Process algebra - CCS [Milner '80]

- Actions $a, b, \ldots, \bar{a}, \bar{b}, \ldots$, silent action τ
- Constants $A, B, \ldots$
- Deadlock process 0
- Prefix $\alpha.P$
- Summation $P + Q$
- Composition $P \parallel Q$
- Restriction $P \setminus L$, where L is a set of labels
- Relabeling $P[f]$, where f is a relabeling function

Transitional semantics

Semantics of processes is given in terms of *labeled transition systems* (LTS).

These are basically labeled directed graphs, where

- vertices are process terms,
- edges between vertices are labeled by actions,
- if $P \stackrel{\mathrm{df}}{=} a.Q$ then there is an edge between P and Q labeled by a,
- the graph of a process P is determined by *structural rules* applied to its subprocesses P'.

Example: Consider these defining equations:

- 0 - deadlock process;
- $A \stackrel{\text{df}}{=} a.0 + b.0$ - sum;
- $B \stackrel{\text{df}}{=} b.0 \parallel \bar{b}.0$ - parallel composition;
- $C \stackrel{\text{df}}{=} a.0 + (b.0 \parallel \bar{b}.0) \setminus \{b\}$ - sum, parallel composition and restriction on actions b and $\bar{b}$;

$$\begin{tikzcd} A \arrow[dd, shift right, "a"'] \arrow[dd, shift left, "b"] \\ \\ 0 \end{tikzcd} \qquad \begin{tikzcd} & B \arrow[dl, "b"'] \arrow[dd, "\tau"] \arrow[dr, "\bar{b}"] & \\ \bar{b}.0 \arrow[dr, "\bar{b}"'] & & b.0 \arrow[dl, "b"] \\ & 0 & \end{tikzcd} \qquad \begin{tikzcd} C \arrow[dd, shift right, "a"'] \arrow[dd, shift left, "\tau"] \\ \\ 0 \end{tikzcd}$$

Strong bisimulation $\sim$ [Park '81]

$$
\begin{tikzcd}
 & P \arrow[dl, "a"'] \arrow[dr, "a"] & & \sim & Q \arrow[d, "a"'] & \not\sim & R \arrow[dd, bend right, "a"'] \arrow[dr, "a"] & \\
P' \arrow[dr, "b"'] & & P'' \arrow[dl, "b"] & & Q' \arrow[d, "b"'] & \not\sim & & R' \arrow[dl, "b"] \\
 & 0 & & & 0 & & 0 &
\end{tikzcd}
$$

Weak bisimulation $\approx$

Composite transition $\quad \stackrel{a}{\Longrightarrow} \stackrel{\mathrm{df}}{=} \stackrel{\tau}{\longrightarrow}^{*} \stackrel{a}{\longrightarrow} \stackrel{\tau}{\longrightarrow}^{*}; \qquad \stackrel{\tau}{\Longrightarrow} \stackrel{\mathrm{df}}{=} \stackrel{\tau}{\longrightarrow}^{*};$

$$\begin{tikzcd} P \arrow[dr, "\tau"] \arrow[dd, bend right, "a"'] & \\ & P' \arrow[dl, "a"] \\ 0 & \end{tikzcd} \qquad \approx \qquad \begin{tikzcd} Q \arrow[d, "a"] \\ 0 \end{tikzcd}$$

Timed algebra - ACSR

[Bremond-Gregoire, Lee '93]

Extention of CCS which features:

- (discrete) time
- value passing
- resource requirements
- dynamic priorities
- parametrised processes

Instantaneous events:

$e ::= (\tau, ve)|(l?, ve)|(l!, ve)|(l?x, ve)|(l!ve_1, ve)$ - pairs (label, priority);

example: $(l?x, 3)$

Timed actions:

