MoDeST Modelling of Hard an Softly Timed Systems

Holger Hermanns

Dependable Systems and Software, Saarland University

E-mail: `hermanns@cs.uni-sb.de`

joint work with Henrik Bohnenkamp, Pedro R. D'Argenio, Joost-Pieter Katoen

November 29, 2006

Background and motivation.

Listened to the talk by Henrik Bohnenkamp on Monday?

Issues of concern.

Important rationales behind the development of MODEST have been:

- *Orthogonality.*
- *Usability.*
- *Practical considerations.*
- *Expressiveness.*

(1) *Action nondeterminism*
(2) *Probabilistic branching*
(3) *Clocks*
(4) *Delay nondeterminism*
(5) *Random variables*

MODEST stands for Modeling and Description language for Stochastic Timed systems.

Stochastic Timed Automata

A stochastic timed automaton (STA) consists of control states, called *locations*, that are connected by edges.

Making things easy for a moment, edges are of the form $\xrightarrow{a,g,d,A}$ with

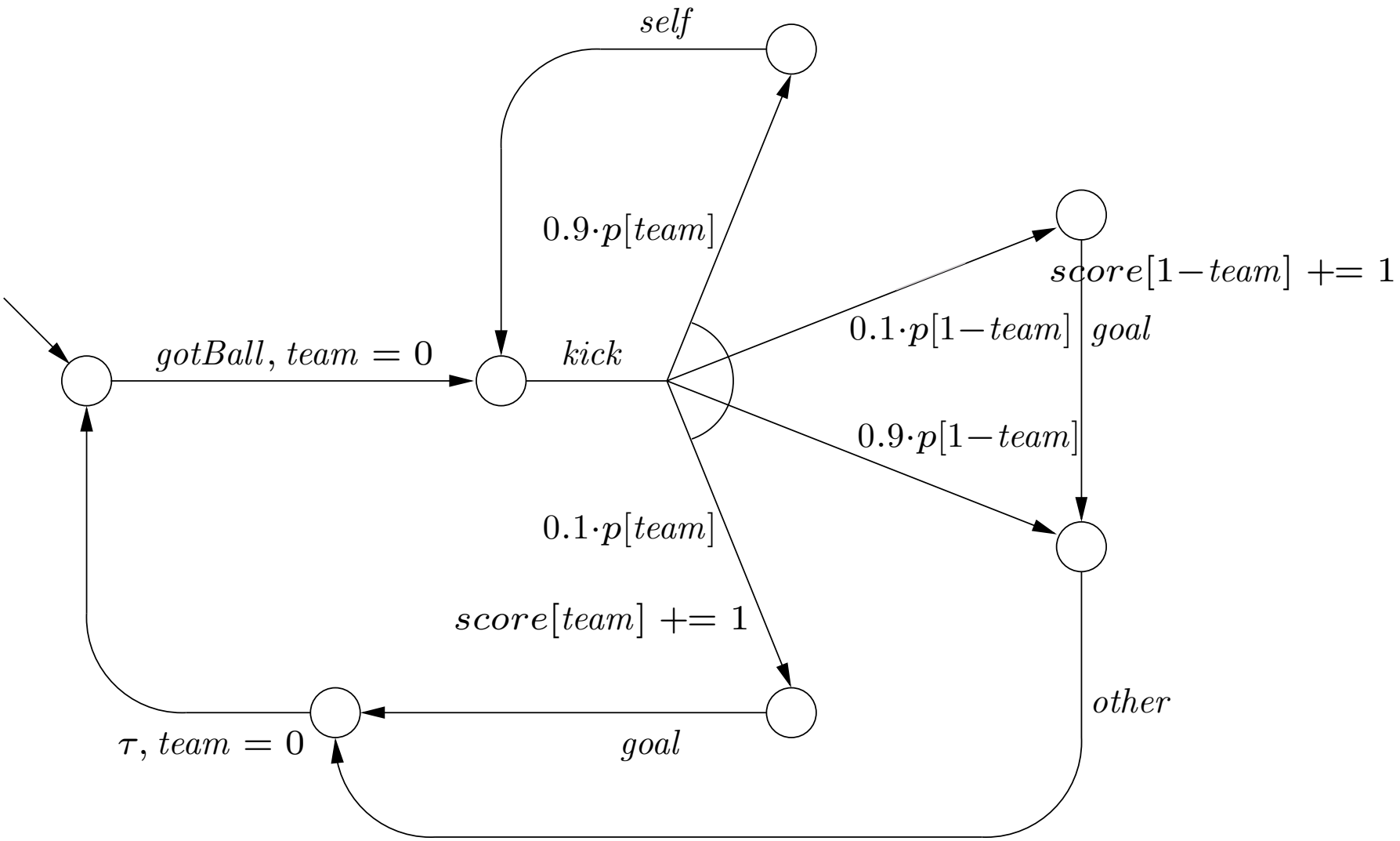
(i) an action a to be performed,

(ii) a guard g specifying when the edge is enabled,

iii) an urgency constraint d specifying when the edge ultimately should be executed (if at all), and

iv) a set A of assignments to be carried out atomically.

Stochastic timed automaton example



Empty assignments, **tt**rue guards and **ff**alse urgency constraints are omitted from edges.

STA, truly

The target of an edge is not just a control state, but rather a probability distribution over pairs $\langle A, s \rangle$ of assignments and control states.

