then so is R^{-1}, and

if R, R' are bisimulations then so is $R \circ R'$

it follows that the maximal bisimulation is an *equivalence relation* (reflexive, symmetric and transitive).

You can always factor an LTS by its maximal bisimulation to get a smaller one which is, in a sense, extensional with respect to the transition relations. *There is an important relationship with (non-well-founded) set theory.*



Firing rules for CSP

The operational semantics treats the set of *closed* process terms as a (large!) LTS.

It takes the form of an inference system which allows to compute the initial actions of any process, together with the result state from each.

An action/result pair is possible if and only if it is deducible using the rules.

Thus there are no rules for $STOP$.

The following deduction requires no preconditions:

$$\frac{}{SKIP \xrightarrow{\checkmark} \Omega}$$

Ω represents an already-terminated process.

Prefixing

In $e \to P$, e might be complex, involving inputs and bindings:

- $comms(e)$ is the set of communications represented by the prefix e. For example, $d.3$ represents $\{d.3\}$ and $c?x{:}A?y$ represents $\{c.a.b \mid a.b \in type(c), a \in A\}$.
- For $a \in comms(e)$, $subs(a, e, P)$ is the result of substituting the appropriate part of a for each identifier in P bound by e. This equals P if there are no identifiers bound (as when e is $d.3$). For example, $subs(c.1.2, c?x?y, d!x \to P(x, y)) = d!1 \to P(1, 2)$.

We then get the rule:

$$\frac{}{e \to P \xrightarrow{a} subs(a, e, P)}\ (a \in comms(e))$$

It would arguably be better to replace the insistence that terms are closed wrt sub-process values by *semantic environments*.

Nondeterministic choice and recursion

Other constructs that give their initial actions without precondition:

$$\frac{}{P \sqcap Q \xrightarrow{\tau} P} \qquad \frac{}{P \sqcap Q \xrightarrow{\tau} Q}$$

$$\frac{}{\bigsqcap S \xrightarrow{\tau} P} \quad (P \in S)$$

In these, we are using the nondeterminism and eagerness of τ to make our selections for us.

The next rule uses the fact that $\tau.P$ is indistinguishable from P:

$$\frac{}{\mu\, p.P \xrightarrow{\tau} P[\mu\, p.P/p]}$$

It allows to give a meaning to all recursions, even $\mu\, p.p$ (identified with $\mathbf{div}$, the simply divergent process) and means there is no infinite regression in our inference system.

Semantic environments would allow us to deal with parameterised and mutual recursions better,

External choice

Since $P \,\Box\, Q$'s actions depend on the initial actions of both arguments

(i) We get non-trivial preconditions to inference rules

(ii) τ actions from any *switched-on* argument are simply promoted, since the operator does not control or observe them.

$$\frac{P \xrightarrow{\tau} P'}{P \,\Box\, Q \xrightarrow{\tau} P' \,\Box\, Q} \qquad \frac{Q \xrightarrow{\tau} Q'}{P \,\Box\, Q \xrightarrow{\tau} P \,\Box\, Q'}$$

Visible actions resolve the choice:

$$\frac{P \xrightarrow{a} P'}{P \,\Box\, Q \xrightarrow{a} P'}(a \neq \tau) \qquad \frac{Q \xrightarrow{a} Q'}{P \,\Box\, Q \xrightarrow{a} Q'}(a \neq \tau)$$

Sequential composition

In $P;\ Q$, P runs normally until it terminates:

$$\frac{P \xrightarrow{x} P'}{P;\ Q \xrightarrow{x} P';\ Q} \quad (x \neq \checkmark)$$

Notice that, in the above, Q is not switched on. This is semantically important, for otherwise it would be right-divergence-strict.