- $\{(r_1, \pi_1), \ldots, (r_k, \pi_k)\}$ – a list of pairs of resources r_i and their priorities π_i;
- every timed action takes one unit of time;
- the empty action $\emptyset$ represents idling;

example: $\{(cpu, y+1)\}$;

ACSR operators

- Deadlock process NIL
- Boolean condition $be \rightarrow P$
- Choice $P + Q$
- Parallel composition $P \parallel Q$
- Resource closure $[P]_I$
- Restriction $P \setminus F$
- Resource hiding $P \setminus\!\!\setminus I$
- Constants with parameters $C(\overrightarrow{x})$

Example: A periodic task on processor cpu with execution time c and relative deadline d:

$$
\begin{aligned}
\mathsf{Job} &\stackrel{\mathrm{df}}{=} (start?, 1).\mathsf{Exec}(0) \\
\mathsf{Exec}(s) &\stackrel{\mathrm{df}}{=} (s < c) \longrightarrow \{(cpu, \pi)\}\colon\ \mathsf{Exec}(s+1) + \\
&\qquad (s = c) \longrightarrow \mathsf{Wait} \\
\mathsf{Wait} &\stackrel{\mathrm{df}}{=} \emptyset\text{:}\mathsf{Wait} + (end?, 1).\mathsf{Job}
\end{aligned}
$$

$$
\mathsf{Act} \stackrel{\mathrm{df}}{=} (start!, 1).\emptyset^d : (end!, 2).\mathsf{Act}
$$

A process that models periodic execution of the task above:

$\mathsf{Task} \stackrel{\mathrm{df}}{=} (\mathsf{Job} \parallel \mathsf{Act}\ )\backslash\{start, end\}$

$$\frac{P \xrightarrow{\alpha} P'}{P+Q \xrightarrow{\alpha} P'} \qquad \frac{Q \xrightarrow{\alpha} Q'}{P+Q \xrightarrow{\alpha} P'} \qquad \frac{P \xrightarrow{\alpha} P', be=tt}{(be \longrightarrow P) \xrightarrow{\alpha} P'}$$

$$\frac{P \xrightarrow{e} P'}{P \| Q \xrightarrow{e} P' \| Q} \qquad \frac{Q \xrightarrow{e} Q'}{P \| Q \xrightarrow{e} P \| Q'} \qquad \frac{P \xrightarrow{(l!k,m)} P', Q \xrightarrow{(l?k,n)} Q'}{P \| Q \xrightarrow{(\tau,m+n)} P' \| Q'}$$

$$\frac{P \xrightarrow{A_1} P', Q \xrightarrow{A_2} Q', \rho(A_1) \cup \rho(A_2) = \emptyset}{P \| Q \xrightarrow{A_1 \cap A_2} P' \| Q'} \qquad \frac{P \xrightarrow{A_1} P_2, A_2 = \{(r,0) | r \in I - \rho(A_1)}{[P]_I \xrightarrow{A_1 \cap A_2} [P']_I}$$

$$\frac{P \xrightarrow{e} P'}{[P]_I \xrightarrow{e} [P']_I} \qquad \frac{P \xrightarrow{A} P'}{P \backslash F \xrightarrow{A} P' \backslash F} \qquad \frac{P \xrightarrow{e} P', e, \bar{e} \notin F}{P \backslash F \xrightarrow{e} P' \backslash F}$$

Preemption relation

An action α is *preempted* by β if

1. α and β are events with same label such that $\pi(\alpha) < \pi(\beta)$
   $[(l?, 1) < (l?, 2)]$
2. α is timed and β has label τ and priority $\pi(\beta) > 0$
   $[\{(r, 2)\} < (\tau, 1)]$
3. α and β are timed ations such that resources of β appear in α with smaller positive priority, and there is a resource r in β that appears in α with strictly smaller priority
   $[\{(r_1, 2), (r_2, 3)\} < \{(r_1, 2), (r_2, 5)$, but
   $\{(r_1, 2), (r_2, 3)\} \not< \{(r_1, 5)\}]$

Prioritized transition system

Prioritized transition system L_π is obtained from unprioritized LTS L by application of this rule: $P \xrightarrow{\alpha}_\pi P'$ is in L_π if $P \xrightarrow{\alpha} P'$ is in L and there is no transition $P \xrightarrow{\beta} P''$ in L with $\pi(\alpha) < \pi(\beta)$.