Definition

A *stochastic timed automaton* (*STA*, for short) is a triple $(\mathrm{Loc}, \mathrm{Act}, \longrightarrow)$, where Loc is a set of *locations* and $\longrightarrow \subseteq \mathrm{Loc} \times (\mathrm{Act} \times \mathrm{Bxp} \times \mathrm{Bxp}) \times \mathrm{Wxp}$ is the edge relation.

Here, Wxp is a set of weight expressions. Intuitively this is a symbolic probability distributions, over pairs of assignments and (target) control states.

On taking the edge $s \xrightarrow{a,g,d} \mathcal{W}$, the system moves to control state s' assigning values according to assignment A' with a probability that is determined by $\mathcal{W}(\langle A', s' \rangle)$.

Randomness, symbolically

Randomness enters in two ways into this symbolic picture.

- Discrete randomness, by weight expressions:

  A weight expression $\mathcal{W}$ is best viewed as a mapping from an assignment and a control state onto an arithmetic expression.

- Continuous or discrete randomness, by sampling expressions:

- A *sampling expression* is of the form sample(F), with the intended meaning that it samples a value for distinguished variable $\xi \notin$ Var according to distribution F.

  F is a possbily parametric function on (the domain of) ξ such that for every instantiation of variables in Var, F is a distribution function on ξ.

We will return to this much later.

Syntax of Modest

A process behaviour P is constructed according to the following grammar:

$$
\begin{array}{rl}
P ::= & \mathsf{stop} \mid \mathsf{abort} \mid \mathsf{break} \mid act \\
& P_1;\ P_2 \\
& \mathsf{alt}\{::P_1\ \ldots\ ::P_k\} \\
& \mathsf{do}\{::P_1\ \ldots\ ::P_k\} \\
& \mathsf{par}\{::P_1\ \ldots\ ::P_k\} \\
& \mathsf{when}(b)\ P \mid \mathsf{urgent}(b)\ P \\
& act\ \mathsf{palt}\ \{:w_1{:}asgn_1;\ P_1\ \ldots\ :w_k{:}asgn_k;\ P_k\} \\
& \mathit{ProcName}(e_1, \ldots, e_k) \\
& \mathsf{throw}(excp) \\
& \mathsf{try}\{P\}\ \mathsf{catch}\ excp_1\ \{P_1\}\ \ldots\ \mathsf{catch}\ excp_k\ \{P_k\} \\
& \mathsf{relabel}\ \{I\}\ \mathsf{by}\ \{G\}\ P \mid \mathsf{extend}\ \{H\}\ P
\end{array}
$$

Operational Semantics

The operational semantics of behaviour P maps on a stochastic timed automaton $(\mathsf{Loc}, \mathsf{Act}, \longrightarrow)$ where

- Act is the set of actions occurring in P,
- Loc is the set of behaviors that are derivable from P using the edge relation $\longrightarrow$, and
- $\longrightarrow$ is the smallest relation that is defined by the inference rules below.

We use $\mathcal{D}(A, s)$ to denote the *deterministic* weight expression which chooses the assignments A and target location s with probability 1: $\mathcal{D}(A, s) = 1$.

Action, Choice and Sequential Composition (First Approximation)

$$act \xrightarrow{act,\mathbf{tt},\mathbf{ff},\varnothing} \surd$$

$$\frac{P_i \xrightarrow{a,g,d,A} P' \qquad (0 < i \leqslant k)}{\mathsf{alt}\{::P_1 \ \ldots \ ::P_k\} \xrightarrow{a,g,d,A} P'}$$

$$\frac{P \xrightarrow{a,g,d,A} P'}{P;\, Q \xrightarrow{a,g,d,A} \mathbf{M}_{;}(P')}$$

where

$$\mathbf{M}_{;}(P') \stackrel{\text{def}}{=} \begin{cases} P';\, Q & \text{if } P' \neq \surd \\ Q & \text{if} P' = \surd \end{cases}$$

Action, Choice and Sequential Composition, truly

$$act \xrightarrow{act,\mathbf{tt},\mathbf{ff}} \mathcal{D}(\varnothing, \surd)$$

$$\frac{P_i \xrightarrow{a,g,d} \mathcal{W}_i \qquad (0 < i \leqslant k)}{\mathsf{alt}\{::P_1 \ \ldots \ ::P_k\} \xrightarrow{a,g,d} \mathcal{W}_i}$$

$$\frac{P \xrightarrow{a,g,d} \mathcal{W}}{P;\, Q \xrightarrow{a,g,d} \mathcal{W} \circ \mathbf{M}_{;}^{-1}}$$

where

$$\mathbf{M}_{;}(A, P') \stackrel{\text{def}}{=} \begin{cases} \langle A, P';\, Q\rangle & \text{if } P' \neq \surd \\ \langle A \cup \mathbf{A}(Q), Q\rangle & \text{if } P' = \surd \end{cases}$$

Assignement collection function.

