A Stochastic Logic for Mobility and Global Computing

R. De Nicola[2], J.-P. Katoen[1], Diego Latella[1], M. Loreti[2], M. Massink[1]

(1) Consiglio Nazionale delle Ricerche
Ist. di Scienza e Tecnologie dell'Informazione "A. Faedo"
Formal Methods && Tools Lab

(2) Univ. di Firenze
Dip. di Sist. e Inf.

PaCO
Lucca - June 25, 2009

Outline

Outline

1. STOKLAIM in one slide;

Outline

1. StoKlaim in one slide;
2. MoSL: General;

Outline

1. StoKlaim in one slide;
2. MoSL: General;
3. MoSL: Operators

Outline

1. StoKlaim in one slide;
2. MoSL: General;
3. MoSL: Operators (informal!);

Outline

1. StoKlaim in one slide;
2. MoSL: General;
3. MoSL: Operators (informal!);
4. Example properties (Airbag);

Outline

1. StoKlaim in one slide;
2. MoSL: General;
3. MoSL: Operators (informal!);
4. Example properties (Airbag);
5. Developments

KLAIM in one slide

KLAIM in one slide

- *System:*

KLAIM in one slide

- *System:*

  Collection of Sites (i.e. physical addresses) with

KLAIM in one slide

- *System:*

  Collection of Sites (i.e. physical addresses) with

  - Processes running

KLAIM in one slide

- *System:*

  Collection of Sites (i.e. physical addresses) with

  - Processes running
  - Tuples stored.

KLAIM in one slide

- *System:*

  Collection of Sites (i.e. physical addresses) with

  - Processes running
  - Tuples stored.

- *System State (Network snapshot):*

KLAIM in one slide

- *System:*

  Collection of Sites (i.e. physical addresses) with

  - Processes running
  - Tuples stored.

- *System State (Network snapshot):*

  Collection of nodes, each located at a specific site

KLAIM in one slide

- *System:*

  Collection of Sites (i.e. physical addresses) with

  - Processes running
  - Tuples stored.

- *System State (Network snapshot):*

  Collection of nodes, each located at a specific site

- Processes execute actions

KLAIM in one slide

- *System:*

  Collection of Sites (i.e. physical addresses) with

  - Processes running
  - Tuples stored.

- *System State (Network snapshot):*

  Collection of nodes, each located at a specific site

- Processes execute actions

  - acting on local or remote sites

KLAIM in one slide

- *System:*

  Collection of Sites (i.e. physical addresses) with

  - Processes running
  - Tuples stored.

- *System State (Network snapshot):*

  Collection of nodes, each located at a specific site

- Processes execute actions

  - acting on local or remote sites
  - using logical addresses (locally mapped to physical ones)

StoKlaim in one slide

- *System:*

  Collection of Sites (i.e. physical addresses) with

  - Processes running
  - Tuples stored.

- *System State (Network snapshot):*

  Collection of nodes, each located at a specific site

- Processes execute actions which take time (exp. dist. r.v.)
  - acting on local or remote sites
  - using logical addresses (locally mapped to physical ones)

StoKlaim in one (more) slide

StoKlaim in one (more) slide

- *Actions*

  $$A ::= \textbf{newloc}(!u) \;\Big|\; \textbf{out}(\vec{f})@\ell \;\Big|\; \textbf{in}(\vec{F})@\ell \;\Big|\; \textbf{read}(\vec{F})@\ell \;\Big|\; \textbf{eval}(P)@\ell$$

StoKlaim in one (more) slide

- *Actions*

  $$A ::= \mathbf{newloc}(!u) \;\Big|\; \mathbf{out}(\vec{f})@\ell \;\Big|\; \mathbf{in}(\vec{F})@\ell \;\Big|\; \mathbf{read}(\vec{F})@\ell \;\Big|\; \mathbf{eval}(P)@\ell$$

- *Processes*

  $$P ::= \mathbf{nil} \;\Big|\; (A, r).P \;\Big|\; P + P \;\Big|\; P \mid P \;\Big|\; X$$

StoKlaim in one (more) slide

- *Actions*

  $$A ::= \mathbf{newloc}(!u) \;\Big|\; \mathbf{out}(\vec{f})@\ell \;\Big|\; \mathbf{in}(\vec{F})@\ell \;\Big|\; \mathbf{read}(\vec{F})@\ell \;\Big|\; \mathbf{eval}(P)@\ell$$

- *Processes*

  $$P ::= \mathbf{nil} \;\Big|\; (A, {\color{red} r}).P \;\Big|\; P + P \;\Big|\; P \mid P \;\Big|\; X$$

- *Networks*

  $$N ::= \mathbf{0} \;\Big|\; i ::_\rho P \;\Big|\; i ::_\rho \langle \vec{f} \rangle \;\Big|\; N \parallel N$$

StoKlaim in one (more) slide

- *Actions*

  $A ::= \textbf{newloc}(!u) \;\Big|\; \textbf{out}(\vec{f})@\ell \;\Big|\; \textbf{in}(\vec{F})@\ell \;\Big|\; \textbf{read}(\vec{F})@\ell \;\Big|\; \textbf{eval}(P)@\ell$

- *Processes*

  $P ::= \textbf{nil} \;\Big|\; (A, r).P \;\Big|\; P + P \;\Big|\; P \mid P \;\Big|\; X$

- *Networks*

  $N ::= \mathbf{0} \;\Big|\; i ::_\rho P \;\Big|\; i ::_\rho \langle \vec{f} \rangle \;\Big|\; N \parallel N$

- *Stored Tuples*

  $\langle f_1, \ldots, f_n \rangle$ with $f_j ::= v \;\Big|\; P$

StoKlaim in one (more) slide

- *Actions*

  $$A ::= \mathbf{newloc}(!u) \;\Big|\; \mathbf{out}(\vec{f})@\ell \;\Big|\; \mathbf{in}(\vec{F})@\ell \;\Big|\; \mathbf{read}(\vec{F})@\ell \;\Big|\; \mathbf{eval}(P)@\ell$$

- *Processes*

  $$P ::= \mathbf{nil} \;\Big|\; (A, r).P \;\Big|\; P + P \;\Big|\; P \mid P \;\Big|\; X$$

- *Networks*

  $$N ::= \mathbf{0} \;\Big|\; i ::_\rho P \;\Big|\; i ::_\rho \langle \vec{f} \rangle \;\Big|\; N \parallel N$$

- *Stored Tuples*

  $\langle f_1, \ldots, f_n \rangle$ with $f_j ::= v \;\Big|\; P$

- *Patterns*

  $F_1, \ldots, F_n$ as usual ...

