Failure Trace Semantics for a Process Algebra with Time-Outs

Rob van Glabbeek

University of Edinburgh

CSIRO, Sydney, Australia

University of New South Wales, Sydney, Australia

September 2023

Context

Classic process algebra

Classic semantics in terms of labelled transition systems

Context

Classic process algebra

Classic semantics in terms of labelled transition systems

No probabilities, no quantified time, no name binding, no nothing

Context

Classic process algebra

Classic semantics in terms of labelled transition systems

No probabilities, no quantified time, no name binding, no nothing

uninterpreted visible actions a, b, c

Context

Classic process algebra

Classic semantics in terms of labelled transition systems

No probabilities, no quantified time, no name binding, no nothing

uninterpreted visible actions a, b, c — instantaneous

Context

Classic process algebra

Classic semantics in terms of labelled transition systems

No probabilities, no quantified time, no name binding, no nothing

uninterpreted visible actions a, b, c — instantaneous

hidden action τ — unobservable and instantaneous

Context

Classic process algebra

Classic semantics in terms of labelled transition systems

No probabilities, no quantified time, no name binding, no nothing

uninterpreted visible actions a, b, c — instantaneous

hidden action τ — unobservable and instantaneous

time-out action t — unobservable and instantaneous

Context

Classic process algebra

Classic semantics in terms of labelled transition systems

No probabilities, no quantified time, no name binding, no nothing

uninterpreted visible actions a, b, c — instantaneous

hidden action τ — unobservable and instantaneous

novelty: time-out action t — unobservable and instantaneous

Context

Classic process algebra

Classic semantics in terms of labelled transition systems

No probabilities, no quantified time, no name binding, no nothing

uninterpreted visible actions a, b, c — instantaneous

hidden action τ — unobservable and instantaneous

novelty: time-out action t — unobservable and instantaneous
models the end of a time-consuming activity from which we abstract.

Context

Classic process algebra

Classic semantics in terms of labelled transition systems

No probabilities, no quantified time, no name binding, no nothing

uninterpreted visible actions a, b, c — instantaneous

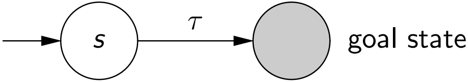
hidden action τ — unobservable and instantaneous

novelty: time-out action t — unobservable and instantaneous
models the end of a time-consuming activity from which we abstract.

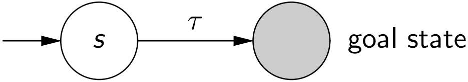
Goal: *find the coarsest reasonable semantics for LTSs with* t.

The passage of time in labelled transition systems

The passage of time in labelled transition systems

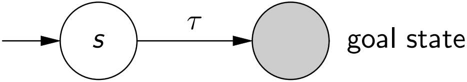


The passage of time in labelled transition systems



I want to assume that one reaches this goal state.
(This assumption is called *progress* in [GH19].)
Without this assumption, no useful liveness properties can be established.

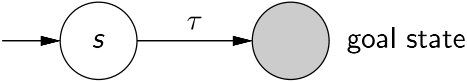
The passage of time in labelled transition systems



I want to assume that one reaches this goal state.
(This assumption is called *progress* in [GH19].)
Without this assumption, no useful liveness properties can be established.

If we allow to system to idle a finite amount of time in state s, and during this time nothing happens at all, by what mechanism will the system ever continue?

The passage of time in labelled transition systems

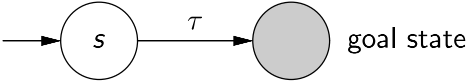


I want to assume that one reaches this goal state.
(This assumption is called *progress* in [GH19].)
Without this assumption, no useful liveness properties can be established.

If we allow to system to idle a finite amount of time in state s, and during this time nothing happens at all, by what mechanism will the system ever continue?

Within state s we await the completion of a task from with we abstract.

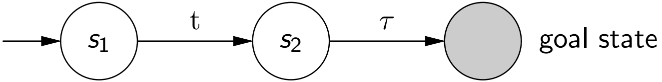
The passage of time in labelled transition systems



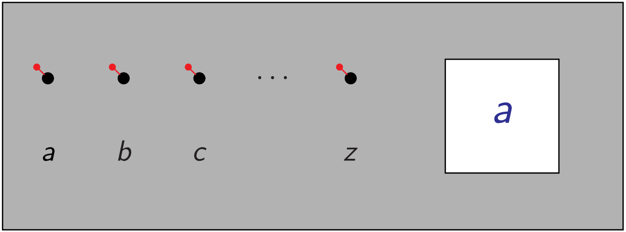
I want to assume that one reaches this goal state.
(This assumption is called *progress* in [GH19].)
Without this assumption, no useful liveness properties can be established.