Example:

$$
\begin{tikzcd}
 & P \arrow[dl, "{(a!,1)}"'] \arrow[dd, "{(a!,2)}"] \arrow[dr, "{\{(r,3)\}}"] & & \rightsquigarrow & & P \arrow[dd, "{(a!,2)}"'] \arrow[dr, "{\{(r,3)\}}"] & \\
P_1 & & P_3 & & & & P_3 \\
 & P_2 & & & & P_2 &
\end{tikzcd}
$$

Earliest Deadline First (EDF)

Is a dynamic-priority scheduling algorithm, where a job with an early deadline has a higher priority than jobs with later deadlines.

Example: System EDFSys $=\mathrm{T}_1 \parallel \ldots \parallel \mathrm{T}_n$, where T_i are preemptable periodic tasks scheduled by EDF, with

- computation time c_i,
- relative deadlines d_i,
- dynamic priorities $\pi_i = d_{max} - (d_i - t)$,

where d_{max} is $1 + \max\{d_1 \ldots d_n\}$, and t represents time elapsed since the task started executing.

$$
\begin{array}{rcl}
\mathrm{T}_i & \stackrel{\mathrm{df}}{=} & (\mathsf{Job}_i \parallel \mathsf{Act}_i\ )\backslash\{start, end\} \\
\\
\mathsf{Job}_i & \stackrel{\mathrm{df}}{=} & (start?, 1).\mathsf{Exec}_i(0{,}0) \\
\mathsf{Exec}_i\ (s,t) & \stackrel{\mathrm{df}}{=} & (s < c_i) \longrightarrow \{(cpu, \pi_i)\}\colon\ \mathsf{Exec}_i\ (s+1, t+1) \\
 & & \qquad\qquad +\emptyset\ :\mathsf{Exec}_i(s, t+1) \\
 & & +(s = c_i) \longrightarrow \mathsf{Wait}_i \\
\mathsf{Wait}_i & \stackrel{\mathrm{df}}{=} & \emptyset{:}\mathsf{Wait}_i + (end?, 1).\mathsf{Job}_i \\
\\
\mathsf{Act}_i & \stackrel{\mathrm{df}}{=} & (start!, 1).\emptyset^{d_i} : (end!, 2).\mathsf{Act}_i
\end{array}
$$

Before we perform analysis, we establish a few desirable properties of the EDF specification.

Guardedness: In the specification of EDFSys, every recurcive constant is *gurded*, i.e. appears prefixed by a timed action of event.

Finite branching: EDFSys is a finite branching process.

Non-Zenoness: EDFSys determines a process where time always progresses.

Finiteness: The labeled transition system of EDFSys is finite.

Analysis of scheduled systems

- validation of specification: shows that a given specification correctly models the required scheduling discipline;
- schedulability analysis: checks whether or not any of the processes misses its deadline.

Analysis is based on *equivalence testing*.

Schedulability analysis

EDFSys $= \mathrm{T}_1 \parallel \ldots \parallel \mathrm{T}_n$ is schedulable iff

- none of the tasks T_i misses its deadline iff

- each job Job_i will be activated infinitely often iff

- each task T_i will execute only infinite sequences iff

- the whole system will not contain a finite computation.

If we restrict all observable actions (i.e. we will only "see" timed action $\emptyset$ - time progress) then this can be expressed as

$$\mathrm{EDFSys} \setminus\!\!\setminus \{cpu\} \;\approx_\pi\; \emptyset^\infty$$

For finite systems, the equivalence can be verified.