MoSL: General

1. a *temporal logic* (dynamic evolution);

1. a *temporal logic* (dynamic evolution);
2. both *action*- and *state*-based;

   actions may bind variables

MoSL: General

1. a *temporal logic* (dynamic evolution);
2. both *action*- and *state*-based;

   actions may bind variables

3. a *real-time* logic (real-time bounds);

MoSL: General

1. a *temporal logic* (dynamic evolution);
2. both *action*- and *state*-based;

   actions may bind variables

3. a *real-time* logic (real-time bounds);
4. a *probabilistic logic* (performance and dependability aspects);

MoSL: General

1. a *temporal logic* (dynamic evolution);
2. both *action*- and *state*-based;

   actions may bind variables

3. a *real-time* logic (real-time bounds);
4. a *probabilistic logic* (performance and dependability aspects);
5. a *spatial logic* (spatial structure of the network);

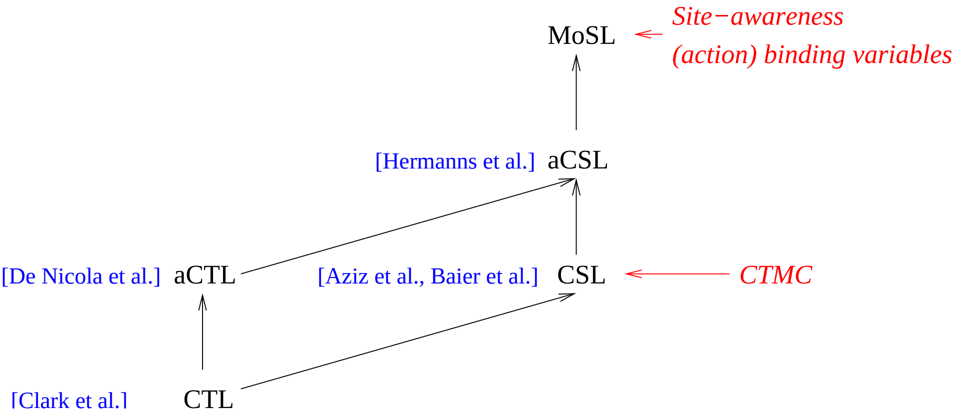
MoSL: General

1. a *temporal logic* (dynamic evolution);
2. both *action*- and *state*-based;

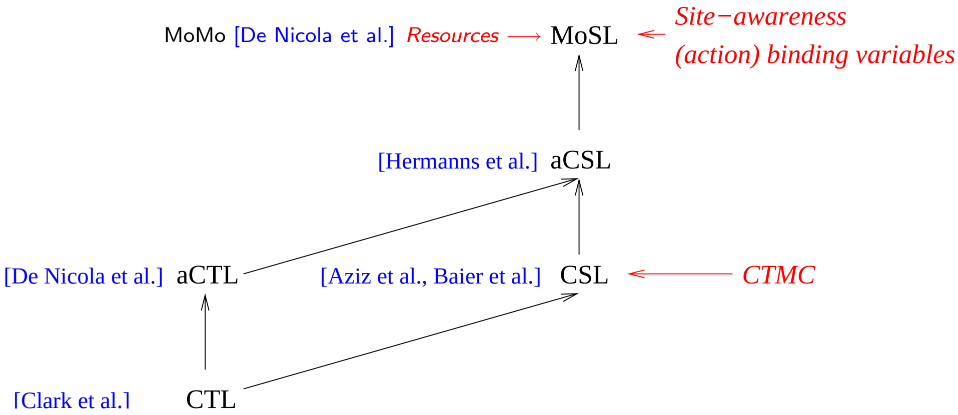
   actions may bind variables

3. a *real-time* logic (real-time bounds);
4. a *probabilistic logic* (performance and dependability aspects);
5. a *spatial logic* (spatial structure of the network);
6. a *resource-oriented logic* (GC open-endedness)

MoSL:General (cont'd)



MoSL:General (cont'd)



MoSL: Atomic propositions

$\aleph$

MoSL: Atomic propositions

$$\aleph \quad ::=$$

MoSL: Atomic propositions

$$\aleph \quad ::= \quad \mathsf{tt}$$

MoSL: Atomic propositions

$$\aleph \quad ::= \quad \mathsf{tt} \mid Q@\imath$$

MoSL: Atomic propositions

$$\aleph \quad ::= \quad \mathsf{tt} \mid Q@\imath \mid \langle \vec{F} \rangle @\imath \quad .$$

$$\aleph \quad ::= \quad \mathsf{tt} \mid Q@\imath \mid \langle\vec{F}\rangle@\imath \quad .$$

Boot@A (satisfied if process *Boot* is ready to start at *A*)

MoSL: Atomic propositions

$$\aleph \quad ::= \quad \mathsf{tt} \mid Q@\imath \mid \langle \vec{F} \rangle @\imath \quad .$$

Boot@*A* (satisfied if process *Boot* is ready to start at *A*)

$\langle GO \rangle @A$ (satisfied if value *GO* is stored at *A*)

MoSL: Action specifiers and action sets

MoSL: Action specifiers and action sets

In CTL

$$\Phi \;\; \mathcal{U} \, \Psi$$

In aCTL

$$\Phi \;{}_{\Delta}\mathcal{U}_{\Omega}\; \Psi$$

Δ, Ω: Sets of actions (uninterpreted, atomic)

In MoSL

$$\Phi \; {}_{\Delta}\mathcal{U}_{\Omega} \; \Psi$$

Δ, Ω: Sets of *action specifiers*, to be matched against KLAIM actions

$$init : \mathbf{o}(GO, A)$$

Satisfied by any action executed at site *init*, by means of which a process uploads value *GO* to site *A*;

MoSL: Action specifiers and action sets (cont'd)

$$init : \mathbf{o}(GO, A)$$

Satisfied by any action executed at site *init*, by means of which a process uploads value *GO* to site *A*;

$$!z_1 : \mathbf{o}(GO, !z_2)$$

Satisfied by any action, executed at some site (z_1), by means of which a process uploads value *GO* to some site (z_2);

MoSL: Action specifiers and action sets (cont'd)

$$init : \mathbf{o}(GO, A)$$

Satisfied by any action executed at site *init*, by means of which a process uploads value *GO* to site *A*;

$$!z_1 : \mathbf{o}(GO, !z_2)$$

Satisfied by any action, executed at some site (z_1), by means of which a process uploads value *GO* to some site (z_2);

$\Delta = \{\xi_1, \ldots \xi_n\}$ satisfied by any action satisfying any of ξ_j;

MoSL: Action specifiers and action sets (cont'd)

$$init : \mathbf{o}(GO, A)$$

Satisfied by any action executed at site *init*, by means of which a process uploads value *GO* to site *A*;

$$!z_1 : \mathbf{o}(GO, !z_2)$$

Satisfied by any action, executed at some site (z_1), by means of which a process uploads value *GO* to some site (z_2);

$\Delta = \{\xi_1, \ldots \xi_n\}$ satisfied by any action satisfying any of ξ_j;

$\top$: satisfied by *any* action.