If we allow to system to idle a finite amount of time in state s, and during this time nothing happens at all, by what mechanism will the system ever continue?

Within state s we await the completion of a task from with we abstract.

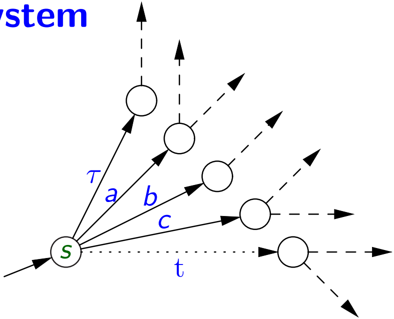


The environment

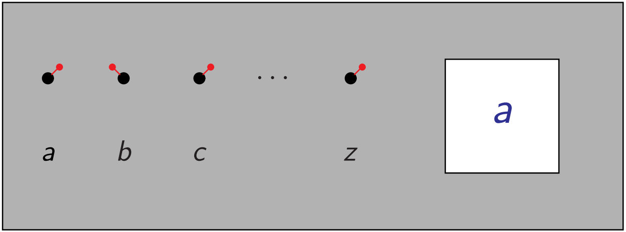


Right blocks

The system

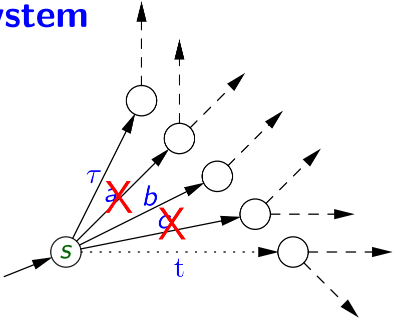


The environment



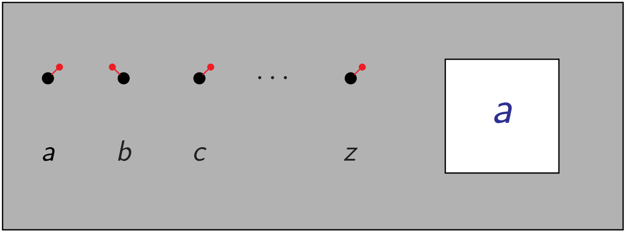
Right blocks

The system



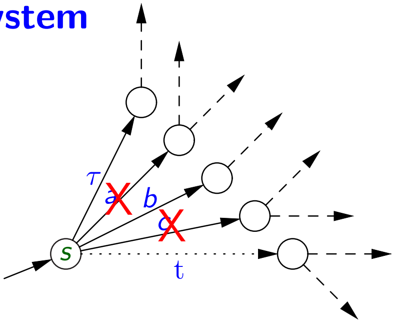
User blocks *a* and *c*:
System makes nondeterministic choice between τ and *b*.

The environment



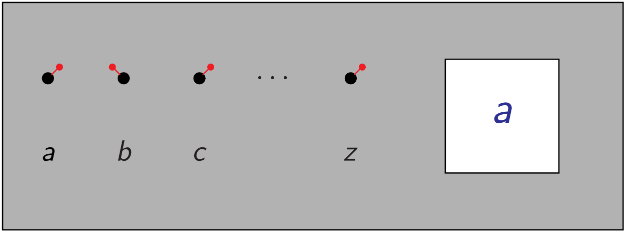
Right blocks

The system



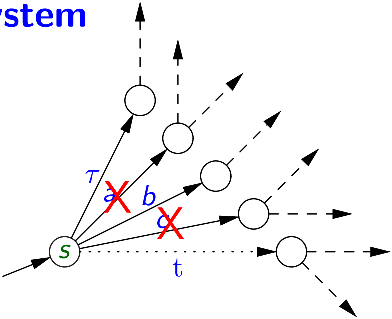
User blocks *a* and *c*:
System makes nondeterministic choice between τ and *b*.
Will not choose t.

The environment



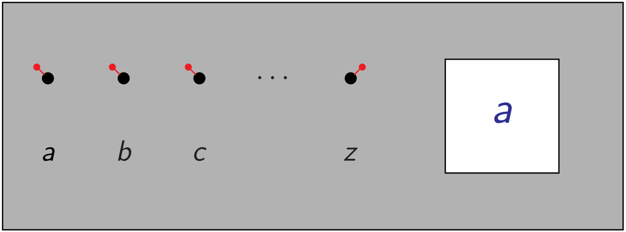
Right blocks

The system



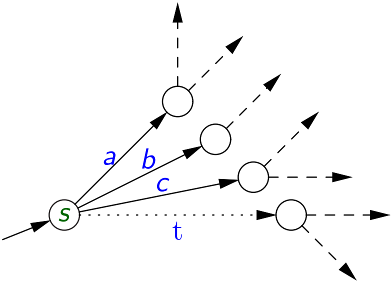
User blocks a and c:
System makes nondeterministic choice between τ and b.
Will not choose t.
No time spend in state s.