$\mathbf{A}(P) = \varnothing$	if P has one of the following forms: act, act palt $\{:w_1:asgn_1;\ P_1\ \ldots\ :w_k:asgn_k;\ P_k\}$, stop, abort, throw$(excp)$, break
$\mathbf{A}(P) = \mathbf{A}(Q)$	if P has one of the following forms: $Q;\ Q'$, when$(b)\ Q$, urgent$(b)\ Q$, try$\{Q\}$ catch $excp_1\ \{P_1\}\ \ldots$ catch $excp_k\ \{P_k\}$, relabel $\{I\}$ by $\{G\}\ Q$, extend $\{H\}\ Q$
$\mathbf{A}(P) = \bigcup_{i=1}^{k} \mathbf{A}(P_i)$	if P has one of the following forms: alt$\{::P_1\ \ldots\ ::P_k\}$, do$\{::P_1\ \ldots\ ::P_k\}$, par$\{::P_1\ \ldots\ ::P_k\}$
$\mathbf{A}(\mathit{ProcName}(e_1, \ldots, e_k)) = \{x_1 = e_1, \ldots, x_k = e_k\} \cup \mathbf{A}(Q)$	if process $\mathit{ProcName}(x_1, \ldots, x_k)\{Q\}$

Process instantiation.

$$\frac{P \xrightarrow{a,g,d} \mathcal{W}}{\mathit{ProcName}(e_1, \ldots, e_k) \xrightarrow{a,g,d} \mathcal{W}}$$

if process $\mathit{ProcName}(x_1, \ldots, x_k)\{P\}$.

Call by value, but you don't see it here.

Conditions.

$$\frac{P \xrightarrow{a,g,d} \mathcal{W}}{\mathsf{when}(b)\ P \xrightarrow{a,b\wedge g,d} \mathcal{W}}$$

$$\frac{P \xrightarrow{a,g,d} \mathcal{W}}{\mathsf{urgent}(b)\ P \xrightarrow{a,g,b\vee d} \mathcal{W}}$$

Loop (First approximation)

$$\mathsf{do}\{::P_1 \ldots ::P_k\} \stackrel{\text{def}}{=} \mathsf{auxdo}\{\mathsf{alt}\{::P_1 \ldots ::P_k\}\}\{\mathsf{alt}\{::P_1 \ldots ::P_k\}\}$$

$$\frac{P \xrightarrow{\flat,g,d,A} P'}{\mathsf{auxdo}\{P\}\{Q\} \xrightarrow{\tau,g,d,A} \surd}$$

$$\frac{P \xrightarrow{a,g,d,A} P' \qquad (a \neq \flat)}{\mathsf{auxdo}\{P\}\{Q\} \xrightarrow{a,g,d,A} \mathbf{M}_{\mathsf{do}}(P')}$$

where

$$\mathbf{M}_{\mathsf{do}}(P') \stackrel{\text{def}}{=} \begin{cases} \mathsf{auxdo}\{P'\}\{Q\} & \text{if } P' \neq \surd \\ \mathsf{auxdo}\{Q\}\{Q\} & \text{if } P' = \surd \end{cases}$$

$$\mathsf{break} \xrightarrow{\flat,\mathbf{tt},\mathbf{ff},\varnothing} \surd$$

Loop, truly

$$\mathsf{do}\{::P_1 \ldots ::P_k\} \stackrel{\text{def}}{=} \mathsf{auxdo}\{\mathsf{alt}\{::P_1 \ldots ::P_k\}\}\{\mathsf{alt}\{::P_1 \ldots ::P_k\}\}$$

$$\frac{P \xrightarrow{\natural,g,d} \mathcal{W}}{\mathsf{auxdo}\{P\}\{Q\} \xrightarrow{\tau,g,d} \mathcal{D}(\varnothing, \surd)}$$

$$\frac{P \xrightarrow{a,g,d} \mathcal{W} \qquad (a \neq \natural)}{\mathsf{auxdo}\{P\}\{Q\} \xrightarrow{a,g,d} \mathcal{W} \circ \mathbf{M}_{\mathsf{do}}^{-1}}$$

where

$$\mathbf{M}_{\mathsf{do}}(A, P') \stackrel{\text{def}}{=} \begin{cases} \langle A, \mathsf{auxdo}\{P'\}\{Q\}\rangle & \text{if } P' \neq \surd \\ \langle A, \mathsf{auxdo}\{Q\}\{Q\}\rangle & \text{if } P' = \surd \end{cases}$$

$$\mathsf{break} \xrightarrow{\natural,\mathbf{tt},\mathbf{ff}} \mathcal{D}(\varnothing, \surd)$$

Relabelling.

$$\frac{P \xrightarrow{a,g,d} \mathcal{W} \qquad f = [a_1/a'_1, \ldots, a_k/a'_k]}{\text{relabel } \{a_1, \ldots, a_k\} \text{ by } \{a'_1, \ldots, a'_k\}\ P \xrightarrow{f(a),g,d} \mathcal{W} \circ \mathbf{M}_{\text{rel}}^{-1}}$$

where

$$\mathbf{M}_{\text{rel}}(A, P') \stackrel{\text{def}}{=} \begin{cases} \langle A, \text{relabel } \{a_1, \ldots, a_k\} \text{ by } \{a'_1, \ldots, a'_k\}\ P' \rangle & \text{if } P' \neq \surd \\ \langle A, \surd \rangle & \text{if } P' = \surd \end{cases}$$

Alphabet extension.