MoSL: Action specifiers and action sets (cont'd)

ACTION SPECIFIER	IS SATISFIED BY
$g_1 : \mathbf{O}(\vec{F}, g_2)$	any **out** action, executed at a site i_1 matching g_1, uploading a tuple $\vec{f}$ matching $\vec{F}$ to a site i_2 matching g_2
$g_1 : \mathbf{I}(\vec{F}, g_2)$	any **in** action, executed at a site i_1 matching g_1, downloading a tuple $\vec{f}$ matching $\vec{F}$ from a site i_2 matching g_2
$g_1 : \mathbf{R}(\vec{F}, g_2)$	any **read** action, executed at a site i_1 matching g_1, reading a tuple $\vec{f}$ matching $\vec{F}$ from a site i_2 matching g_2
$g_1 : \mathbf{E}(F, g_2)$	any **eval** action, executed at a site i_1 matching g_1, spawning a process P matching F to a site i_2 matching g_2
$g_1 : \mathbf{N}(g_2)$	any **newloc** action executed at a site i_1 matching g_1, creating a node with physical address i_2 matching g_2

MoSL: Path formulae

$$\Phi \ {}_{\Delta}\mathcal{U}_{\Omega}^{<t} \ \Psi$$

- Satisfied by those paths where eventually a Ψ-state is reached, by time t, via a Φ-path, *and*, in addition, while evolving between Φ states, actions are performed satisfying Δ and the Ψ-state is entered via an action satisfying Ω.

MoSL: Path formulae

$$\Phi \;\; {}_{\Delta}\mathcal{U}^{<t}_{\Omega} \; \Psi$$

- Satisfied by those paths where eventually a Ψ-state is reached, by time t, via a Φ-path, *and*, in addition, while evolving between Φ states, actions are performed satisfying Δ and the Ψ-state is entered via an action satisfying Ω.
- Variables occurring in Ω can be used in Ψ.

MoSL: Path formulae

$$\Phi \; {}_{\Delta}\mathcal{U}^{<t}_{\Omega} \; \Psi$$

- Satisfied by those paths where eventually a Ψ-state is reached, by time t, via a Φ-path, *and*, in addition, while evolving between Φ states, actions are performed satisfying Δ and the Ψ-state is entered via an action satisfying Ω.
- Variables occurring in Ω can be used in Ψ.
- Simpler operator: $\Phi \; {}_{\Delta}\mathcal{U}^{<t} \; \Psi$.

$$\Phi \ {}_{\Delta}\mathcal{U}_{\Omega}^{<t} \ \Psi$$

- Satisfied by those paths where eventually a Ψ-state is reached, by time t, via a Φ-path, *and*, in addition, while evolving between Φ states, actions are performed satisfying Δ and the Ψ-state is entered via an action satisfying Ω.
- Variables occurring in Ω can be used in Ψ.
- Simpler operator: $\Phi \ {}_{\Delta}\mathcal{U}^{<t} \ \Psi$.
- Time t can be omitted (assumed as ∞).

$$\Phi \; {}_{\Delta}\mathcal{U}_{\Omega}^{<t} \; \Psi$$

- Satisfied by those paths where eventually a Ψ-state is reached, by time t, via a Φ-path, *and*, in addition, while evolving between Φ states, actions are performed satisfying Δ and the Ψ-state is entered via an action satisfying Ω.
- Variables occurring in Ω can be used in Ψ.
- Simpler operator: $\Phi \; {}_{\Delta}\mathcal{U}^{<t} \; \Psi$.
- Time t can be omitted (assumed as ∞).

$$\mathrm{tt} \; {}_{\top}\mathcal{U}_{\{init:\mathbf{O}(GO,A)\}}^{<t} \; \mathrm{tt}$$

$$\Phi \;\; {}_{\Delta}\mathcal{U}^{<t}_{\Omega} \; \Psi$$

- Satisfied by those paths where eventually a Ψ-state is reached, by time t, via a Φ-path, *and*, in addition, while evolving between Φ states, actions are performed satisfying Δ and the Ψ-state is entered via an action satisfying Ω.
- Variables occurring in Ω can be used in Ψ.
- Simpler operator: $\Phi \;\; {}_{\Delta}\mathcal{U}^{<t} \; \Psi$.
- Time t can be omitted (assumed as ∞).

$\mathrm{tt} \; {}_{\top}\mathcal{U}^{<t}_{\{init:\mathbf{O}(GO,A)\}} \; \mathrm{tt} \qquad \mathrm{tt} \; {}_{\top}\mathcal{U}^{\leq t}_{\top} \; \langle GO \rangle @ A$

$$\Phi \;\; {}_{\Delta}\mathcal{U}^{<t}_{\Omega} \; \Psi$$

- Satisfied by those paths where eventually a Ψ-state is reached, by time t, via a Φ-path, *and*, in addition, while evolving between Φ states, actions are performed satisfying Δ and the Ψ-state is entered via an action satisfying Ω.
- Variables occurring in Ω can be used in Ψ.
- Simpler operator: $\Phi \;\; {}_{\Delta}\mathcal{U}^{<t} \; \Psi$.
- Time t can be omitted (assumed as ∞).

$\mathrm{tt} \; {}_{\top}\mathcal{U}^{<t}_{\{init:\mathbf{O}(GO,A)\}} \; \mathrm{tt}$ $\quad$ $\mathrm{tt} \; {}_{\top}\mathcal{U}^{<t}_{\top} \; \langle GO\rangle @A$ $\quad$ $\mathrm{tt} \; {}_{\top}\mathcal{U}^{<\infty}_{\{i_1:\mathbf{N}(!z)\}} \; \mathbf{nil}@z$

MoSL: State formulae

$$\Phi \quad ::= \quad \chi$$

MoSL: State formulae

$$\Phi \quad ::= \quad \mathsf{tt} \mid \langle \vec{F} \rangle @ \imath \mid Q @ \imath \mid$$

MoSL: State formulae

$$\Phi \quad ::= \quad \mathsf{tt} \mid \langle \vec{F} \rangle @\imath \mid Q @\imath \mid$$

$$\neg\, \Phi \mid \Phi \vee \Phi \mid$$

MoSL: State formulae

$$\Phi \quad ::= \quad \mathsf{tt} \mid \langle \vec{F} \rangle @ \imath \mid Q @ \imath \mid$$

$$\neg\, \Phi \mid \Phi \vee \Phi \mid$$

$$\langle \vec{F} \rangle @ \imath \rightarrow \Phi$$

MoSL: State formulae

$$\Phi \quad ::= \quad \mathsf{tt} \mid \langle \vec{F} \rangle @\imath \mid Q@\imath \mid$$

$$\neg \Phi \mid \Phi \vee \Phi \mid$$

$$\langle \vec{F} \rangle @\imath \rightarrow \Phi \text{ } \textit{consumption}$$

MoSL: State formulae

$$\Phi \quad ::= \quad \mathsf{tt} \mid \langle \vec{F} \rangle @\imath \mid Q@\imath \mid$$

$$\neg\, \Phi \mid \Phi \,\vee\, \Phi \mid$$

$$\langle \vec{F} \rangle @\imath \rightarrow \Phi \mid Q@\imath \rightarrow \Phi$$

MoSL: State formulae

$$\Phi \quad ::= \quad \mathsf{tt} \mid \langle \vec{F} \rangle @ \imath \mid Q @ \imath \mid$$

$$\neg\, \Phi \mid \Phi \vee \Phi \mid$$

$$\langle \vec{F} \rangle @ \imath \rightarrow \Phi \mid Q @ \imath \rightarrow \Phi \mid$$

$$\langle \vec{f} \rangle @ \imath \leftarrow \Phi$$

MoSL: State formulae

$$
\begin{aligned}
\Phi \quad ::= \quad & \mathsf{tt} \mid \langle \vec{F} \rangle @\imath \mid Q@\imath \mid \\
& \neg\, \Phi \mid \Phi \,\vee\, \Phi \mid \\
& \langle \vec{F} \rangle @\imath \rightarrow \Phi \mid Q@\imath \rightarrow \Phi \mid \\
& \langle \vec{f} \rangle @\imath \leftarrow \Phi \;\; \textit{production}
\end{aligned}
$$

