Performability Measure Specification: Combining CSRL and MSL (ongoing work)

Alessandro Aldini\textsuperscript{1}, Marco Bernardo\textsuperscript{1}, Jeremy Sproston\textsuperscript{2}

\textsuperscript{1}Dipartimento di Matematica, Fisica e Informatica, University of Urbino “Carlo Bo”, Italy

\textsuperscript{2}Dipartimento di Informatica, University of Turin, Italy

L’Aquila, 2nd March 2010
Motivation

• Aim: further develop methods for the specification of performability measures.
• Define UMSL: a unified measure specification language.
• Combine two proposals made previously:
  • Stochastic temporal logic: CSRL [BaierHaverkortHermannsKatoen00].
  • Component-oriented language based on first-order logic: MSL [AldiniBernardo07].
Motivation

• CSRL is used to specify *path-based properties* of stochastic systems.
  
  • For example: “with probability at most 0.02, the system exhibits an execution path such that:
    
    • there is a point along the path at which the atomic proposition *fail* holds directly after performing a *break* action;
    • at this point no more than 30 time units have elapsed and at most 7 units of reward have been accumulated,
    • *up* holds at all preceding points,
    • and the action *ok* is the only action seen on the path before the action *break*.”
Motivation

- MSL is used to specify *reward structures* of stochastic systems.
  - Reward structure: assign a real number to each state corresponding to reward/cost per time unit accumulated in that state.
  - Component-oriented:
    - System state: vector of local states, one for each component.
    - MSL uses first-order logic to characterize the assignment of rewards to states.
    - Statements of the logic are built from local states and/or local activities (actions).
Motivation

- What is missing?
  - MSL can handle classical measures (throughput, utilization, response time etc.), but not path-based properties.
  - CSRL assumes the presence of a reward structure; its atomic propositions are not (necessarily) component-oriented.
Reference model

- Finite labeled CTMC: $\mathcal{M} = (S, T; L_s, L_t; N, Loc, Act)$
  - $S$ is a finite set of states.
  - $T \subseteq S \times \mathbb{R}_{>0} \times S$ is a finitely branching transition relation.
  - $L_s : S \rightarrow \text{Loc}^N$ is a labeling function for states.
  - $L_t : T \rightarrow \text{Act}$ is a labeling function for transitions.

- Use $s \xrightarrow{a, \lambda} \mathcal{M} s'$ to denote a labeled transition.
- Use $[z_1, z_2, \ldots, z_N]$ to denote vectors of local states (which label CTMC states).
- $z \in s$ (resp. $z \notin s$) denotes that $z = z_i$ for some $1 \leq i \leq N$ (resp. $z \neq z_i$ for all $1 \leq i \leq N$).
Reference model

- **Example:** system with two identical servers.
  - Requests arrive at the system with rate $\lambda \in \mathbb{R}_{>0}$.
  - When a request finds both servers busy, it must immediately leave the system.
  - When a request finds both servers idle, it has the same probability to be accepted by the two servers.
  - For $i \in \{1, 2\}$, server $i$ processes requests with rate $\mu_i$, fails with rate $\varphi_i$, is repaired with rate $\varrho_i$.

- **In Markovian process algebra:**
  - Arrival process $A$: $Arrivals \triangleq <\text{arrive}, \lambda>.Arrivals$
  - Server $S_i$:
    
    \[
    Idle \triangleq <\text{arrive}, \ast>.Busy
    \]
    
    \[
    Busy \triangleq <\text{serve}, \mu_i>.Idle + <\text{fail}, \varphi_i>.Failed
    \]
    
    \[
    Failed \triangleq <\text{repair}, \varrho_i>.Busy
    \]

- $A$ has a single local state: $A.Arrivals$.
- $S_i$ has three local states: $S_i.Idle$, $S_i.Busy$, and $S_i.Failed$. 
Formulas on local states and activities

- Define $dnfLoc$: disjunctive normal forms on local states

$$Z = (z_{1,1} \land \ldots \land z_{1,m_1}) \lor \ldots \lor (z_{n,1} \land \ldots \land z_{n,m_n})$$

where each literal $z_{i,j}$ is either a local state or the negation of a local state.

- Define $dnfAct$: disjunctive normal forms on activities

$$A = (a_{1,1} \land \ldots \land a_{1,m_1}) \lor \ldots \lor (a_{n,1} \land \ldots \land a_{n,m_n})$$

where each literal $a_{i,j}$ is either an activity or the negation of an activity.