If and when it does terminate, the termination is hidden from the outside and it hands over control to Q

$$\frac{\exists\, P'.P \xrightarrow{\checkmark} P'}{P;\ Q \xrightarrow{\tau} Q}$$

Hiding and renaming

These simply apply the appropriate mappings to events, being careful that the state after ✓ is Ω.

$$\frac{P \xrightarrow{x} P'}{P \setminus B \xrightarrow{x} P' \setminus B} \quad (x \notin B \cup \{\checkmark\})$$

$$\frac{P \xrightarrow{\checkmark} P'}{P \setminus B \xrightarrow{\checkmark} \Omega}$$

$$\frac{P \xrightarrow{a} P'}{P \setminus B \xrightarrow{\tau} P' \setminus B} \quad (a \in B)$$

$$\frac{P \xrightarrow{\tau} P'}{P[\![R]\!] \xrightarrow{\tau} P'[\![R]\!]} \qquad \frac{P \xrightarrow{\checkmark} P'}{P[\![R]\!] \xrightarrow{\checkmark} \Omega}$$

$$\frac{P \xrightarrow{a} P'}{P[\![R]\!] \xrightarrow{b} P'[\![R]\!]} \quad (a\,R\,b)$$

Parallel

$$\frac{P \xrightarrow{\tau} P'}{P \underset{X}{\parallel} Q \xrightarrow{\tau} P' \underset{X}{\parallel} Q} \qquad \frac{Q \xrightarrow{\tau} Q'}{P \underset{X}{\parallel} Q \xrightarrow{\tau} P \underset{X}{\parallel} Q'}$$

for $a \notin X$

$$\frac{P \xrightarrow{a} P'}{P \underset{X}{\parallel} Q \xrightarrow{a} P' \underset{X}{\parallel} Q} \quad (a \in \Sigma \setminus X)$$

$$\frac{Q \xrightarrow{a} Q'}{P \underset{X}{\parallel} Q \xrightarrow{a} P \underset{X}{\parallel} Q'} \quad (a \in \Sigma \setminus X)$$

$$\frac{P \xrightarrow{a} P' \quad Q \xrightarrow{a} Q'}{P \underset{X}{\parallel} Q \xrightarrow{a} P' \underset{X}{\parallel} Q'} \quad (a \in X)$$

Distributed termination:

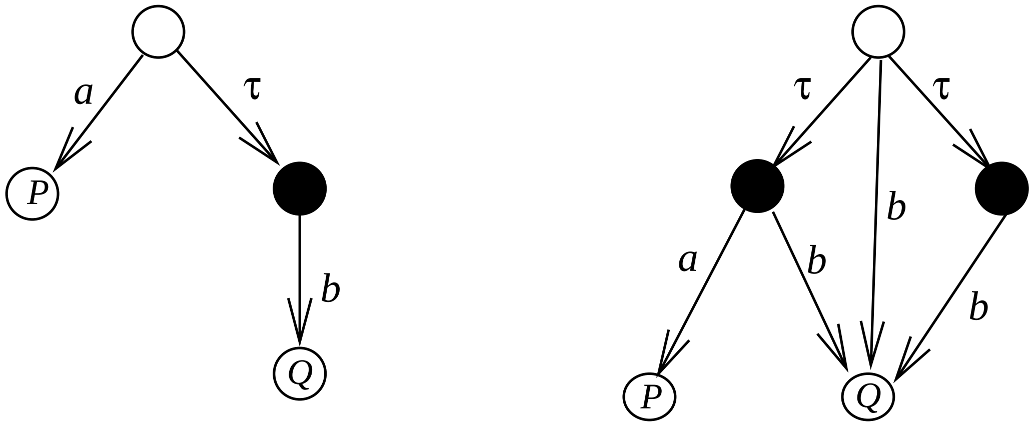
$$\frac{P \xrightarrow{\checkmark} P'}{P \underset{X}{\parallel} Q \xrightarrow{\tau} \Omega \underset{X}{\parallel} Q} \qquad \frac{Q \xrightarrow{\checkmark} Q'}{P \underset{X}{\parallel} Q \xrightarrow{\tau} P \underset{X}{\parallel} \Omega}$$

$$\frac{}{\Omega \underset{X}{\parallel} \Omega \xrightarrow{\checkmark} \Omega}$$

Time out/sliding choice

All other operators can have their operational semantics derived from the above. These usually give natural results, the exception being $P \triangleright Q = (P \sqcap STOP) \Box Q$.