MoSL: State formulae

$$
\begin{aligned}
\Phi \quad ::= \quad & \mathsf{tt} \mid \langle \vec{F} \rangle @\imath \mid Q@\imath \mid \\
& \neg \Phi \mid \Phi \vee \Phi \mid \\
& \langle \vec{F} \rangle @\imath \rightarrow \Phi \mid Q@\imath \rightarrow \Phi \mid \\
& \langle \vec{f} \rangle @\imath \leftarrow \Phi \mid Q@\imath \leftarrow \Phi
\end{aligned}
$$

MoSL: State formulae

$$\begin{aligned}\Phi \quad ::= \quad & \mathsf{tt} \mid \langle \vec{F} \rangle @ \imath \mid Q @ \imath \mid \\ & \neg\, \Phi \mid \Phi \,\vee\, \Phi \mid \\ & \langle \vec{F} \rangle @ \imath \rightarrow \Phi \mid Q @ \imath \rightarrow \Phi \mid \\ & \langle \vec{f} \rangle @ \imath \leftarrow \Phi \mid Q @ \imath \leftarrow \Phi \mid \\ & \mathcal{P}_{\bowtie p}(\varphi)\end{aligned}$$

with $\bowtie \in \{<, >, \leq, \geq\}$ and $p \in [0, 1]$

$$
\begin{aligned}
\Phi \quad ::= \quad & \mathsf{tt} \mid \langle \vec{F} \rangle @ \imath \mid Q @ \imath \mid \\
& \neg\, \Phi \mid \Phi \,\vee\, \Phi \mid \\
& \langle \vec{F} \rangle @ \imath \rightarrow \Phi \mid Q @ \imath \rightarrow \Phi \mid \\
& \langle \vec{f} \rangle @ \imath \leftarrow \Phi \mid Q @ \imath \leftarrow \Phi \mid \\
& \mathcal{P}_{\bowtie p}(\varphi) \mid \\
& \mathcal{S}_{\bowtie p}(\Phi)
\end{aligned}
$$

with $\bowtie \in \{<, >, \leq, \geq\}$ and $p \in [0, 1]$

State Formulae Satisfaction relation

$$
\begin{array}{lllll}
s & \models & \langle\vec{F}\rangle @\imath \rightarrow \Phi & \text{iff} & s \equiv \imath ::_\rho \langle\vec{f}\rangle \parallel s', match(\vec{F},\vec{f}) = \theta, \text{ and } s' \models \Phi \\
s & \models & \langle\vec{f}\rangle @\imath \leftarrow \Phi & \text{iff} & s \equiv \imath ::_\rho \mathbf{nil} \parallel s' \text{ and } \imath ::_\rho \langle\vec{f}\rangle \parallel s \models \Phi \\
s & \models & \mathcal{P}_{\bowtie p}(\varphi) & \text{iff} & \mathbf{IP}\{\pi \in \text{Paths(s)} \mid \pi \models \varphi\} \bowtie p \\
s & \models & \mathcal{S}_{\bowtie p}(\Phi) & \text{iff} & lim_{t\rightarrow\infty}\mathbf{IP}\{\pi \in \text{Paths(s)} \mid \pi[t] \models \Phi\} \bowtie p
\end{array}
$$

Path Formulae Satisfaction relation

$\pi \models \Phi \; {}_{\Delta}\mathcal{U}_{\Omega}^{<t} \; \Psi$ if and only if there exists $k, 0 < k \leq \mathrm{len}(\pi)$ s.t.:

1. $\sum_{j=0}^{k-1} \mathrm{time}(\pi, j) < t$
2. there exists Θ_{k-1} s.t.:
   1. $\mathrm{state}(\pi, k-1) \models \Phi$,
   2. $\mathrm{act}(\pi, k-1), \Theta_{k-1} \models \Omega$,
   3. $\mathrm{state}(\pi, k) \models \Psi\Theta_{k-1}$
3. if $k > 1$ there exist $\Theta_0, \ldots, \Theta_{k-2}$ s.t. for all $j, 0 \leq j \leq k-2$:
   1. $\mathrm{state}(\pi, j) \models \Phi$,
   2. $\mathrm{act}(\pi, j), \Theta_j \models \Delta$

$\gamma, \Theta \models \top$
$\gamma, \Theta \models \{\xi_1, \ldots, \xi_n\}$ iff there exists $0 < j \leq n$ s.t. $\gamma, \Theta \models \xi_j$
$\gamma, \Theta \models \xi$ iff $\mathrm{match}(\xi, \gamma) = \Theta$

MoSL: Examples

MoSL: Examples

- In the long run, the probability of finding the resource at A free is at least 0.2:

MoSL: Examples

- In the long run, the probability of finding the resource at A free is at least 0.2:

$$\mathcal{S}_{\geq 0.2}(\langle AF \rangle @ A)$$

MoSL: Examples

- In the long run, the probability of finding the resource at A free is at least 0.2:

$$\mathcal{S}_{\geq 0.2}(\langle AF \rangle @ A)$$

- If, in the current state, there is a request for a $S2$-type service placed on site A, the probability that such a request gets served within 72.04 time-units is at least 0.85:

MoSL: Examples

- In the long run, the probability of finding the resource at A free is at least 0.2:

$$\mathcal{S}_{\geq 0.2}(\langle AF \rangle @ A)$$

- If, in the current state, there is a request for a $S2$-type service placed on site A, the probability that such a request gets served within 72.04 time-units is at least 0.85:

$$\langle S2 \rangle @ A \Rightarrow \ \mathcal{P}_{\geq 0.85}(\mathrm{tt}\ {}_{\top}\mathcal{U}^{<72.04}_{\{A:\mathbf{I}(S2,A)\}}\ \mathrm{tt})$$

MoSL: Examples

- In the long run, the probability of finding the resource at A free is at least 0.2:

$$\mathcal{S}_{\geq 0.2}(\langle AF \rangle @ A)$$

- If, in the current state, there is a request for a *S2*-type service placed on site A, the probability that such a request gets served within 72.04 time-units is at least 0.85:

$$\langle S2 \rangle @ A \Rightarrow \; \mathcal{P}_{\geq 0.85}(\mathrm{tt}\ {}_{\top}\mathcal{U}^{<72.04}_{\{A:\mathbf{I}(S2,A)\}}\ \mathrm{tt})$$

- In equilibrium, the probability is at least 0.87 that in at least 75% of the cases a *S1*-type request is placed at site A within 500 time units:

MoSL: Examples

- In the long run, the probability of finding the resource at A free is at least 0.2:

$$\mathcal{S}_{\geq 0.2}(\langle AF \rangle @ A)$$

- If, in the current state, there is a request for a *S2*-type service placed on site A, the probability that such a request gets served within 72.04 time-units is at least 0.85:

$$\langle S2 \rangle @ A \Rightarrow \ \mathcal{P}_{\geq 0.85}(\mathrm{tt}\ {}_{\top}\mathcal{U}^{<72.04}_{\{A:\mathbf{I}(S2,A)\}}\ \mathrm{tt})$$