$$\frac{P \xrightarrow{a,g,d} \mathcal{W}}{\mathsf{extend}\,\{act_1, \ldots, act_k\}\ P \xrightarrow{a,g,d} \mathcal{W} \circ \mathbf{M}_{\mathsf{ext}}^{-1}}$$

where

$$\mathbf{M}_{\mathsf{ext}}(A, P') \stackrel{\text{def}}{=} \begin{cases} \langle A, \mathsf{extend}\,\{act_1, \ldots, act_k\}\ P' \rangle & \text{if } P' \neq \surd \\ \langle A, \surd \rangle & \text{if } P' = \surd \end{cases}$$

Exception handling.

$$\mathsf{throw}(excp) \xrightarrow{excp,\mathbf{tt},\mathbf{ff}} \mathcal{D}(\varnothing, \mathsf{abort})$$

$$\mathsf{abort} \xrightarrow{\perp,\mathbf{tt},\mathbf{ff}} \mathcal{D}(\varnothing, \mathsf{abort})$$

$$\frac{P \xrightarrow{a,g,d} \mathcal{W} \qquad (a \notin \{excp_1, \ldots, excp_k\})}{\mathsf{try}\{P\}\ \mathsf{catch}\ excp_1\ \{P_1\}\ \ldots\ \mathsf{catch}\ excp_k\ \{P_k\} \xrightarrow{a,g,d} \mathcal{W} \circ \mathbf{M}_{\mathsf{try}}^{-1}}$$

where

$$\mathbf{M}_{\mathsf{try}}(A, P') \stackrel{\text{def}}{=} \begin{cases} \langle A, \mathsf{try}\{P'\}\ \mathsf{catch}\ excp_1\ \{P_1\}\ \ldots\ \mathsf{catch}\ excp_k\ \{P_k\}\rangle & \text{if } P' \neq \surd \\ \langle A, \surd \rangle & \text{if } P' = \surd \end{cases}$$

$$\frac{P \xrightarrow{excp_i,g,d} \mathcal{W} \qquad (0 < i \leqslant k)}{\mathsf{try}\{P\}\ \mathsf{catch}\ excp_1\ \{P_1\}\ \ldots\ \mathsf{catch}\ excp_k\ \{P_k\} \xrightarrow{\tau,g,d} \mathcal{D}(\mathbf{A}(P_i), P_i)}$$

Probabilistic prefix.

Let predicates $neg \equiv \bigwedge_{i=1}^{k} w_i < 0$ and $zero \equiv \sum_{i=1}^{k} w_i = 0$.

$$act \text{ palt } \{:w_1:asgn_1;\ P_1\ \ldots\ :w_k:asgn_k;\ P_k\} \xrightarrow{act,\neg(neg \vee zero),\mathbf{ff}} \mathcal{W}$$

with $\mathcal{W}$ being the weight expression:

$$\mathcal{W}(asgn_i \cup \mathbf{A}(P_i), P_i) \stackrel{\text{def}}{=} \sum_{j=1}^{k} \mathbf{I}(i,j) \cdot w_j$$

where

$$\mathbf{I}(i,j) \stackrel{\text{def}}{=} \textbf{if } (asgn_i \cup \mathbf{A}(P_i) = asgn_j \cup \mathbf{A}(P_j) \wedge P_i = P_j) \textbf{ then } 1 \textbf{ else } 0.$$

The guard $\neg(neg \vee zero)$ ensures that the weights are legal. Otherwise:

$$act \text{ palt } \{:w_1:asgn_1;\ P_1\ \ldots\ :w_k:asgn_k;\ P_k\} \xrightarrow{neg\_weight,neg,\mathbf{ff}} \mathcal{D}(\varnothing, \mathsf{abort})$$

$$act \text{ palt } \{:w_1:asgn_1;\ P_1\ \ldots\ :w_k:asgn_k;\ P_k\} \xrightarrow{no\_weight,zero,\mathbf{ff}} \mathcal{D}(\varnothing, \mathsf{abort})$$

Alphabet of a Modest term

$\alpha(\mathsf{stop}) = \alpha(\mathsf{abort}) = \alpha(\mathsf{break}) = \alpha(\mathsf{throw}(excp)) = \varnothing$

$\alpha(act) = \{act\} - \{\tau\}$

$\alpha(act\ \mathsf{palt}\ \{:w_1:asgn_1;\ P_1\ \ldots\ :w_k:asgn_k;\ P_k\}) = \alpha(act) \cup \bigcup_{i=1}^{k} \alpha(P_i)$

$\alpha(\mathsf{when}(b)\ P) = \alpha(\mathsf{urgent}(b)\ P) = \alpha(P)$

$\alpha(\mathsf{alt}\{::P_1\ \ldots\ ::P_k\}) = \alpha(\mathsf{do}\{::P_1\ \ldots\ ::P_k\}) = \alpha(\mathsf{par}\{::P_1\ \ldots\ ::P_k\}) = \bigcup_{i=1}^{k} \alpha(P_i)$

$\alpha(P_1;\ P_2) = \alpha(P_1) \cup \alpha(P_2)$

$\alpha(\mathsf{try}\{P\}\ \mathsf{catch}\ excp_1\ \{P_1\}\ \ldots\ \mathsf{catch}\ excp_k\ \{P_k\}) = \alpha(P) \cup \bigcup_{i=1}^{k} \alpha(P_i)$