Consider $(a \to P) \triangleright (b \to Q)$:



We get the simpler result from

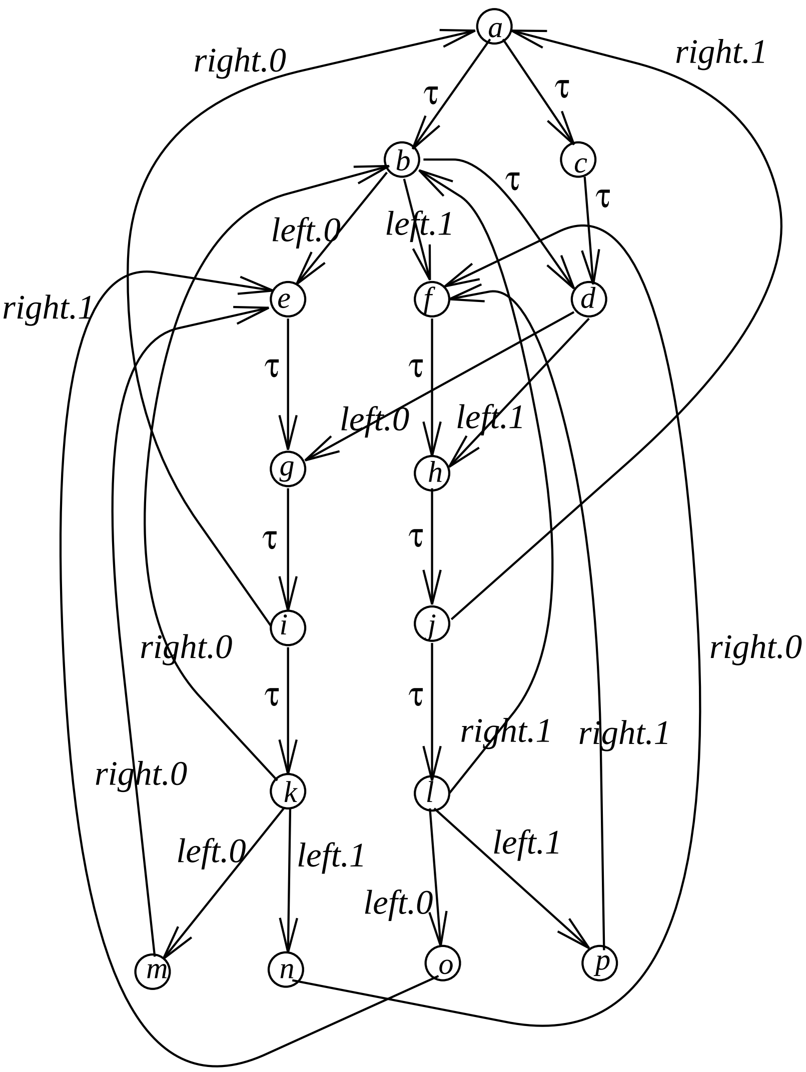
$$\frac{P \xrightarrow{\tau} P'}{P \triangleright Q \xrightarrow{\tau} P' \triangleright Q}$$

$$\frac{P \xrightarrow{x} P'}{P \triangleright Q \xrightarrow{x} P'} \quad (x \neq \tau)$$

$$\frac{}{P \triangleright Q \xrightarrow{\tau} Q}$$

Example

Transition systems get complex for humans to manipulate rather quickly: $COPY \gg COPY$ (for type $\{0, 1\}$) gives



Extended transition relations

If $P, Q \in \hat{C}$ and $s = \langle x_i \mid 0 \leq i < n \rangle \in (\Sigma^\tau)^{*\checkmark}$ then $P \overset{s}{\longmapsto} Q$ if there are $P_0 = P, P_1, \ldots, P_n = Q$ such that $P_k \xrightarrow{x_k} P_{k+1}$ for $k \in \{0, 1, \ldots, n-1\}$.