- In equilibrium, the probability is at least 0.87 that in at least 75% of the cases a *S1*-type request is placed at site A within 500 time units:

$$\mathcal{S}_{\geq 0.87}(\mathcal{P}_{\geq 0.75}(\mathrm{tt}\ {}_{\top}\mathcal{U}^{<500}_{\{!z:\mathbf{O}(S1,A)\}}\ \mathrm{tt}))$$

The Airbag scenario - Sensoria D8.0

When an accident occurs to a car registered with the Accident Assistance Service and the airbag of the car deploys, the following happens. First, an automated message is sent to the Accident Assistance Server which contains the vehicle's GPS data, the vehicle identification number and a collection of sensor data. Then, the Accident Assistance Server places a call to the driver's mobile phone. If, due to his injuries, the driver is unable to answer the call, and the severity of the accident is confirmed also by the sensor data, the emergency services are alerted and the vehicle location is communicated to them. Incoming car are alerted as well.

The Airbag scenario - Sensoria D8.0

Simplifications

The Airbag scenario - Sensoria D8.0

Simplifications

- Airabag deployment is not considered.

Simplifications

- Airabag deployment is not considered.
- If the driver does not answer the phone call, the emergency services are alerted *anyway*.

The Airbag scenario - Sensoria D8.0

Simplifications

- Airabag deployment is not considered.
- If the driver does not answer the phone call, the emergency services are alerted *anyway*.
- Alerting cars which are approaching the accident area *not considered*.

The Airbag scenario - Sensoria D8.0

Modelling assumptions

The Airbag scenario - Sensoria D8.0

Modelling assumptions

- Each entity of interest is (assumed modeled as a StoKlaim site and is) provided with its physical address.
    - Accident Assistance Server address: *AccAssSrv*,
    - Phone Server address: *PhSrv*,
    - Emergency Server address: *EmSrv*,
    - Each car has its own address.

Modelling assumptions

- Each entity of interest is (assumed modeled as a StoKlaim site and is) provided with its physical address.
  - Accident Assistance Server address: *AccAssSrv*,
  - Phone Server address: *PhSrv*,
  - Emergency Server address: *EmSrv*,
  - Each car has its own address.
- Each car is uniquely identified by a car identifier $CarID \in CID$, *finite*.

The Airbag scenario - Sensoria D8.0

Modelling assumptions

- Each entity of interest is (assumed modeled as a StoKlaim site and is) provided with its physical address.
  - Accident Assistance Server address: *AccAssSrv*,
  - Phone Server address: *PhSrv*,
  - Emergency Server address: *EmSrv*,
  - Each car has its own address.
- Each car is uniquely identified by a car identifier $CarID \in CID$, *finite*.
- Accident notification:
  storing $(CarID, GPSData, SenData)$ @ $AccAssSrv$

Modelling assumptions

- Each entity of interest is (assumed modeled as a StoKlaim site and is) provided with its physical address.
  - Accident Assistance Server address: *AccAssSrv*,
  - Phone Server address: *PhSrv*,
  - Emergency Server address: *EmSrv*,
  - Each car has its own address.
- Each car is uniquely identified by a car identifier $CarID \in CID$, *finite*.
- Accident notification:
  storing $(CarID, GPSData, SenData)$ @ *AccAssSrv*
- $PhNr : CID \rightarrow \mathrm{PhN}$; $PhNr(carID) = carID$ driver's mobile phone nr.

The Airbag scenario - Sensoria D8.0

Modelling assumptions

- Each entity of interest is (assumed modeled as a StoKlaim site and is) provided with its physical address.
  - Accident Assistance Server address: *AccAssSrv*,
  - Phone Server address: *PhSrv*,
  - Emergency Server address: *EmSrv*,
  - Each car has its own address.
- Each car is uniquely identified by a car identifier $CarID \in CID$, *finite*.
- Accident notification:
  storing $(CarID, GPSData, SenData)$ @ $AccAssSrv$
- $PhNr : CID \rightarrow$ PhN; $PhNr(carID) = carID$ driver's mobile phone nr.
- Placing a phone-call: storing the phone number to site *PhSrv*

The Airbag scenario - Sensoria D8.0

Modelling assumptions

- Each entity of interest is (assumed modeled as a StoKlaim site and is) provided with its physical address.
  - Accident Assistance Server address: *AccAssSrv*,
  - Phone Server address: *PhSrv*,
  - Emergency Server address: *EmSrv*,
  - Each car has its own address.
- Each car is uniquely identified by a car identifier $CarID \in CID$, *finite*.
- Accident notification:
  storing $(CarID, GPSData, SenData)$ `@` *AccAssSrv*
- $PhNr : CID \rightarrow \mathrm{PhN}$; $PhNr(carID) = carID$ driver's mobile phone nr.
- Answering a phone-call: removing the phone number from site *PhSrv*

Modelling assumptions

- Each entity of interest is (assumed modeled as a StoKlaim site and is) provided with its physical address.
    - Accident Assistance Server address: *AccAssSrv*,
    - Phone Server address: *PhSrv*,
    - Emergency Server address: *EmSrv*,
    - Each car has its own address.
- Each car is uniquely identified by a car identifier $CarID \in CID$, *finite*.
- Accident notification:
  storing $(CarID, GPSData, SenData)$ @ *AccAssSrv*
- $PhNr : CID \to$ PhN; $PhNr(carID) =$ *carID* driver's mobile phone nr.
- Answering a phone-call: removing the phone number from site *PhSrv*
- The Accident Assistance Server alerts the Emergency Server by uploading the GPS data to site *EmSrv*.

Guaranteeing acceptable timing for the detection of an accident which seriously injured the driver and for rescue alerting.

Performance (Response time)

Suppose it has been detected that an accident involving car *car_id* has taken place and that the driver is seriously injured

Ideal Requirement:

$\boxed{\text{Emergency Service alerted within maximal time } t\_alert}$

Performance (Response time)

Suppose it has been detected that an accident involving car *car_id* has taken place and that the driver is seriously injured

Realistic Requirement:

Emergency Service alerted within maximal time *t_alert* in at least 99.7% of the cases

Performance (Response time)

Suppose it has been detected that an accident involving car *car_id* has taken place and that the driver is seriously injured

Realistic Requirement:

Emergency Service alerted within maximal time *t_alert* in at least 99.7% of the cases

■-property:

$$\mathcal{P}_{\geq 0.997}\left({}_{\top}\Diamond^{<t\_alert}_{\{AccAssSrv:\mathbf{o}((car\_id,gps),EmSrv)\}}\ \mathrm{tt}\right)$$

Performance (Response time, cont'd)

Focus on those states reached after a phone call has been placed to the driver mobile phone, and nobody has answered for a given period of time, *t_answ*.

Ideal Requirement:

> It should never happen that the phone is ringing for *t_answ* time units without a ■-state being reached.

Performance (Response time, cont'd)

Focus on those states reached after a phone call has been placed to the driver mobile phone, and nobody has answered for a given period of time, t_answ.