$\alpha(\mathsf{relabel}\ \{a_1, \ldots, a_k\}\ \mathsf{by}\ \{a'_1, \ldots, a'_k\}\ P) = \alpha(P)[a_1/a'_1, \ldots, a_k/a'_k] - \{\tau\}$

$\alpha(\mathsf{extend}\ \{act_1, \ldots, act_k\}\ P) = \alpha(P) \cup \{act_1, \ldots, act_k\}$

$\alpha(\mathit{ProcName}(e_1, \ldots, e_k)) = \alpha(P) \qquad \text{provided process } \mathit{ProcName}(x_1, \ldots, x_k)\ \{P\}$

Parallel composition.

$$\mathsf{par}\{::P_1 \ \ldots \ ::P_k\} \stackrel{\text{def}}{=} (\ldots((P_1 \,||_{B_1}\, P_2) \,||_{B_2}\, P_3)\ldots) \,||_{B_{k-1}}\, P_k$$

with $B_j = (\bigcup_{i=1}^{j} \alpha(P_i)) \cap \alpha(P_{j+1})$.

$$\frac{P_1 \xrightarrow{a,g,d} \mathcal{W} \qquad (a \notin B)}{P_1 \,||_B\, P_2 \xrightarrow{a,g,d} \mathcal{W} \circ \mathbf{M}^{-1}_{\mathsf{par}P_2}} \qquad \frac{P_2 \xrightarrow{a,g,d} \mathcal{W} \qquad (a \notin B)}{P_1 \,||_B\, P_2 \xrightarrow{a,g,d} \mathcal{W} \circ \mathbf{M}^{-1}_{\mathsf{par}P_1}}$$

with $\mathbf{M}_{\mathsf{par}P}(A, P') \stackrel{\text{def}}{=} \langle A, P' \,||_B\, P\rangle$, where $\surd \,||_B\, \surd = \surd$.

Patient and impatient synchronisation.

- Patience:

$$\frac{P_1 \xrightarrow{a,g_1,d_1} \mathcal{W}_1 \quad P_2 \xrightarrow{a,g_2,d_2} \mathcal{W}_2 \qquad (a \in B \cap \mathsf{PAct})}{P_1 \,||_B\, P_2 \xrightarrow{a,g_1 \wedge g_2,d_1 \wedge d_2} (\mathcal{W}_1 \times \mathcal{W}_2) \circ \mathbf{M}_{\mathsf{par}}^{-1}}$$

- Impatience:

$$\frac{P_1 \xrightarrow{a,g_1,d_1} \mathcal{W}_1 \quad P_2 \xrightarrow{a,g_2,d_2} \mathcal{W}_2 \qquad (a \in B \cap \mathsf{IAct})}{P_1 \,||_B\, P_2 \xrightarrow{a,g_1 \wedge g_2,d_1 \vee d_2} (\mathcal{W}_1 \times \mathcal{W}_2) \circ \mathbf{M}_{\mathsf{par}}^{-1}}$$

In both cases,

$$\mathbf{M}_{\mathsf{par}}(\langle A_1, P_1' \rangle, \langle A_2, P_2' \rangle) \stackrel{\text{def}}{=} \begin{cases} \textbf{if } A_1 \cup A_2 \text{ is not a function } \textbf{then} \\ \quad \langle \varnothing, \mathsf{throw}(\mathit{inconsistent}) \rangle \\ \textbf{else} \quad \langle A_1 \cup A_2, P_1' \,||_B\, P_2' \rangle \end{cases}$$

where, as before, $\surd \,||_B\, \surd = \surd$.

Concrete Semantics

Dense probabilistic transition systems.

Let $\mathit{Prob}(\Omega)$ denote the probability measure on some Borel measurable space Ω.

Definition
A *probabilistic transition system* (*PTS*, for short) is a triple $(\Sigma, \mathcal{L}, \longrightarrow)$ where Σ is a set of *states*, $\mathcal{L}$ is a set of *labels*, and $\longrightarrow \subseteq \Sigma \times \mathcal{L} \times \mathit{Prob}(\Sigma)$ is the *(probabilistic) transition relation*.

We write $\sigma \xrightarrow{\ell} \mathbf{P}$ whenever $\langle \sigma, \ell, \mathbf{P} \rangle \in \longrightarrow$.

Timed PTS

Definition
A *timed probabilistic transition system* is a PTS $(\Sigma, \mathcal{L}, \longrightarrow)$ such that:

- $\mathcal{L}$ is the disjoint union of a set Act of actions and the set $\mathbb{R}_{>0}$ of delays

- every transition labeled with $t \in \mathbb{R}_{>0}$ is has a single target state with probability 1 and satisfies (Wang's axioms):

  - time additivity: $\sigma \xrightarrow{t+t'} \sigma' \iff \sigma \xrightarrow{t} \sigma'' \xrightarrow{t'} \sigma'$ for some σ'', and
  - time determinism: $\sigma \xrightarrow{t} \sigma'$ and $\sigma \xrightarrow{t} \sigma''$ imply $\sigma' = \sigma''$.

Valuations and randomness.

- A *valuation* is a function that, to each variable in Var, assigns a value of its type. Let Val be the set of all valuations.

- Since we support a sampling expressions, we now have to say how they are evaluated concretely.

- Let $F[v] \stackrel{\text{def}}{=} \lambda\xi.\, v(F(\xi))$ denote the instantiation of the sampling expression F with valuation v. $F[v]$ is a distribution function on variable ξ.