The second ignores these τs: $s \in \Sigma^{*\checkmark}$ $P \overset{s}{\Longrightarrow} Q$ $s' \in (\Sigma^\tau)^{*\checkmark}$ such that $P \overset{s'}{\longmapsto} Q$ and $s' \setminus \tau = s$.

$$
\begin{array}{ll}
\text{(a)} & P \overset{\langle\rangle}{\Longrightarrow} P \ \wedge \ P \overset{\langle\rangle}{\longmapsto} P \\
\text{(b)} & P \overset{s}{\Longrightarrow} Q \wedge Q \overset{t}{\Longrightarrow} R \Rightarrow P \overset{s\hat{}t}{\Longrightarrow} R \\
\text{(c)} & P \overset{s}{\longmapsto} Q \wedge Q \overset{t}{\longmapsto} R \Rightarrow P \overset{s\hat{}t}{\longmapsto} R \\
\text{(d)} & P \overset{s\hat{}t}{\Longrightarrow} R \Rightarrow \exists\, Q . P \overset{s}{\Longrightarrow} Q \wedge Q \overset{t}{\Longrightarrow} R \\
\text{(e)} & P \overset{s\hat{}t}{\longmapsto} R \Rightarrow \exists\, Q . P \overset{s}{\longmapsto} Q \wedge Q \overset{t}{\longmapsto} R
\end{array}
$$

Observing behaviour

We can extract a process' traces from its operational semantics:

$$traces(P) = \{s \in \Sigma^{*\checkmark} \mid \exists\, Q.P \stackrel{s}{\Longrightarrow} Q\}$$

P can *diverge*, written $P\,div$, if there exist $P_0 = P, P_1, P_2, \ldots$ such that, for all $n \in \mathbb{N}$, $P_n \stackrel{\tau}{\longrightarrow} P_{n+1}$. Then

$$\begin{aligned} divergences(P) \quad &= \quad \{s\hat{\ }t \mid s \in \Sigma^* \wedge t \in \Sigma^{*\checkmark} \\ &\qquad\qquad \wedge \exists\, Q.P \stackrel{s}{\Longrightarrow} Q \wedge Q\ div\} \end{aligned}$$

Note that this is extension closed for consistency with denotational models. Here it is a matter of choice, but it will be a great help later.

Refusals and failures

$P\ \mathit{ref}\ B$ ($B \subseteq \Sigma^{\checkmark}$) if and only if *either* P is stable and $B \cap P^0 = \{\}$ *or* there is Q with $P \xrightarrow{\checkmark} Q$ and $B \subseteq \Sigma$.

(The latter because a state that can terminate may choose to do so and give the environment no other option.)

$$\begin{array}{rcl} \mathit{failures}(P) & = & \{(s, X) \mid \exists\, Q.P \overset{s}{\Longrightarrow} Q \wedge Q\ \mathit{ref}\ X\} \\ & & \cup\{(s\hat{}\langle\checkmark\rangle, X) \mid \exists\, Q.P \overset{s\hat{}\langle\checkmark\rangle}{\Longrightarrow} Q\} \end{array}$$

It is sometimes necessary to ignore what a process does after possible divergence. Therefore we define the following:

$$\begin{array}{rcl} \mathit{traces}_{\perp}(P) & = & \mathit{traces}(P) \cup \mathit{divergences}(P) \\ \mathit{failures}_{\perp}(P) & = & \mathit{failures}(P) \\ & & \cup\ \{(s, X) \mid s \in \mathit{divergences}(P)\} \end{array}$$

Infinite traces

The notations $\overset{u}{\longmapsto}$ and $\overset{u}{\Longrightarrow}$ extend to infinite u, becoming unary relations

- If $u = \langle x_i \mid i \in \mathbb{N}\rangle \in (\Sigma^\tau)^\omega$ $P \overset{u}{\longmapsto}$ if there exist $P_0 = P, P_1, P_2, \ldots$ such that $P_i \overset{x_i}{\longrightarrow} P_{i+1}$ for all i.
- If $u \in \Sigma^\omega$ then $P \overset{u}{\Longrightarrow}$ if $u' \in (\Sigma^\tau)^\omega$ such that $P \overset{u'}{\longmapsto}$ and $u = u' \setminus \tau$.

$\checkmark$s, being final, play no part in infinite traces.