Realistic Requirement:

> In at most 0.2% of the cases the phone is ringing for t_answ time units without a ■-state being reached.

Performance (Response time, cont'd)

Focus on those states reached after a phone call has been placed to the driver mobile phone, and nobody has answered for a given period of time, t_answ.

Realistic Requirement:

In at most 0.2% of the cases the phone is ringing for t_answ time units without a ■-state being reached.

■-property:

$$\mathcal{P}_{<0.002}(\langle PhNr(car\_id)\rangle @PhSrv\ {}_{\top}\mathcal{U}^{[t\_answ,t\_answ]}\ \neg ■)$$

Performance (Response time, cont'd)

Consider now the situation in which an accident has been just notified.

Ideal Requirement:

> A phone call is placed to the driver mobile phone with a delay of at most *t_call* time units, bringing to a ■-state.

Performance (Response time, cont'd)

Consider now the situation in which an accident has been just notified.

Realistic Requirement:

> In at least 99.8% of the cases, a phone call is placed to the driver mobile phone with a delay of at most *t_call* time units, bringing to a ■-state.

Performance (Response time, cont'd)

Consider now the situation in which an accident has been just notified.

Realistic Requirement:

In at least 99.8% of the cases, a phone call is placed to the driver mobile phone with a delay of at most *t_call* time units, bringing to a ■-state.

■-property:

$$\mathcal{P}_{\geq 0.998}\left(\top \Diamond^{<t\_call}_{\{AccAssSrv:\mathbf{o}(PhNr(car\_id),PhSrv)\}} \blacksquare\right)$$

Consider now the situation in which an accident has been just notified.

Realistic Requirement:

In at least 99.8% of the cases, a phone call is placed to the driver mobile phone with a delay of at most *t_call* time units, bringing to a ■-state.

■-property:

Acceptable Behaviour

Performance (Response time, cont'd)

Consider now the situation in which an accident has been just notified.

Realistic Requirement:

> In at least 99.8% of the cases, a phone call is placed to the driver mobile phone with a delay of at most *t_call* time units, bringing to a ■-state.

■-property:

$$\mathcal{P}_{\geq 0.998}\left(\top \Diamond^{<t\_call}_{\{AccAssSrv:\mathbf{o}(PhNr(car\_id),PhSrv)\}} \blacksquare\right)$$

Performance (Response time, cont'd)

Consider now the situation in which an accident has been just notified.

Realistic Requirement:

> In at least 99.8% of the cases, a phone call is placed to the driver mobile phone with a delay of at most *t_call* time units, bringing to a ■-state.

■-property:

> $\mathcal{A}cceptable\ \mathcal{B}ehaviour$

Performance (Response time, cont'd)

REQUIREMENT:

Ideal Requirement:

The system does not behave *unacceptably*, after an accident has been notified.

Performance (Response time, cont'd)

REQUIREMENT:

Realistic Requirement:

The probability that the system behaves *unacceptably*, after an accident has been notified, is less than 0.004.

Performance (Response time, cont'd)

REQUIREMENT:

Realistic Requirement:

The probability that the system behaves *unacceptably*, after an accident has been notified, is less than 0.004.

FINAL property:

$$\mathcal{P}_{<0.004}\left({}_{\top}\Diamond_{\{!car\_addr:\mathbf{o}((!car\_id,!gps,!sdata),AccAssSrv)\}}\, \neg \blacksquare\right)$$

Performance (Response time, cont'd)

REQUIREMENT:

Realistic Requirement:

> The probability that the system behaves *unacceptably*, after an accident has been notified, is less than 0.004.

FINAL property:

$$\mathcal{P}_{<0.004}\left({}_{\top}\Diamond_{\{!car\_addr:\mathbf{o}((!car\_id,!gps,!sdata),AccAssSrv)\}}\ \neg\blacksquare\right)$$

Performance (Response time, cont'd)

REQUIREMENT:

Realistic Requirement:

The probability that the system behaves *unacceptably*, after an accident has been notified, is less than 0.004.

FINAL property:

$$\mathcal{P}_{<0.004}\left({}_{\top}\Diamond_{\{!car\_addr:\mathbf{o}((!car\_id,!gps,!sdata),AccAssSrv)\}}\ \neg\blacksquare\right)$$

REQUIREMENT:

Realistic Requirement:

> The probability that the system behaves *unacceptably*, after an accident has been notified, is less than 0.004.

FINAL property:

$$\mathcal{P}_{<0.004}\left({}_{\top}\Diamond_{\{!car\_addr:\mathbf{o}((!car\_id,!gps,!sdata),AccAssSrv)\}}\ \neg\blacksquare\right)$$

AB_REQUIREMENT altogether:

$$\mathcal{P}_{<0.004}({}_{\top}\Diamond_{\{!car\_addr:\mathbf{O}((!car\_id,!gps,!sdata),AccAssSrv)\}}\ \neg\blacksquare)$$

$$\mathcal{P}_{\geq 0.998}({}_{\top}\Diamond^{<t\_call}_{\{AccAssSrv:\mathbf{O}(PhNr(car\_id),PhSrv)\}}\ \blacksquare)$$

$$\mathcal{P}_{<0.002}(\langle PhNr(car\_id)\rangle @PhSrv\ {}_{\top}\mathcal{U}^{[t\_answ,t\_answ]}\ \neg\blacksquare)$$

$$\mathcal{P}_{\geq 0.997}({}_{\top}\Diamond^{<t\_alert}_{\{AccAssSrv:\mathbf{O}((car\_id,gps),EmSrv)\}}\ \mathrm{tt})$$

Performance (Response time, cont'd)

AB_REQUIREMENT altogether:

$$\mathcal{P}_{<0.004}({}_{\top}\Diamond_{\{!car\_addr:\mathbf{O}((!car\_id,!gps,!sdata),AccAssSrv)\}}\ \neg\blacksquare)$$

$$\mathcal{P}_{\geq 0.998}({}_{\top}\Diamond^{<t\_call}_{\{AccAssSrv:\mathbf{O}(PhNr(car\_id),PhSrv)\}}\ \blacksquare)$$

$$\mathcal{P}_{<0.002}(\langle PhNr(car\_id)\rangle @PhSrv\ {}_{\top}\mathcal{U}^{[t\_answ,t\_answ]}\ \neg\blacksquare)$$

$$\mathcal{P}_{\geq 0.997}({}_{\top}\Diamond^{<t\_alert}_{\{AccAssSrv:\mathbf{O}((car\_id,gps),EmSrv)\}}\ \mathrm{tt})$$

... trivially satisfied if there are no car accidents!

