Compositionality in Real-Time Model Checking

Jasper Berendsen and Frits Vaandrager

Radboud Universiteit Nijmegen

Laquso Lunch Colloquium, 8 november 2007

A Dichotomy

Modeling languages for reactive systems typically either support communication via shared variables or communication via synchronization of actions:

- TLA, Reactive Modules, etc,
- CCS, I/O automata, ACP, mCRL2, etc

Lamport: *"In reality there are only states"*

A Non-Issue?

Both types of communication can be defined in terms of each other:

- A shared variable can be modeled as a separate process/automaton that communicates with its environment via read/write synchronization actions.
- Synchronization of actions can be modeled using some auxiliary flag variables and handshake transitions of the synchronizing automata.

However, these encodings blow up the state space and make it more difficult to understand the model!
We want to have it all!

Refinement

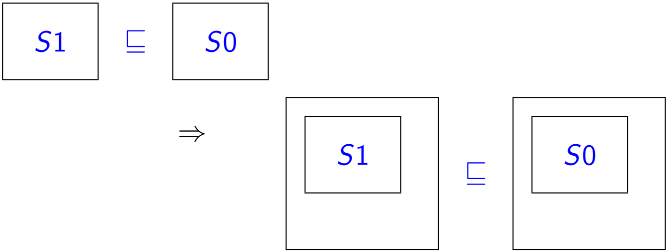
Abstraction/refinement is a key technique to combat state space explosions in model checking.

implementation preorder

$$\cdots \quad \sqsubseteq \boxed{S2} \sqsubseteq \boxed{S1} \sqsubseteq \boxed{S0}$$

Compositionality

Compositional abstraction is even more useful!



Main Problem

Existing approaches for compositional abstractions do not apply to settings with both shared variables and syncronization of actions.

Uppaal

Model checker for timed automata developed originally by universities of Uppaal and Aalborg, with recent contributions by Nijmegen. Many industrial applications!

- Bang & Olufsen protocol, Philips Audio Control
- Biphase Mark protocol
- IEEE 1394 "Firewire", Zeroconf, SHIM6
- scheduling of lacquer production at Axxom
- throughput optimization for a wafer scanner from ASML
- car periphery supervision system from Bosch
- architecture evaluation for a distributed in-car navigation system by Siemens
- mutex and semaphore examples
- ...

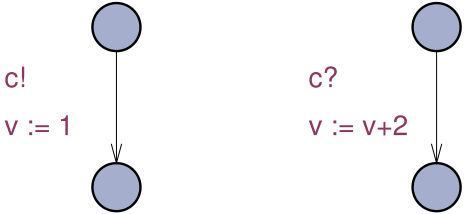
Uppaal Modelling Language

- Recently, Uppaal has been extended with C-like functions and the verification engine has become much more powerful (e.g. due to symmetry reduction).
- Uppaal supports both shared variables and synchronization of actions
- Many other features: committed locations, urgent channels, broadcast communication,..

Our Result

A framework for compositional abstraction for UPPAAL based on simulation relations that does support synchronization of actions, communication via shared variables, and committed locations.

Combining the Two Means for Communication



Proposals:

(a) Rule out syntactically: only one automaton has write permission for each variable;

(b) Rule out semantically: results of both assignments should be the same;

(c) First perform assignment for $c!$, then assignment for $c?$.

Notation

For functions f and g, we let $f \rhd g$ denote the *left-merge*, the combined function where f overrides g for all elements in the intersection of their domains:

$$(f \rhd g)(z) \triangleq \begin{cases} f(z) & \text{if } z \in \mathit{dom}(f) \\ g(z) & \text{if } z \in \mathit{dom}(g) - \mathit{dom}(f) \end{cases}$$

We define the dual *right-merge* operator by $f \lhd g \triangleq g \rhd f$.

Notation (cnt)

Two functions f and g are *compatible*, notation $f \heartsuit g$, if they agree on the intersection of their domains

For compatible functions f and g, we define their *merge* by $f \| g \triangleq f \rhd g$.

We write $f[g]$ for the *update* of function f according to g, that is, $f[g] \triangleq (f \lhd g) \lceil dom(f)$.

Laws

$$
\begin{array}{ll}
(f \rhd g) \rhd h = f \rhd (g \rhd h) & f[g][h] = f[h \rhd g] \\
f \| (g \| h) = (f \| g) \| h & f \heartsuit g \wedge (f \| g) \heartsuit h \Leftrightarrow (f \heartsuit g \wedge f \heartsuit h \wedge g \heartsuit h) \\
f \| g = g \| f & f \heartsuit g \Leftrightarrow g \heartsuit f \\
f \heartsuit g \Leftrightarrow f = f[g] & \\
f \rhd g = f \| g[f] & f \heartsuit g[f] \\
(f \rhd g)[h] = f[h] \rhd g[h] & f \heartsuit g \Rightarrow f[h] \heartsuit g[h]
\end{array}
$$

Actions

We consider labeled transition systems with several types of state transitions, corresponding to different sets of actions.
We assume a set $\mathcal{C}$ of *channels* and let c range over $\mathcal{C}$.
The set of *external actions* is defined as $\mathcal{E} \triangleq \{c!, c? \mid c \in \mathcal{C}\}$.
We assume the existence of a special *internal action* τ.
Finally, we assume a set of *durations* or *time-passage actions*, which are just the nonnegative real numbers in $\mathbb{R}_{\geq 0}$.