- Valuations are extended to expressions in the usual way: $v(e)$, for expression $e \in$ Exp, is obtained by replacing each variable x in e by $v(x)$ and by replacing each sampling expression sample(F) by a unique random variable (name) X with distribution $F[v]$.

- Uniqueness means that each occurrence of sample(F) in expression e is replaced by a distinct random variable, and, hence, sampled with possibly different values.

Interpretation of a stochastic timed automaton.

A *state* in the behaviour of a STA is completely identified by the location in which the system is located and the value of all its variables.

Let $\Sigma_{\text{Loc}} \stackrel{\text{def}}{=} \text{Loc} \times \mathit{Val}$ be the set of states and $\mathcal{B}(\Sigma_{\text{Loc}})$ be the Borel algebra with sample space Σ_{Loc}. $\pi^v_{\mathcal{W}}$ denotes the discrete distribution function derived from the weight expression evaluated in v, i.e., $\pi^v_{\mathcal{W}}(\alpha) \stackrel{\text{def}}{=} \frac{v(\mathcal{W}(\alpha))}{\sum_\alpha v(\mathcal{W}(\alpha))}$ for all α.

Semantics of STA

Definition

The semantics of stochastic timed automaton $(\mathsf{Loc}, \mathsf{Act}, \longrightarrow)$ is the timed PTS $(\Sigma_{\mathsf{Loc}}, \mathsf{Act} \cup \mathbb{R}_{>0}, \longrightarrow)$ where $\longrightarrow$ is the smallest relation satisfying the following inference rules:

$$(1) \quad \frac{s \xrightarrow{a,g,d} \mathcal{W} \qquad v(g) \text{ holds}}{\langle s, v \rangle \xrightarrow{a} \mathbf{P}^v_{\mathcal{W}}}$$

where

$$\mathbf{P}^v_{\mathcal{W}}(B) \stackrel{\text{def}}{=} \sum_{s \in \mathsf{Loc}, A \in \mathsf{Asgn}} \pi^v_{\mathcal{W}}(\langle A, s \rangle) \cdot (\mathbf{P}^v_A \circ (F^v_{\langle A,s \rangle})^{-1})(B)$$

and $F^v_{\langle A,s \rangle} : (\mathsf{RVar}(v \circ A) \to \mathbb{R}) \to \Sigma_{\mathsf{Loc}}$ is defined by $F^v_{\langle A,s \rangle}(u) \stackrel{\text{def}}{=} \langle s, (u \circ v \circ A) \rangle$.

For the timed transitions we have:

$$(2) \quad \frac{\forall\, t' < t : (v + t')(\mathsf{tp}_s) \text{ holds}}{\langle s, v \rangle \xrightarrow{t} \langle s, v + t \rangle}$$

where $\mathsf{tp}_s \stackrel{\text{def}}{=} \neg \bigvee \{ d \mid s \xrightarrow{a,g,d} \mathcal{W} \}$ is the *time progress condition*, and $(v+t)(x) \stackrel{\text{def}}{=} v(x)+t$ if $x \in \mathsf{Ck}$ and $v(x)$ otherwise.

Some shorthand notations.

- Both alt- and do allow an else alternative (as in Promela).

$$\begin{array}{rl} & \mathsf{alt}\{::\mathsf{when}(b_1)\ P_1\ \ldots\ ::\mathsf{when}(b_k)\ P_k\ ::\mathsf{else}\ Q\} \\ \stackrel{\text{def}}{=} & \mathsf{alt}\{::\mathsf{when}(b_1)\ P_1\ \ldots\ ::\mathsf{when}(b_k)\ P_k\ ::\mathsf{when}(\neg \bigvee_{i=1}^{k} b_i)\ Q\}. \end{array}$$

- In a probabilistic alternative, either assignments or processes (but not both) can be omitted.

- Other useful standard programming constructs, can be defined, such as:

$$\mathsf{while}(b)\{P\} \stackrel{\text{def}}{=} \mathsf{do}\{::\mathsf{when}(b)\ P\ ::\mathsf{else\ break}\}.$$

- Hiding is a particular form of relabeling:

$$\mathsf{hide}\{act_1, \ldots, act_k\}\ P \stackrel{\text{def}}{=} \mathsf{relabel}\ \{act_1, \ldots, act_k\}\ \mathsf{by}\ \{\underbrace{\tau, \ldots, \tau}_{k\ \text{times}}\}\ P$$

Rationales

- Probabilistic branching.
- Clocks and distribution sampling.
- Patience and impatience.
- Invariants and urgency constraints.
- Data model and assignments.
- Synchronisation discipline and value passing.
- Exception handling and scoping.
- Priorities. (not included here, can be added)

Submodels of STA (and thus Modest)

	LTS	PTS	TA	PTA	DTMC	CTMC	CTMDP	GSMP	SA	STA
prob. branching	-	+	-	+	+	+	+	+	+	+
clocks	-	-	+	+	-	R	R	+	+	+
random variables	-	-	-	-	-	EXP	EXP	+	+	+
delay nondet.	-	-	+	+	-	-	-	-	-	+
action nondet.	+	+	+	+	-	-	+	-	+	+

Past and Present Applications

- Reliability analysis of the upcoming European Train Control System standard (within SFB AVACS)
- Analysis of an innovative plug-and-play comunication protocol (see Henrik's presentation)
- Optimisation of production schedules in combination with timed automata-based schedule synthesis (via UPPAAL).
- Evaluation of Energy Efficiency of ZigBee sensor networks protocol parameters.