Performance (Response time, cont'd)

AB_REQUIREMENT altogether:

$$\mathcal{P}_{<0.004}\left({}_{\top}\Diamond_{\{!car\_addr:\mathbf{O}((!car\_id,!gps,!sdata),AccAssSrv)\}}\ \neg\blacksquare\right)$$

$$\mathcal{P}_{\geq 0.998}\left({}_{\top}\Diamond^{<t\_call}_{\{AccAssSrv:\mathbf{O}(PhNr(car\_id),PhSrv)\}}\ \blacksquare\right)$$

$$\mathcal{P}_{<0.002}\left(\langle PhNr(car\_id)\rangle @ PhSrv\ {}_{\top}\mathcal{U}^{[t\_answ,t\_answ]}\ \neg\blacksquare\right)$$

$$\mathcal{P}_{\geq 0.997}\left({}_{\top}\Diamond^{<t\_alert}_{\{AccAssSrv:\mathbf{O}((car\_id,gps),EmSrv)\}}\ tt\right)$$

... trivially satisfied if there are no car accidents! Thus we also assume

there are no paths without accident notifications reported

Performance (Response time, cont'd)

AB_REQUIREMENT altogether:

$$\mathcal{P}_{<0.004}({}_{\top}\Diamond_{\{!car\_addr:\mathbf{O}((!car\_id,!gps,!sdata),AccAssSrv)\}}\ \neg\blacksquare)$$

$$\mathcal{P}_{\geq 0.998}({}_{\top}\Diamond^{<t\_call}_{\{AccAssSrv:\mathbf{O}(PhNr(car\_id),PhSrv)\}}\ \blacksquare)$$

$$\mathcal{P}_{<0.002}(\langle PhNr(car\_id)\rangle @ PhSrv\ {}_{\top}\mathcal{U}^{[t\_answ,t\_answ]}\ \neg\blacksquare)$$

$$\mathcal{P}_{\geq 0.997}({}_{\top}\Diamond^{<t\_alert}_{\{AccAssSrv:\mathbf{O}((car\_id,gps),EmSrv)\}}\ tt)$$

... trivially satisfied if there are no car accidents! Thus we also assume

$$\mathcal{P}_{\leq 0}({}_{\top}\Box\ \neg(\langle(!car\_id,!gps,!sdata)\rangle @ AccAssSrv))$$

Performance (Response time, cont'd)

AB_REQUIREMENT altogether:

$$\mathcal{P}_{<0.004}({}_{\top}\Diamond_{\{!car\_addr:\mathbf{O}((!car\_id,!gps,!sdata),AccAssSrv)\}}\ \neg\blacksquare)$$

$$\mathcal{P}_{\geq 0.998}({}_{\top}\Diamond^{<t\_call}_{\{AccAssSrv:\mathbf{O}(PhNr(car\_id),PhSrv)\}}\ \blacksquare)$$

$$\mathcal{P}_{<0.002}(\langle PhNr(car\_id)\rangle @ PhSrv\ {}_{\top}\mathcal{U}^{[t\_answ,t\_answ]}\ \neg\blacksquare)$$

$$\mathcal{P}_{\geq 0.997}({}_{\top}\Diamond^{<t\_alert}_{\{AccAssSrv:\mathbf{O}((car\_id,gps),EmSrv)\}}\ tt)$$

... trivially satisfied if there are no car accidents! Thus we also assume

$$\mathcal{P}_{\leq 0}({}_{\top}\Box\ \neg(\langle(!car\_id, !gps, !sdata)\rangle @ AccAssSrv))$$

Periodic testing the system with *fake* accident notifications

Performance (Response time, cont'd)

AB_REQUIREMENT altogether:

$$\mathcal{P}_{<0.004}({}_{\top}\Diamond_{\{!car\_addr:\mathbf{O}((!car\_id,!gps,!sdata),AccAssSrv)\}}\ \neg\blacksquare)$$

$$\mathcal{P}_{\geq 0.998}({}_{\top}\Diamond^{<t\_call}_{\{AccAssSrv:\mathbf{O}(PhNr(car\_id),PhSrv)\}}\ \blacksquare)$$

$$\mathcal{P}_{<0.002}(\langle PhNr(car\_id)\rangle @PhSrv\ {}_{\top}\mathcal{U}^{[t\_answ,t\_answ]}\ \neg\blacksquare)$$

$$\mathcal{P}_{\geq 0.997}({}_{\top}\Diamond^{<t\_alert}_{\{AccAssSrv:\mathbf{O}((car\_id,gps),EmSrv)\}}\ tt)$$

... trivially satisfied if there are no car accidents! Thus we also assume

$$\mathcal{P}_{\leq 0}({}_{\top}\Box\ \neg(\langle(!car\_id,!gps,!sdata)\rangle @AccAssSrv))$$

Liveness of diagnostic routines:

AB_REQUIREMENT altogether:

$$\mathcal{P}_{<0.004}({}_{\top}\Diamond_{\{!car\_addr:\mathbf{O}((!car\_id,!gps,!sdata),AccAssSrv)\}}\ \neg\blacksquare)$$

$$\mathcal{P}_{\geq 0.998}({}_{\top}\Diamond^{<t\_call}_{\{AccAssSrv:\mathbf{O}(PhNr(car\_id),PhSrv)\}}\ \blacksquare)$$

$$\mathcal{P}_{<0.002}(\langle PhNr(car\_id)\rangle @ PhSrv\ {}_{\top}\mathcal{U}^{[t\_answ,t\_answ]}\ \neg\blacksquare)$$

$$\mathcal{P}_{\geq 0.997}({}_{\top}\Diamond^{<t\_alert}_{\{AccAssSrv:\mathbf{O}((car\_id,gps),EmSrv)\}}\ tt)$$

... trivially satisfied if there are no car accidents! Thus we also assume

$$\mathcal{P}_{\leq 0}({}_{\top}\Box\ \neg(\langle(!car\_id,!gps,!sdata)\rangle @ AccAssSrv))$$

Liveness of diagnostic routines:

$$\mathcal{P}_{\geq 1}({}_{\top}\Box\ \mathcal{P}_{\geq 1}({}_{\top}\Diamond_{\{AccAssSrv:\mathbf{O}((TEST\_ID,TEST\_GPS,TEST\_SD),AccAssSrv)\}}\ tt))$$

Openess

Openess

- Take a model: the italian road infrastructure $\mathcal{M}$

Openess

- Take a model: the italian road infrastructure $\mathcal{M}$
- $\mathcal{M} \models AB\_REQUIREMENT$

Openess

- Take a model: the italian road infrastructure $\mathcal{M}$
- $\mathcal{M} \models \mathit{AB\_REQUIREMENT}$
- What happens if I buy a car in Pisa and drive around?

Openess

- Take a model: the italian road infrastructure $\mathcal{M}$
- $\mathcal{M} \models AB\_REQUIREMENT$
- What happens if I buy a car in Pisa and drive around?
- Production furmulae can be used:

Openess

- Take a model: the italian road infrastructure $\mathcal{M}$
- $\mathcal{M} \models AB\_REQUIREMENT$
- What happens if I buy a car in Pisa and drive around?
- Production furmulae can be used:
- $\mathcal{M} \models MY\_CAR@Pisa \leftarrow AB\_REQUIREMENT$

Openess

- Take a model: the italian road infrastructure $\mathcal{M}$
- $\mathcal{M} \models AB\_REQUIREMENT$
- What happens if I buy a car in Pisa and drive around?
- Production furmulae can be used:
- $\mathcal{M} \models MY\_CAR@Pisa \leftarrow AB\_REQUIREMENT$
- $\mathcal{M} \models MICHELE\_CAR@Empoli \leftarrow MY\_CAR@Pisa \leftarrow AB\_REQUIREMENT$

Availability

Assume furthermore that the presence of token *FAULT* at address *AccAssSrv* signals that the Accident Assistance Server is down.