The environment



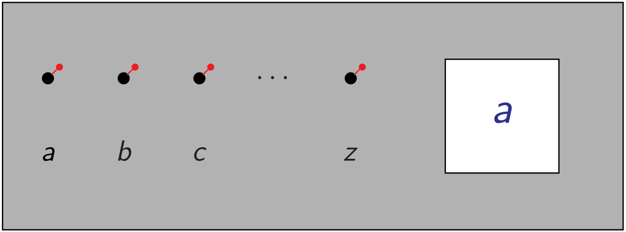
Right blocks

The system



System chooses a, b or c.

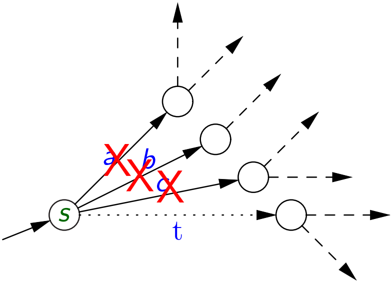
The environment



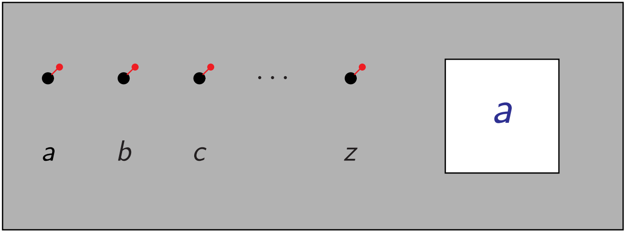
Right blocks

The system

User blocks a, b, and c:

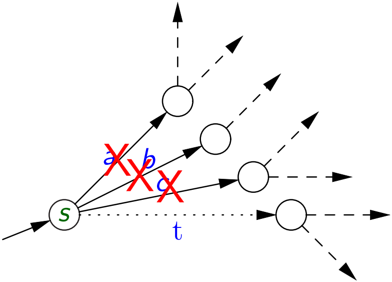


The environment



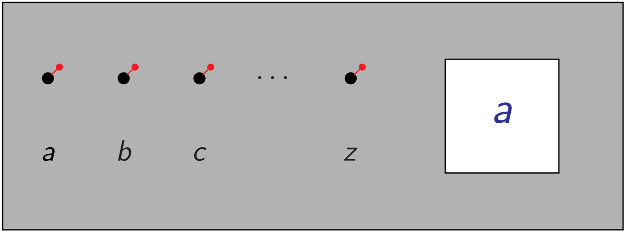
Right blocks

The system



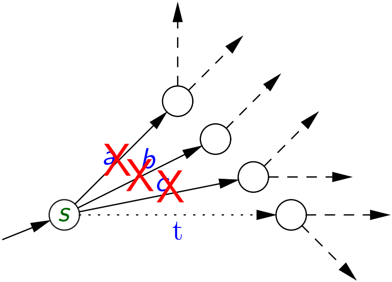
User blocks a, b, and c:
System stays in state s.

The environment



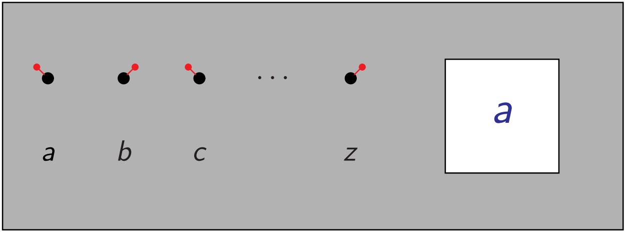
Right blocks

The system



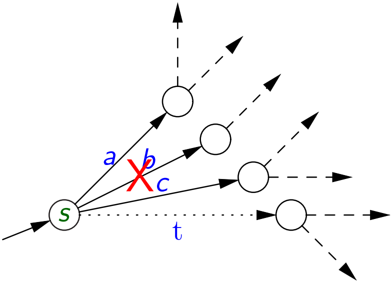
User blocks a, b, and c:
System stays in state s.
User may change mind.