$u' \in (\Sigma^\tau)^\omega$ may have $u' \setminus \tau$ finite
in which case it is a divergence.

$$\begin{array}{rcl} \mathit{infinites}(P) & = & \{u \mid P \overset{u}{\Longrightarrow}\} \cup \\ & & \{s\hat{\ }u \mid s \in \mathit{divergences}(P) \cap \Sigma^* \wedge u \in \Sigma^\omega\} \end{array}$$

This is closed under divergence by choice for consistency with denotational models, and for a reason we will see shortly..

Closure of trace sets

Let $Traces(P) = traces_{\perp}(P) \cup infinites(P)$

Evidently this is prefix-closed. The infinite traces are completely determined by the finite ones when

$$Traces(P) = \overline{traces_{\perp}(P)}$$

where $\overline{A} = A \cup \{u \in \Sigma^{\omega} \mid \{s \mid s < u\} \subseteq A\}$.

$Traces(P)$ is *closed* when this happens.

LTS's with the following property always satisfy this

C is said to be *finitely branching* if for all nodes P and each $x \in \Sigma^{\checkmark,\tau}$, the set

$$\{Q \mid P \xrightarrow{x} Q\}$$

is finite.

König's Lemma

This fact is a corollary to the following result:

Suppose X_i is, for each $i \in \mathbb{N}$, a nonempty finite set and that $f_i : X_{i+1} \to X_i$ is a (total) function. Then there is a sequence $\langle x_i \mid i \in \mathbb{N} \rangle$ such that $x_i \in X_i$ and $f_i(x_{i+1}) = x_i$. ▮

because, when $u \in \overline{traces_\perp(P)}$ is infinite, the sets

$$X_n = \{(Q, s) \mid s \in (\Sigma^\tau)^n \wedge P \stackrel{s}{\longmapsto} Q \wedge s \setminus \tau < u\}$$

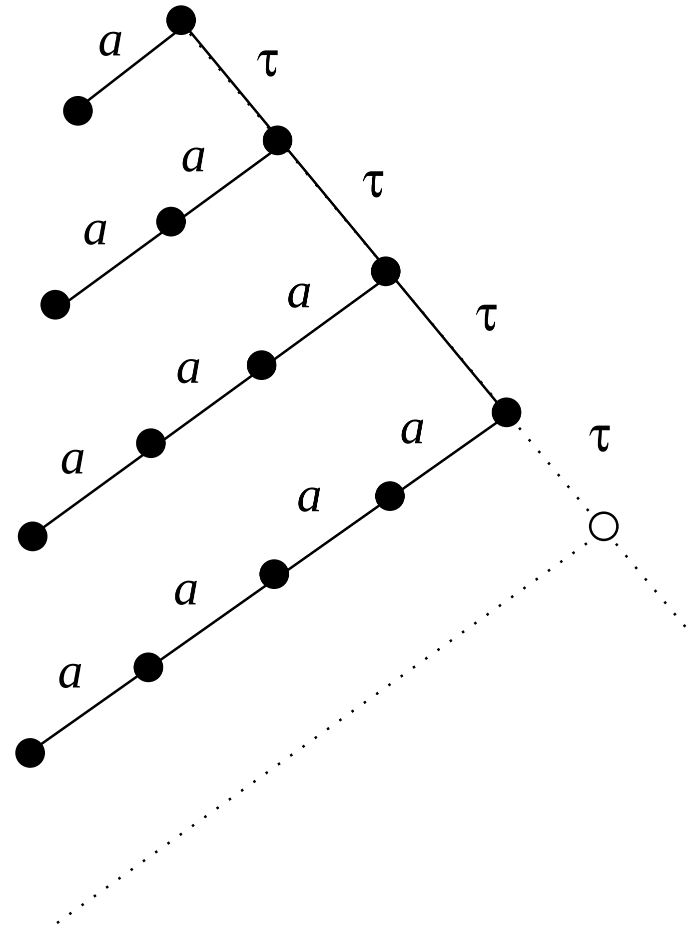
satisfy this, proving that

either u is an infinite trace

or some $s < u$ is a divergence.