References

[1] M. Abadi and A.D. Gordon. A calculus for cryptographic protocols: The spi calculus. *Inf. and Comp.*, **148**(1):1–70, 1999.

[2] R. Alur, C. Courcoubetis, N. Halbwachs, T.A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theor. Comput. Sci.*, **138**(1): 3–34, 1995.

[3] R. Alur and D.L. Dill. A theory of timed automata. *Th. Comp. Sc.*, **126**(2):183–235, 1994.

[4] C. Baier, F. Ciezinski, and M. Groesser. PROBMELA: a modeling language for communicating probabilistic processes. In: *ACM-IEEE Int. Conf. on Formal Methods and Models for Codesign*, ACM Press, 2004.

[5] G. Behrmann, A. David, and K.G. Larsen. A tutorial on UPPAAL. In: *Formal Methods for the Design of Real-Time Systems*, LNCS 3185: 200–237. Springer-Verlag, 2004. (see also `www.uppaal.com`).

[6] M. Ben-Ari. *Principles of Concurrent and Distributed Programming*. Prentice Hall, 1990.

[7] G. Berry. Preemption and concurrency. In: *Found. of Software Techn. and Th. Comp. Sc.*, LNCS 761: 72–93. Springer-Verlag, 1993.

[8] Henrik Bohnenkamp, Johan Gorter, Jarno Guidi, and Joost-Pieter Katoen. Are you still there? — A lightweight algorithm to monitor node presence in self-configuring networks. 2005. Submitted to IPDS 2005.

[9] H. Bohnenkamp, H. Hermanns, J.-P. Katoen and J. Klaren. The MODEST modelling tool and its implementation. In: *Modelling Techniques and Tools for Computer Perfermance Evaluation*, LNCS 2794: 116–133. Springer-Verlag, 2003.

[10] H. Bohnenkamp, H. Hermanns, J. Klaren, A. Mader and Y.S. Usenko. Synthesis and stochastic assessment of schedules for lacquer production. In: *Quantitative Evaluation of Systems*, IEEE CS Press, pp. 28–38, 2004.

[11] T. Bolognesi and E. Brinksma. Introduction to the formal description technique LOTOS. *Computer Networks*, **14**: 25–59, 1987.

[12] S. Bornot and J. Sifakis. An algebraic framework for urgency. *Inf. and Comp.*, **163**:172–202, 2001.

[13] V. Bos and J. J. T. Kleijn. *Formal Specification and Analysis of Industrial Systems*. PhD thesis, Eindhoven University of Technology, 2002.

[14] M. Bravetti and P.R. D'Argenio. Tutte le algebre insieme: Concepts, discussions and relations of stochastic process algebras with general distributions. In *Validation of Stochastic Systems*, LNCS 2925: 44–88. Springer-Verlag, 2004.

[15] M. Bravetti and R. Gorrieri. The theory of interactive generalised semi-Markov processes. *Th. Comp. Sc.*, **286**(1): 5–32, 2002.

[16] S. Cattani, R. Segala, M.Z. Kwiatkowska and G. Norman. Stochastic transition systems for continuous state spaces and non-determinism. In V. Sassone, editor, *Foundations of Software Science and Computation Structures (FOSSACS)*, LNCS, Springer-Verlag, 2005 (to appear).

[17] D. Cavin, Y. Sasson, and A. Schiper. On the accuracy of MANET simulators. In: *Principles of Mobile Computing*, pp. 38–43. ACM Press, 2002.

[18] G. Ciardo and R. Zijal. Well-defined stochastic Petri nets. In *Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, pp. 278–284. SCS Simulation Series, 1996.

[19] Special issue on embedded systems. *IEEE Computer*, **33**(9), 2000.

[20] D. Deavours, G. Clark, T. Courtney, D. Daly, S. Derasavi, J. Doyle, W.H. Sanders and P. Webster. The MÖBIUS framework and its implementation. *IEEE Tr. on Softw. Eng.*, **28**(10):956–970, 2002.

[21] D.D. Deavours, and W.H. Sanders. An efficient well-specified check. In *Petri Nets and Performance Models*, IEEE CS Press, 1999.

[22] P.R. D'Argenio. *Algebras and Automata for Timed and Stochastic Systems*. PhD thesis, Dept. of Computer Science, University of Twente, 1999.

[23] P.R. D'Argenio and E. Brinksma. A calculus for timed automata (extended abstract). In: *Formal Techniques in Real-Time and Fault Tolerant Systems*, LNCS 1135: 110-129. Springer-Verlag, 1996.

[24] P.R. D'Argenio, H. Hermanns and J.-P. Katoen. On generative parallel composition. *Electr. Notes on Th. Comp. Sc.*, **22**, 1999.

[25] P.R. D'Argenio, H. Hermanns, J.-P. Katoen and J. Klaren. MODEST: A modelling language for stochastic timed systems. In: *Process Algebra and Probabilistic Methods*, LNCS 2165: 87–104. Springer-Verlag, 2001.

[26] P.R. D'Argenio, J.-P. Katoen and E. Brinksma. An algebraic approach to the specification of stochastic systems (extended abstract). In: *Programming Concepts and Methods*, pp. 126–147, Chapman & Hall, 1998.