The environment



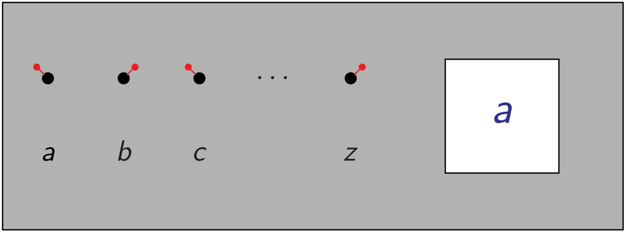
Right blocks

The system



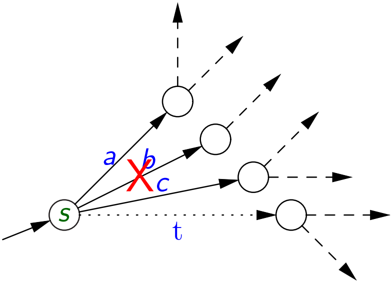
User blocks a, b, and c:
System stays in state s.
User may change mind.

The environment



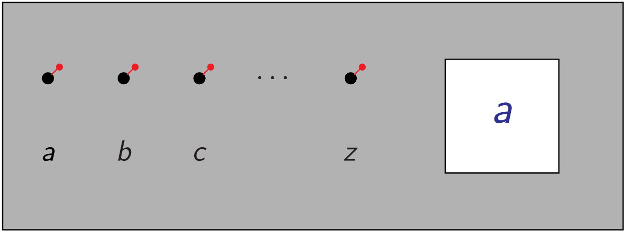
Right blocks

The system



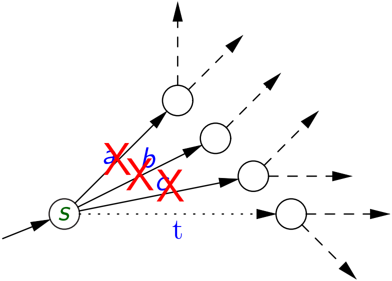
User blocks *a*, *b*, and *c*:
System stays in state *s*.
User may change mind.
System chooses *a* or *c*.

The environment



Right blocks

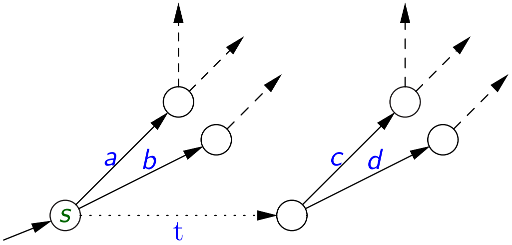
The system



User blocks *a*, *b*, and *c*:
System stays in state *s*.
User may change mind.

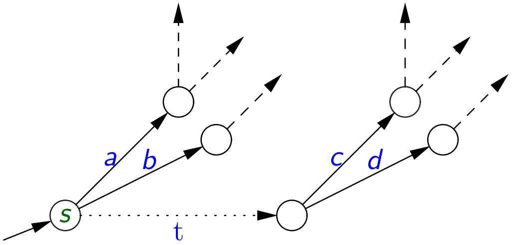
Or time-out occurs first and system takes t.

More expressiveness



We can now express priorities.

More expressiveness



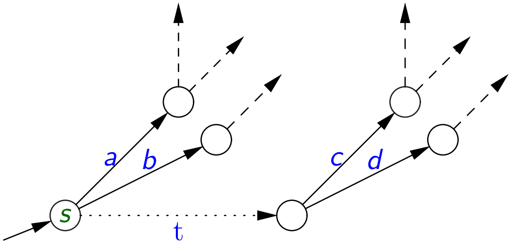
We can now express priorities.

It has been shown [GH15] that mutual exclusion cannot be adequately expressed in CCS-like languages. We need to
(a) extend the language, e.g. with "signals", and
(b) adopt "justness", a very weak fairness assumption

(weaker than "weak fairness").

(Bouwman et al. show that (a) and (b) can be combined into one assumption.)

More expressiveness



We can now express priorities.

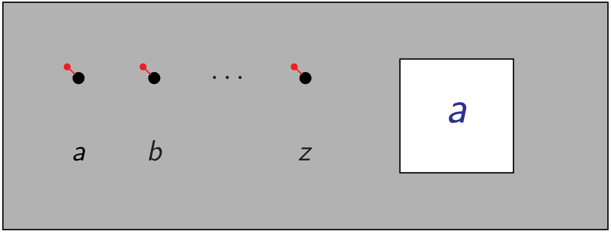
It has been shown [GH15] that mutual exclusion cannot be adequately expressed in CCS-like languages. We need to
(a) extend the language, e.g. with “signals”, and
(b) adopt “justness”, a very weak fairness assumption

(weaker than “weak fairness”).

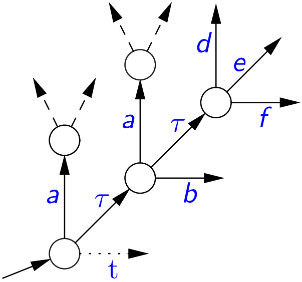
(Bouwman et al. show that (a) and (b) can be combined into one assumption.)