- Shorthand: write $Z_i$ for a disjunct of $Z$ (same for $A_i$ and $A$).
Formulas on local states and activities

- Given state $s \in S$, local state $z \in Loc$, and activity $a \in Act$, define predicate $\text{sat}$ by letting:

  \[
  \begin{align*}
  s \text{ sat } z & \iff z \in s \\
  s \text{ sat } \overline{z} & \iff z \notin s \\
  s \text{ sat } a & \iff \exists \lambda \in \mathbb{R}_{>0}, s' \in S. s \xrightarrow{a, \lambda} M s' \\
  s \text{ sat } \overline{a} & \iff \forall \lambda \in \mathbb{R}_{>0}, s' \in S. s \xrightarrow{a, \lambda} M s'
  \end{align*}
  \]

- Then extend $\text{sat}$ to literal conjunction and disjunction in the expected way.
dnfMSL (preliminaries)

- \( eq : \mathbb{R} \times \mathbb{R} \rightarrow \{ \text{true}, \text{false} \} \) such that:
  \[
  eq(x, y) = \begin{cases} 
  \text{true} & \text{if } x = y \\
  \text{false} & \text{otherwise}
  \end{cases}
  \]

- \( state\_rew : S \rightarrow \mathbb{R} \) such that \( state\_rew(s) \) is the reward accumulated while staying in state \( s \) due to either local states of \( s \) or activities enabled by \( s \).

- \( lstate\_rew : \text{Loc} \cup \neg \text{Loc} \rightarrow \mathbb{R} \) such that \( lstate\_rew(z) \) is the reward contribution given by local state \( z \) and \( lstate\_rew(\bar{z}) \) is the reward contribution given by the negation of local state \( \bar{z} \).

- \( act\_rew : \text{Act} \cup \neg \text{Act} \rightarrow \mathbb{R} \) such that \( act\_rew(a) \) is the reward contribution given by activity \( a \) and \( act\_rew(\bar{a}) \) is the reward contribution given by the negation of activity \( \bar{a} \).
eq(state rew(s), sum lstate contrib(s, Z, af))

where:

- \(\text{sum lstate contrib} : S \times \text{dnfLoc} \times AF \rightarrow \mathbb{R}\) such that:

\[
\text{sum lstate contrib}(s, Z, af) = \sum_{Z_i \in Z \text{ s.t. } s \text{ sat } Z_i} af\{\mid \text{lstate rew}(z) \mid \text{z occurs in } Z_i \}\]

- \(af : 2^\mathbb{R} \rightarrow \mathbb{R}\): arithmetical function, e.g., sum, min, max, or average.
eq(state rew(s), sum act contrib(s, A, af))

where:

- \( \text{sum act contrib} : S \times \text{dnfAct} \times AF \to \mathbb{R} \) such that:

\[
\text{sum act contrib}(s, A, af) = \sum_{A_i \in A \text{ s.t. } s \text{ sat } A_i} af\{| \text{act rew}(a) | a \text{ occurs in } A_i \}.
\]
eq(state rew(s), choose lstate contrib(s, Z, af, cf))

where:

- \( \text{choose\_lstate\_contrib} : S \times \text{dnfLoc} \times AF \times CF \to \mathbb{R} \) such that:

\[
\text{choose\_lstate\_contrib}(s, Z, af, cf) = \frac{cf}{\{Z_i \in Z \mid \text{s sat } Z_i \text{ and } Z_i \text{ occurs in } Z_i\}}
\]

- \( cf : 2^\mathbb{R} \to \mathbb{R} \): choice functions; i.e., \( cf(\emptyset) = 0 \) and \( cf(\{x_1, \ldots, x_n\}) \in \{x_1, \ldots, x_n\} \) for all \( n \in \mathbb{N}_{>0} \).
eq(state rew(s), choose act contrib(s, A, af, cf))

where:

- \(\text{choose act contrib} : S \times \text{dnfAct} \times AF \times CF \rightarrow \mathbb{R}\) such that:

\[
\text{choose act contrib}(s, A, af, cf) = \sum_{A_i \in A \text{ s.t. } s \text{ sat } A_i} cf \cdot af\{\mid \text{act rew}(a) \mid a \text{ occurs in } A_i \}\]
aCSRL: syntax

- State formulas of aCSRL:

  \[
  \Phi ::= Z \mid A \mid \Phi \land \Phi \mid \neg \Phi \mid S_{\bowtie p}(\Phi) \mid P_{\bowtie p}(\varphi)
  \]

  where \(Z \in \text{dnfLoc}\) is a disjunctive normal form on local states, \(A \in \text{dnfAct}\) is a disjunctive normal form on activities, \(\bowtie \in \{<, \leq, \geq, >\}\) is a comparison operator, \(p \in \mathbb{R}[0,1]\) is a probability.

- Path formulas:

  \[
  \varphi ::= \Phi A U_{<r}^t \Phi \mid \Phi A_1 U_{<r}^t A_2 \Phi
  \]

  where \(A_1, A_2 \subseteq \text{Act}\) are sets of actions and \(t, r \in \mathbb{R}_{\geq 0}\) are nonnegative reals.
aCSRL: semantics (informally)