[27] P.R. D'Argenio, J.-P. Katoen and E. Brinksma. Specification and Analysis of Soft Real-Time Systems: Quantity and Quality. In: *Proc. of the 20th IEEE Real-Time Systems Symposium*, pp. 104–114. IEEE Society Press, 1999.

[28] Josée Desharnais. *Labeled Markov Process*. PhD thesis, McGill University, Montréal, 1999.

[29] S. Edwards, L. Lavagno, E.A. Lee and A. Sangiovanni-Vincentelli. Design of

embedded systems: formal models, validation and synthesis. *Proceedings of the IEEE*, **85**(3):366–390, 1997.

[30] E.A. Feinberg and A. Shwartz. *Handbook of Markov Decision Processes.* Kluwer, 2002.

[31] H. Garavel and M. Sighireanu. A graphical parallel composition operator for process algebras. In *Formal Methods for Protocol Engineering and Distributed Systems*, pp. 185–202, Kluwer, 1999.

[32] H. Garavel and M. Sighireanu. On the introduction of exceptions in E-LOTOS. In: *Formal Description Techniques*, pp. 469–484. Kluwer, 1996.

[33] P.W. Glynn. A GSMP formalism for discrete event simulation. *Proc. of the IEEE*, **77**(1):14–23, 1989.

[34] M.R. Hansen and H. Rischel. *Introduction to Programming using SML*. Addison-Wesley, 1999.

[35] T.A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model checking for real-time systems. *Inf. and Comp.*, **111**:193–244, 1994.

[36] Holger Hermanns, Ulrich Herzog, and Joost-Pieter Katoen. Process algebra for performance evaluation. *Theoretical Computer Science*, 274:43–87, 2002.

[37] H. Hermanns and D. Turetayev. A generalisation of the well-specified check. In: *Performability Modeling of Computer and Communication Systems*, pp. 62–66, 2003.

[38] Jane Hillston. *A Compositional Approach to Performance Modelling*. PhD thesis, University of Edinburgh, 1994.

[39] G.J. Holzmann. *The* SPIN *Model Checker*. Addison-Wesley, 2002.

[40] C. Hoare *Communicating Sequential Processes*. Prentice-Hall, 1985.

[41] ISO/IEC. *Information Technology - E-LOTOS*. ISO/IEC International Standard 15437, 2001.

[42] D.N. Jansen, H. Hermanns and Y.S. Usenko. From StoCharts to MoDeST: a comparative reliability analysis of train radio communications. Submitted to WOSP 2005.

[43] R.M. Keller. Formal verification of parallel programs. *Communication of the ACM*, **19**(7): 371–384, 1976.

[44] J. Kramer and J. McGee. *Concurrency: State Models and Java Programs*. John Wiley and Sons, 1999.

[45] V.G. Kulkarni. *Modeling and Analysis of Stochastic Systems*. Chapman & Hall, 1995.

[46] M.Z. Kwiatkowska, G. Norman, R. Segala and J. Sproston. Automatic verification of real-time systems with discrete probability distributions. *Theoretical Computer Science*, **282**: 101–150, 2002.

[47] K.G. Larsen and A. Skou. Bisimulation through probabilistic testing. *Inf. and Comp.*, **94**(1): 1–28, 1991.

[48] E.A. Lee. Embedded software. In: M. Zelkowitz, editor, *Advances in Computers*, vol. **56**, Academic Press, 2002.

[49] D. Luckham and W. Polak. ADA exception handling: an axiomatic approach. *ACM Trans. on Prog. Lang. and Sys.*, **2**(2): 225–233, 1980.

[50] N. Lynch and F.W. Vaandrager. Action transducers and timed automata. *Formal Aspects of Comput.*, **8**(5): 499–538, 1996.

[51] V. Mertsiotakis. *Approximate Analysis Methods for Stochastic Process Algebras*. PhD thesis, University of Erlangen-Nürnberg, 1998.

[52] R. Milner. *Communication and Concurrency*. Prentice-Hall, 1989.

[53] R. Milner. *Communicating and Mobile Systems: The π-Calculus*, Cambridge Univ. Press, 1999.

[54] R. Segala and N.A. Lynch. Probabilistic simulations for probabilistic processes. *Nordic J. of Computing*, **2**(2): 250–273, 1995.

[55] A.N. Shiryaev. *Probability*. Graduate texts in Mathematics, vol. 95, 1996.

[56] J. Sifakis. Personal communication, 2004.

[57] M. Sighireanu. LOTOS NT user's manual (version 2.4). Tech. Rep. INRIA Rhône-Alpes/VASY, 2004 (`ftp://ftp.inrialpes.fr/pub/vasy/traian`).

[58] A. Sokolova and E.P. de Vink Probabilistic automata: system types, parallel composition and comparison In *Validation of Stochastic Systems*, LNCS 2925: 1–43. Springer-Verlag, 2004.

[59] W. Yi, P. Pettersson and M. Daniels. Automatic verification of real-time communicating systems by constraint solving. In: *Formal Description Techniques*, pp. 223–238, North-Holland, 1994.

[60] W. Yi. Real-time behaviour of asynchronous agents. In: *Concurrency Theory*, LNCS 458: 502–520, 1990.