In the presence of time-out transitions mutual exclusion can be correctly expressed in CCS, even without assuming justness.

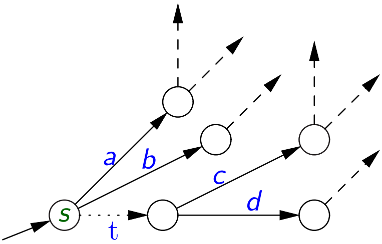
The environment



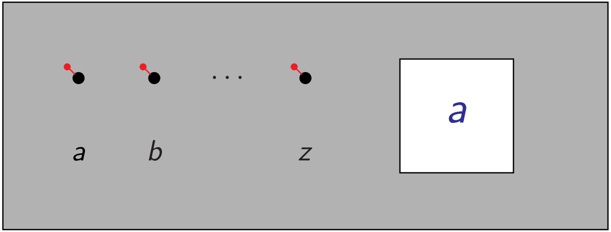
Environment $\|_A$ System



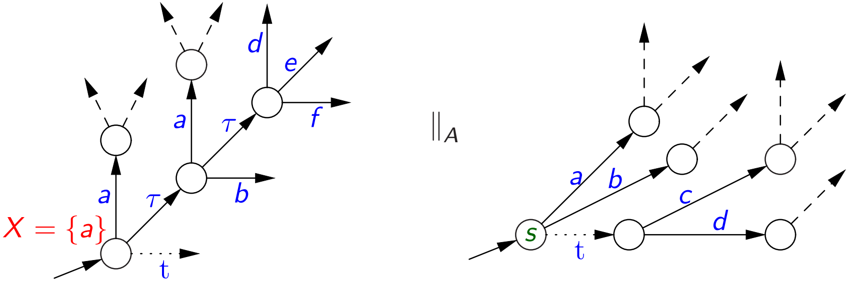
$\|_A$



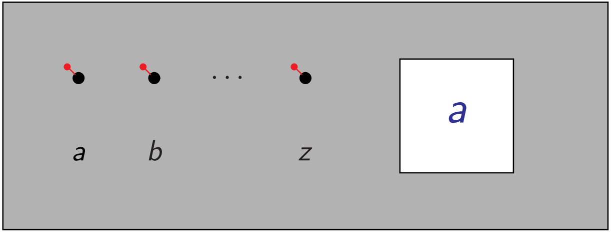
The environment



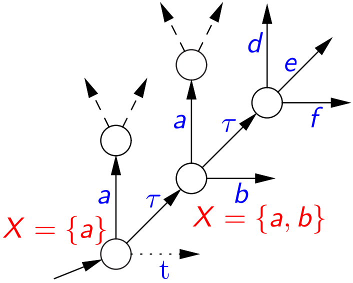
Environment $\|_A$ System



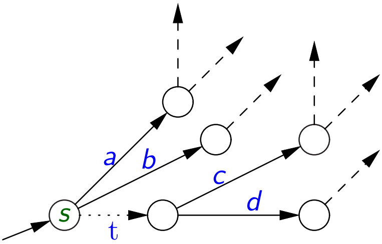
The environment



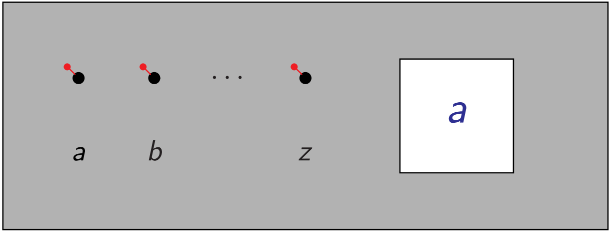
Environment $\|_A$ System



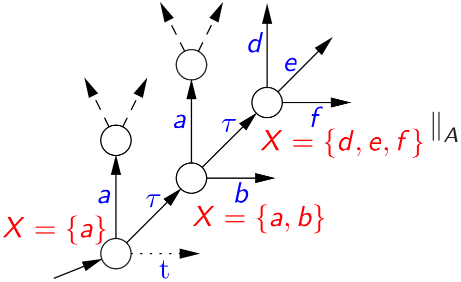
$\|_A$

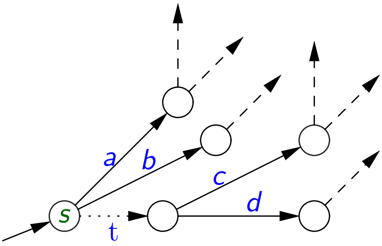


The environment



Environment $\|_A$ System





The process algebra $\mathrm{CCSP_t}$

$$\begin{array}{rl} E ::= & 0 \;\big|\; \alpha.E \;\big|\; E+E \;\big|\; E \,\|_S\, E \;\big|\; \tau_I(E) \;\big|\; \mathcal{R}(E) \;\big| \\ & x \;\big|\; \langle x|\mathcal{S}\rangle \text{ (with } x \in V_\mathcal{S}) \end{array}$$