König's Lemma is often stated in graph-theoretic terms: any finitely-branching tree with nodes at every (natural number) depth below a root r has an infinite path from r.

Divergence from König's Lemma



The finite trace $\langle a, a, a, \ldots\rangle$ belongs to $\overline{traces(P)}$, but only to $\mathit{infinites}(P)$ by divergence-closure.

Proof of König's Lemma

We can compose f_i to get, for $r > s$, functions $f_{r,s} : X_r \to X_s$:

$$f_{r,s} = f_s \circ \ldots \circ f_{r-1}$$

Fix s in $\mathbb{N}$.
$f_{r,s}(X_r)$ is a decreasing series of subsets of X_s for $r > s$, since

$$\begin{array}{rcl} f_{r+1,s}(X_{r+1}) & = & f_{r,s}(f_r(X_{r+1})) \\ & \subseteq & f_{r,s}(X_r) \end{array}$$

and hence eventually constant. Thus

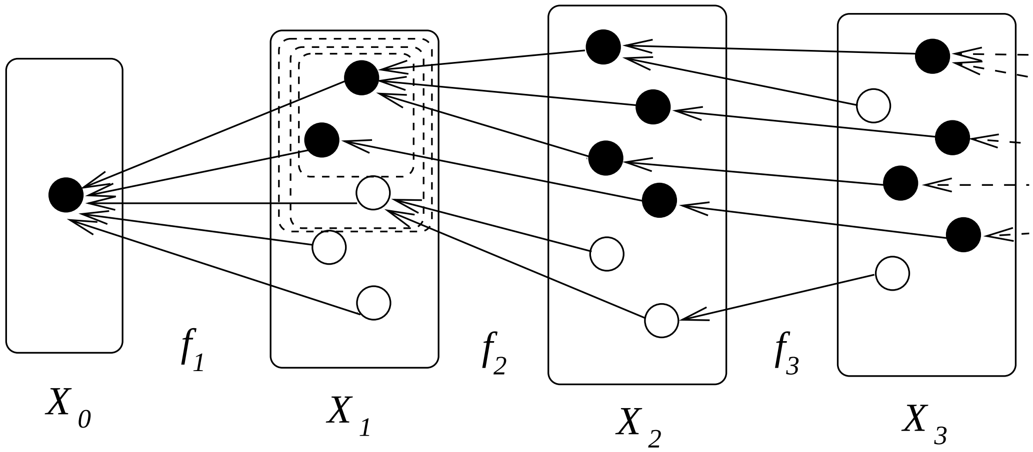
$$X_s^* = \bigcap_{r>s} f_{r,s}(X_r) \neq \{\}$$

and equals $f_{r,s}(X_r)$ for all sufficiently large r. For all s, $f_s(X_{s+1}^*) = X_s^*$.

The X_s^*, $f_s \upharpoonright X_{s+1}^*$ thus satisfy the assumptions of the lemma with the addition of the functions being onto.

Now simply pick $x_0 \in X_0^*$, and for each s let x_{s+1} be any member of X_{s+1}^* such that $f_s(x_{s+1}) = x_s$.

Pictorial proof of König's Lemma



For this initial group of sets, we see the chain of decreasing subsets of X_1 forming, and (in black) the nodes which, on the evidence before us, should be in the X_i^*.

Unbounded nondeterminism

This result makes it important to understand which CSP operators can generate infinitely-branching LTS's

because we will then not be able to infer infinite traces, and hence divergences after hiding, from finite ones.

For example, if $A_0 = STOP$ and $A_{n+1} = a \rightarrow A_n$,

$$\begin{array}{rcl} A^* & = & \bigsqcap\{A_n \mid n \in \mathbb{N}\} \\ AS & = & STOP \sqcap a \rightarrow AS \end{array}$$

have identical sets of finite traces, but different infinite ones.

The operators that can do this are $\bigsqcap$, infinite hiding and infinite-to-one renaming (the last two only being possible for infinite Σ).

One can prove inductively that any process definition without these is always finitely nondeterministic and therefore has a closed $Traces(P)$.