- Let $\omega = s_0 \xrightarrow{a_0,t_0} s_1 \xrightarrow{a_1,t_1} \cdots$ be a path (infinite sequence of steps).
- $\omega \models \Phi_1 A U_{<r}^t \Phi_2$: the path visits a state satisfying $\Phi_2$ within $t$ time units, while accumulating at most $r$ reward, and visits states satisfying $\Phi_1$ while performing only actions in $A$ until that point.
- $\omega \models \Phi_1 A_1 U_{<r}^t A_2 \Phi_2$: the path visits a state satisfying $\Phi_2$ within $t$ time units, while accumulating at most $r$ reward, after performing an action in $A_2$, and visits states satisfying $\Phi_1$ while performing only actions in $A_1$ until that point.
aCSRL: semantics (informally)

- $s \models P_{\mathcal{M}}(\varphi)$ iff $\text{Prob}_s^{\mathcal{M}}(\{\omega \in \text{Path}^{\mathcal{M}} \mid \omega \models \varphi\}) \preccurlyeq p$ where $\text{Prob}_s^{\mathcal{M}}(\Omega)$ is the probability of exhibiting paths in $\Omega$ from $s$ in $\mathcal{M}$.

- Example: “with probability at most 0.02, the system exhibits an execution path such that:
  - there is a point along the path at which the atomic proposition $C.fail$ holds directly after performing a $\text{break}$ action;
  - at this point no more than 30 time units have elapsed and at most 7 units of reward have been accumulated,
  - $C.up$ holds at all preceding points,
  - and the action $\text{ok}$ is the only action seen on the path before the action $\text{break}$.”

results in $\mathcal{P}_{\leq 0.02}(C.up \{\text{ok}\} U_{\leq 30} \{\text{break}\} C.fail)$. 
UMSL: combining aCSRL and dnfMSL

• Proposal:
  
  • Define dnfMSL+:
    a “formula” of dnfMSL+ is a pair $(\psi, \Phi)$, where $\psi$ is a dnfMSL formula and $\Phi$ is an aCSRL+ formula.
  
  • Define aCSRL+:
    a “formula” of aCSRL+ is a pair $(\phi, \Psi)$, where $\phi$ is an aCSRL formula and $\Psi$ is a dnfMSL+ formula.
  
  • Can nest aCSRL+ formulas within an aCSRL formula.

• Intuition:
  
  • dnfMSL+ formula $(\psi, \Phi)$:
    defines a reward structure by applying $\psi$ only to states satisfying $\Phi$.
  
  • aCSRL+ formula $(\phi, \Psi)$:
    is satisfied by the states satisfying $\phi$ where the reward structure is given by $\Psi$.

• Details to be worked on...
Measure definition mechanism for UMSL (1)

- **Aim**: to create a set of high-level of performance measures formed by UMSL.
- **Syntax**:
  
  \[
  \text{MEASURE} \triangleleft \text{name} \triangleright (\triangleleft \text{parameters} \triangleright ) \text{ IS } \triangleleft \text{body} \triangleright \\
  \]

  where \text{parameters} refers to component-oriented arguments, and \text{body} contains the UMSL formula.
- **Example**: system throughput with respect to the activities \( C_1.a_1, \ldots, C_N.a_N \):
  
  \[
  \text{MEASURE} \ \text{throughput}(C_1.a_1, \ldots, C_N.a_N) \ \text{IS} \ \left( \psi, (\Phi, \text{true}) \right) \\
  \]

  where \( \Phi = A \) and \( \psi \) is defined by

  \[
  \text{eq} \left( \text{state rew}(s), \text{sum act contrib}(s, A, \text{sum}) \right) \\
  \]

  such that \( A = (C_1.a_1 \lor \ldots \lor C_N.a_N) \) and \( \text{act rew}(a) = \lambda \) whenever \( a = C_i.a_i \) for some \( 1 \leq i \leq N \) and the rate associated with \( a \) is \( \lambda \).
Measure definition mechanism for UMSL (2)

- Example: states (1) not including the behavior $C.B$ and (2) from which $C.B$ becomes true within $t$ time units with at most reward $r$, where throughput is accumulated by $C.a$ actions, with probability $\leq p$:

$$\text{MEASURE } \text{throughput\_until\_beh}(C.B, C.a, p) \text{ IS } (\psi, (\Phi, \psi'))$$

where $\psi$ is defined by

$$eq(state\_rew_\Phi(s), choose\_lstate\_contrib(s, \{-C.B\}, \text{sum}, \text{min}))$$

such that $lstate\_rew(-C.B) = 1$, where:

$$\Phi = (\mathcal{P}_{\leq p}(-C.B \ Act U_{\leq t}^t C.B), \Psi),$$

and where $\psi'$ is defined by:

$$eq(state\_rew(s), sum\_act\_contrib(s, \{a\}, \text{sum}))$$

where $act\_rew_{\text{path}}(a) = \lambda$ whenever the rate associated with $a$ is $\lambda$.

- $state\_rew_\Phi(s)$ is assigned a reward only if $s \models \Phi$, otherwise reward is 0.
Remaining work

- Define formally the syntax and semantics of UMSL.
- Describe the model-checking algorithm of aCSRL (combination of that of aCSL [HKMS00] and CSRL [BHHK00,HCHKB02]).
- Work further on measure definition mechanism (patterns [Grunske09] and nesting).
- Examples.