with $\alpha \in Act := A \uplus \{\tau, \mathrm{t}\}$, $S, I \subseteq A$, $\mathcal{R} \subseteq A \times A$, $x \in Var$ and $\mathcal{S}$ a *recursive specification*:
a set of equations $\{y = \mathcal{S}_y \mid y \in V_\mathcal{S}\}$ with $V_\mathcal{S} \subseteq Var$ (the *bound variables* of $\mathcal{S}$) and each $\mathcal{S}_y$ a $\mathrm{CCSP_t}$ expression.

$$\alpha.x \xrightarrow{\alpha} x \qquad \frac{x \xrightarrow{\alpha} x'}{x+y \xrightarrow{\alpha} x'} \qquad \frac{y \xrightarrow{\alpha} y'}{x+y \xrightarrow{\alpha} y'} \qquad \frac{x \xrightarrow{\alpha} x'}{\mathcal{R}(x) \xrightarrow{\beta} \mathcal{R}(x')} \begin{pmatrix} \alpha=\beta=\tau \\ \vee\ \alpha=\beta=\mathrm{t} \\ \vee\ (\alpha,\beta)\in\mathcal{R} \end{pmatrix}$$

$$\frac{x \xrightarrow{\alpha} x'}{x \,\|_S\, y \xrightarrow{\alpha} x' \,\|_S\, y}\ (\alpha \notin S) \qquad \frac{x \xrightarrow{a} x' \quad y \xrightarrow{a} y'}{x \,\|_S\, y \xrightarrow{a} x' \,\|_S\, y'}\ (a \in S) \qquad \frac{y \xrightarrow{\alpha} y'}{x \,\|_S\, y \xrightarrow{\alpha} x \,\|_S\, y'}\ (\alpha \notin S)$$

$$\frac{x \xrightarrow{\alpha} x'}{\tau_I(x) \xrightarrow{\alpha} \tau_I(x')}\ (\alpha \notin I) \qquad \frac{x \xrightarrow{a} x'}{\tau_I(x) \xrightarrow{\tau} \tau_I(x')}\ (a \in I) \qquad \frac{\langle \mathcal{S}_X|\mathcal{S}\rangle \xrightarrow{\alpha} y}{\langle X|\mathcal{S}\rangle \xrightarrow{\alpha} y}$$

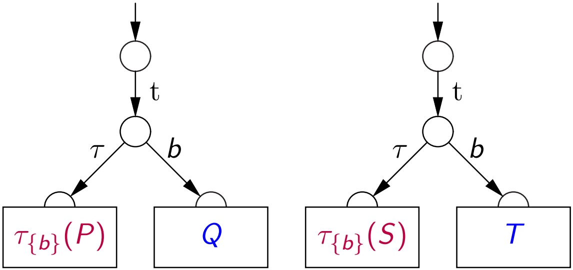
Three crucial laws

$$\tau.P + \mathrm{t}.Q = \tau.P$$

Three crucial laws

$$\tau.P + \mathrm{t}.Q = \tau.P$$

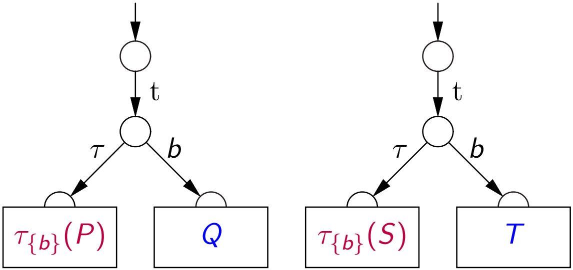
$$\mathrm{t}.(\tau.\tau_{\{b\}}(P) + b.Q) \|_{\{b\}} \mathrm{t}.(\tau.\tau_{\{b\}}(S) + b.T) = \mathrm{t}.\tau.\tau_{\{b\}}(P) \|_{\{b\}} \mathrm{t}.\tau.\tau_{\{b\}}(S)$$