Timed Transition Systems

A *timed transition system (TTS)* is a tuple

$$\mathcal{T} = \langle E, H, S, s^0, \longrightarrow^1, \longrightarrow^0 \rangle,$$

where $E, H \subseteq \mathcal{V}$ are disjoint sets of external and internal variables, respectively, $V = E \cup H$, $S \subseteq \mathit{Val}(V)$, and $\langle S, s^0, \mathit{Act}, \longrightarrow^1 \cup \longrightarrow^0 \rangle$ is an LTS.
We write $r \xrightarrow{a,b} s$ if $(r, a, s) \in \longrightarrow^b$. The value b determines whether or not a transition is committed. We often omit b when it equals 0. We write LTS($\mathcal{T}$) to denote the underlying LTS of $\mathcal{T}$.

Axioms for TTS

We require the following axioms, for all $s, t \in S$, $a, a' \in Act$, $b \in \mathbb{B}$, $d \in \mathbb{R}_{\geq 0}$ and $u \in Val(E)$,

$$s \xrightarrow{a,1} \wedge s \xrightarrow{a',b} \;\Rightarrow\; a' \in \mathcal{E} \vee (a' = \tau \wedge b)$$

$$s[u] \in S$$

$$s \xrightarrow{c?,b} \;\Rightarrow\; s[u] \xrightarrow{c?,b}$$

$$s \xrightarrow{d} t \;\Rightarrow\; t = s \oplus d$$

A state s of a TTS is called committed, notation $Comm(s)$, iff it enables an outgoing committed transition.

Rules for Parallel Composition of TTSs

$$\frac{r \xrightarrow{e,b}_i r'}{r \| s \xrightarrow{e,b} r' \triangleright s}$$

$$\frac{r \xrightarrow{c!,b}_i r' \qquad s[r'] \xrightarrow{c?,b'}_j s' \qquad i \neq j \qquad Comm(r) \vee Comm(s) \Rightarrow b \vee b'}{r \| s \xrightarrow{\tau, b \vee b'} r' \triangleleft s'}$$

$$\frac{r \xrightarrow{\tau,b}_i r' \qquad Comm(s) \Rightarrow b}{r \| s \xrightarrow{\tau,b} r' \triangleright s}$$

$$\frac{r \xrightarrow{d}_i r' \qquad s \xrightarrow{d}_j s' \qquad i \neq j}{r \| s \xrightarrow{d} r' \| s'}$$

Timed Step Simulation

Two TTSs $\mathcal{T}_1$ and $\mathcal{T}_2$ are *comparable* if they have the same external variables, that is $E_1 = E_2$. Given comparable TTSs $\mathcal{T}_1$ and $\mathcal{T}_2$, we say that a relation $\mathrm{R} \subseteq S_1 \times S_2$ is a *timed step simulation* from $\mathcal{T}_1$ to $\mathcal{T}_2$, provided that $s_1^0 \,\mathrm{R}\, s_2^0$ and if $s \,\mathrm{R}\, r$ then

1. $s\lceil E_1 = r\lceil E_2$,
2. $\forall u \in \mathit{Val}(E_1) : s[u] \;\; \mathrm{R} \;\; r[u]$,
3. if $\mathit{Comm}(r)$ then $\mathit{Comm}(s)$,
4. if $s \xrightarrow{a,b} s'$ then either there exists an r' such that $r \xrightarrow{a,b} r'$ and $s' \,\mathrm{R}\, r'$, or $a = \tau$ and $s' \,\mathrm{R}\, r$.

We write $\mathcal{T}_1 \preceq \mathcal{T}_2$ when there exists a timed step simulation from $\mathcal{T}_1$ to $\mathcal{T}_2$.

Compositionality

Theorem

Let $\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3$ be TTSs with $\mathcal{T}_1$ and $\mathcal{T}_2$ comparable, $\mathcal{T}_1 \preceq \mathcal{T}_2$, and both $\mathcal{T}_1$ and $\mathcal{T}_2$ compatible with $\mathcal{T}_3$. Then $\mathcal{T}_1 \| \mathcal{T}_3 \preceq \mathcal{T}_2 \| \mathcal{T}_3$.

Consistency with Uppaal Semantics

Theorem

Let $\mathcal{N} = \langle \mathcal{A}_1, \ldots, \mathcal{A}_n \rangle$ be a network of timed automata. Then

$$\mathsf{LTS}(\mathcal{N}) \;\cong\; \mathsf{LTS}((\mathsf{TTS}(\mathcal{A}_1) \| \cdots \| \mathsf{TTS}(\mathcal{A}_n)) \backslash \mathcal{C}).$$