Availability

Assume furthermore that the presence of token *FAULT* at address *AccAssSrv* signals that the Accident Assistance Server is down.

The system must be alive at least 99% in the average, in the long run

Availability

Assume furthermore that the presence of token *FAULT* at address *AccAssSrv* signals that the Accident Assistance Server is down.

The system must be alive at least 99% in the average, in the long run

$$\mathcal{S}_{>0.99}(\neg(\langle \mathit{FAULT} \rangle @ \mathit{AccAssSrv}))$$

Reliability

Suppose that *retrieval* of a specific emergency token $ERR003$ from site *AccAssSrv* is a symptom of a serious fault in the system (and that the diagnostic routines rate is sufficiently high).

Reliability

Suppose that *retrieval* of a specific emergency token *ERR*003 from site *AccAssSrv* is a symptom of a serious fault in the system (and that the diagnostic routines rate is sufficiently high).

Study of distribution of *time to failure*, w.r.t. *ERR003*

Reliability

Suppose that *retrieval* of a specific emergency token *ERR*003 from site *AccAssSrv* is a symptom of a serious fault in the system (and that the diagnostic routines rate is sufficiently high).

Study of distribution of *time to failure*, w.r.t. *ERR003*

$$\mathcal{P}_{\bowtie p}(\neg(\langle ERR003\rangle @ AccAssSrv)\ {}_{\top}\mathcal{U}^{[t,t]}\ \langle ERR003\rangle @ AccAssSrv).$$

Reliability

Suppose that *retrieval* of a specific emergency token $ERR003$ from site $AccAssSrv$ is a symptom of a serious fault in the system (and that the diagnostic routines rate is sufficiently high).

Study of distribution of *time to failure*, w.r.t. *ERR003*

$$\mathcal{P}_{\bowtie p}(\neg(\langle ERR003 \rangle @ AccAssSrv)\ {}_{\top}\mathcal{U}^{[t,t]}\ \langle ERR003 \rangle @ AccAssSrv).$$

- $\mathcal{U}^{[t,t']}$ generalises $\mathcal{U}^{<t}$

Reliability

Suppose that *retrieval* of a specific emergency token $ERR003$ from site *AccAssSrv* is a symptom of a serious fault in the system (and that the diagnostic routines rate is sufficiently high).

Study of distribution of *time to failure*, w.r.t. *ERR003*

$$\mathcal{P}_{\bowtie p}(\neg(\langle ERR003\rangle @AccAssSrv)\ {}_{\top}\mathcal{U}^{[t,t]}\ \langle ERR003\rangle @AccAssSrv).$$

- $\mathcal{U}^{[t,t']}$ generalises $\mathcal{U}^{<t}$
- model-checking $\mathcal{P}_{\bowtie p}(\varphi)$ automatically returns also $\mathbb{P}(\{\pi \mid \pi \models \varphi \text{ and } \pi \text{ from } s\})$ for all states s.

Developments

Developments

1. SAM Model Checker :
   Implementation of a model-checking algorithm for full MoSL *which uses* CSL model-checkers (MRMC)

Developments

1. SAM Model Checker :
   Implementation of a model-checking algorithm for full MoSL *which uses* CSL model-checkers (MRMC)
   - coding actions into (next) states plus some more technicalities;

Developments

1. SAM Model Checker :
   Implementation of a model-checking algorithm for full MoSL *which uses* CSL model-checkers (MRMC)
   - coding actions into (next) states plus some more technicalities;
   - variable binding (action specifiers);

Developments

1. SAM Model Checker :
   Implementation of a model-checking algorithm for full MoSL *which uses* CSL model-checkers (MRMC)

   - coding actions into (next) states plus some more technicalities;
   - variable binding (action specifiers);
   - **production & consumption operators**

Developments

1. SAM Model Checker :
   Implementation of a model-checking algorithm for full MoSL *which uses* CSL model-checkers (MRMC)
   - coding actions into (next) states plus some more technicalities;
   - variable binding (action specifiers);
   - **production & consumption operators**
2. More advanced Model-checking techniques

Developments

1. SAM Model Checker :
   Implementation of a model-checking algorithm for full MoSL *which uses* CSL model-checkers (MRMC)
   - coding actions into (next) states plus some more technicalities;
   - variable binding (action specifiers);
   - **production & consumption operators**
2. More advanced Model-checking techniques
   - On the fly path model-checking (DT - bounded/unbounded until);

Developments

1. SAM Model Checker :
   Implementation of a model-checking algorithm for full MoSL *which uses* CSL model-checkers (MRMC)

   - coding actions into (next) states plus some more technicalities;
   - variable binding (action specifiers);
   - **production & consumption operators**

2. More advanced Model-checking techniques

   - On the fly path model-checking (DT - bounded/unbounded until);
   - StoKLAIM uIMC semantics plus CTMDP model-checking.

SAM Model Checker

- StoKlaim simulation and reachability graph generation.
- StoKLAIM $\models$ MoSL verification
- implemented in OCaML
  - OCaML provides mechanisms for interoperability with the C language
  - these primitives are used for interacting with MRMC (computing *steady* and *path* probabilities)
- The tool is still at a prototype level
  - systems with about 10^6 states can be handled

References

- R. De Nicola, D. Latella, and M. Massink. Formal modeling and quantitative analysis of KLAIM-based mobile systems. ACM SAC 2005, pages 428–435.
- M. Bravetti, D. Latella, M. Loreti, M. Massink, and G. Zavattaro. Combining Timed Coordination Primitives and Probabilistic Tuple Spaces. TGC 2008. LNCS, Springer, (To appear)
- R. De Nicola, J.-P. Katoen, D. Latella, and M. Massink. Towards a logic for performance and mobility. QAPL 2005. volume 153(2) of *ENTCS*, pages 161–175. Elsevier Science Publishers B.V., 2006.
- R. De Nicola, J.-P. Katoen, D. Latella, M. Loreti, and M. Massink. Model Checking Mobile Stochastic Logic. *Theoretical Computer Science. Elsevier.*, 382(1):42–70, 2007.

References

- R. De Nicola, D. Latella, and M. Massink. Formal modeling and quantitative analysis of KLAIM-based mobile systems. ACM SAC 2005, pages 428–435.
- M. Bravetti, D. Latella, M. Loreti, M. Massink, and G. Zavattaro. Combining Timed Coordination Primitives and Probabilistic Tuple Spaces. TGC 2008. LNCS, Springer, (To appear)
- R. De Nicola, J.-P. Katoen, D. Latella, and M. Massink. Towards a logic for performance and mobility. QAPL 2005. volume 153(2) of *ENTCS*, pages 161–175. Elsevier Science Publishers B.V., 2006.
- R. De Nicola, J.-P. Katoen, D. Latella, M. Loreti, and M. Massink. Model Checking Mobile Stochastic Logic. *Theoretical Computer Science. Elsevier.*, 382(1):42–70, 2007.
- R. De Nicola, D. Latella, M. Loreti, and M. Massink. MarCaSPiS: a Markovian Extension of a Calculus for Services. SOS 2008, ENTCS, Elsevier (to appear)