Three crucial laws

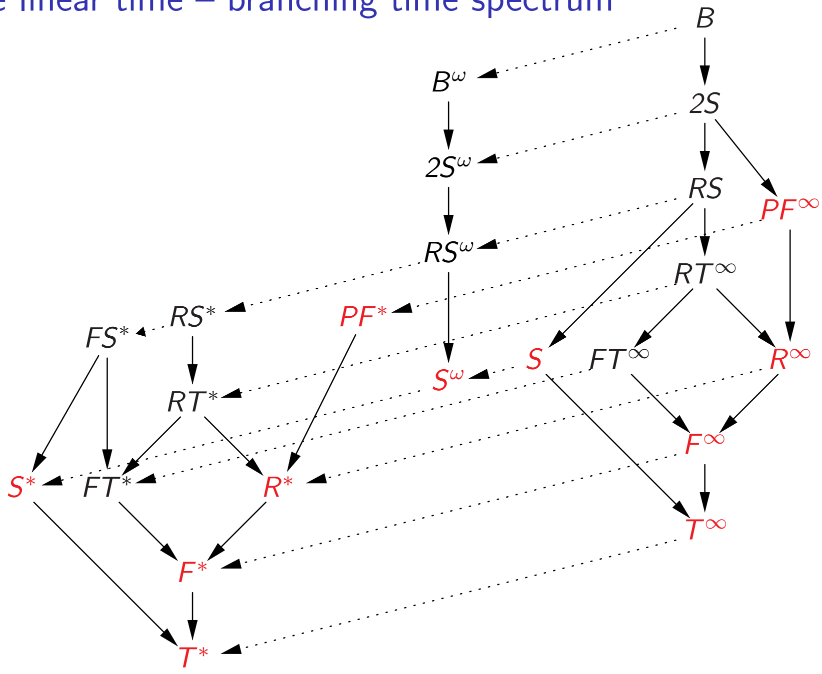
$$\tau.P + \mathrm{t}.Q = \tau.P$$

$$\mathrm{t}.(\tau.\tau_{\{b\}}(P)+b.Q)\|_{\{b\}}\mathrm{t}.(\tau.\tau_{\{b\}}(S)+b.T) = \mathrm{t}.\tau.\tau_{\{b\}}(P)\|_{\{b\}}\mathrm{t}.\tau.\tau_{\{b\}}(S)$$

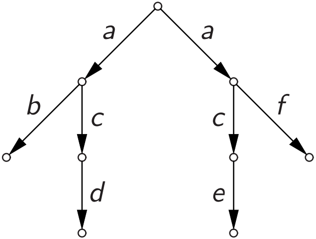


$$a.P + \mathrm{t}.(Q + \tau.R + a.S) = a.P + \mathrm{t}.(Q + \tau.R)$$

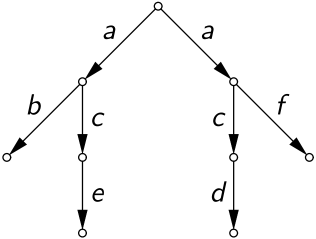
The linear time – branching time spectrum



Trace and failures equivalence fail to be a congruence



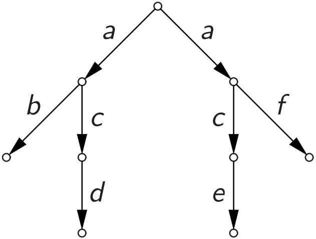
$=_F$
$\neq_{FT}$
$=_R$
$\neq_{RT}$



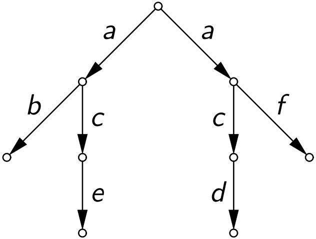
$a.(b + c.d) + a.(f + c.e)$

$a.(b + c.e) + a.(f + c.d)$

Trace and failures equivalence fail to be a congruence



$=_F$
$\neq_{FT}$
$=_R$
$\neq_{RT}$



$a.(b + c.d) + a.(f + c.e)$ $\qquad$ $a.(b + c.e) + a.(f + c.d)$

Use the context $\mathcal{C}[\_] := \tau_{\{a,b,c\}}(a.(b + \mathrm{t}.c) \parallel_{\{a,b,c,f\}} \_)$.

This context implements a priority of b over c.

Only the RHS can ever reach d.

Simulation equivalence fails to be a congruence

Take $P := a.b + a$ and $Q = a.b$, and use the context
$\mathcal{C}[\_] := \tau_{\{a,b\}}(a.(b + \mathrm{t}.d) \parallel_{\{a,b\}} \_)$.
Then only $\mathcal{C}[P]$ can ever perform the action d.

Operational def. of failure trace semantics

Each execution of a system generates a *failure trace*, such as

$$a\, b\, X\, c\, Y\, d\, e\, Z\, W \top$$

a sequence of actions $a \in A$ and sets of refused actions $X \subseteq A$. It is the observation of a sequence of instantaneous actions $a\, b\, c\, d\, e$ interspersed with periods of idling. Each period of idling is denoted by the set $X \subseteq A$ of actions that are offered by the environment during this period. The sequence ends with $\top$, the act of the observer of ending the observation.

$$\top \in FT^*(x) \qquad \frac{x \stackrel{a}{\longrightarrow} y \quad \rho \in FT^*(y)}{a\rho \in FT^*(x)} \qquad \frac{x \stackrel{\tau}{\longrightarrow} y \quad \rho \in FT^*(y)}{\rho \in FT^*(x)}$$

$$\frac{x \stackrel{\alpha}{\not\longrightarrow} \text{ for all } \alpha \in X \cup \{\tau\} \quad \rho \in FT^*(x)}{X\rho \in FT^*(x)} \qquad \frac{x \stackrel{\alpha}{\not\longrightarrow} \text{ for all } \alpha \in X \cup \{\tau\} \quad x \stackrel{\mathrm{t}}{\longrightarrow} y \quad X\rho \in FT^*(y)}{X\rho \in FT^*(x)} \qquad \frac{x \stackrel{\alpha}{\not\longrightarrow} \text{ for all } \alpha \in X \cup \{\tau\} \quad x \stackrel{\mathrm{t}}{\longrightarrow} y \quad a\rho \in FT^*(y)}{Xa\rho \in FT^*(x)}\ (a \in X)$$

Congruence

Systems P, Q are *failure trace equivalent*, $P \equiv^*_{FT} Q$, if $FT^*(P) = FT^*(Q$

Theorem: $\equiv^*_{FT}$ is a congruence for the operators of $\mathrm{CCSP_t}$, except $+$.

A *rooted* version of $\equiv^*_{FT}$ is a congruence for all of $\mathrm{CCSP_t}$.

Congruence

Systems P, Q are *failure trace equivalent*, $P \equiv^*_{FT} Q$, if $FT^*(P) = FT^*(Q$

Theorem: $\equiv^*_{FT}$ is a congruence for the operators of $\mathrm{CCSP_t}$, except $+$.

A *rooted* version of $\equiv^*_{FT}$ is a congruence for all of $\mathrm{CCSP_t}$.

Write $P \sqsubseteq^*_{FT} Q$ iff $FT^*(P) \supseteq FT^*(Q)$.

The coarsest preorder respecting safety properties

Assume that the alphabet A of visible actions contains one specific action b, whose occurrence is *bad*.
The *canonical safety property* says that b **will never happen**.

A process P satisfies this property, notation $P \models \mathit{safety}(b)$, if no partial failure trace of P contains the action b.

A preorder $\sqsubseteq$ *respects* the canonical safety property if $P \sqsubseteq Q$ and $P \models \mathit{safety}(b)$ implies $Q \models \mathit{safety}(b)$.

Theorem: $\sqsubseteq^*_{FT}$ is the coarsest preorder that respects the canonical safety property.

In other words, if $P \not\sqsubseteq^*_{FT} Q$, then there is a context $\mathcal{C}[\_]$ such that $\mathcal{C}[P] \models \mathit{safety}(b)$, yet $\mathcal{C}[Q] \not\models \mathit{safety}(b)$.

May testing

Let $\omega \notin A$ be a special action, that does not occur in ordinary processes, but may be used in *testing contexts* $\mathcal{C}[\_]$.
Occurrence of ω denotes a successful test run.
We say that $\mathcal{C}[P]$ **may succeed** if it has a trace containing ω.

The *may-testing preorder* is defined by $P \sqsubseteq_{may} Q$ if

$$\forall \mathcal{C}[\_]. \qquad \mathcal{C}[P] \textbf{ may succeed} \Rightarrow \mathcal{C}[Q] \textbf{ may succeed}$$

Theorem: $\sqsupseteq^*_{FT}$ equals $\sqsubseteq_{may}$.

In other words, if $P \not\sqsupseteq^*_{FT} Q$, then there is a context $\mathcal{C}[\_]$ such that $\mathcal{C}[P]$ **may succeed**, yet $\mathcal{C}[Q]$ **may not**.

Concluding remarks

I added a time-out action to standard untimed process algebra.

Failure trace equivalence is now the coarsest reasonable congruence:
the coarsest that satisfies the canonical safety property,
the coarsest that satisfies all safety properties,
the congruence closure of trace equivalence,
and the equivalence generated by may testing.

Future work includes

- proving a congruence result for recursion
- finding complete axiomatisations
- and extending the approach from partial to complete failure traces, so that liveness properties will be respected.