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.

\[
\ldots \subseteq S_2 \subseteq S_1 \subseteq S_0
\]
Compositional abstraction is even more useful!

\[ S_1 \subseteq S_0 \Rightarrow S_1 \subseteq S_0 \]
Main Problem

Existing approaches for compositional abstractions do not apply to settings with both shared variables and synchronization 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
- ...
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 \( f \triangleright g \) denote the left-merge, the combined function where \( f \) overrides \( g \) for all elements in the intersection of their domains:

\[
(f \triangleright g)(z) \triangleq \begin{cases} 
  f(z) & \text{if } z \in \text{dom}(f) \\
  g(z) & \text{if } z \in \text{dom}(g) - \text{dom}(f)
\end{cases}
\]

We define the dual right-merge operator by \( f \triangleleft g \triangleq g \triangleright f \).
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 $f \parallel g \triangleq f \triangleright g$.

We write $f[g]$ for the update of function $f$ according to $g$, that is, $f[g] \triangleq (f \triangleright g) \upharpoonright \text{dom}(f)$. 
Laws

\[(f \triangleright g) \triangleright h = f \triangleright (g \triangleright h)\]
\[f \Vert (g \Vert h) = (f \Vert g) \Vert h\]
\[f \Vert g = g \Vert f\]
\[f \heartsuit g \iff f = f[g]\]
\[f \triangleright g = f \Vert g[f]\]
\[(f \triangleright g)[h] = f[h] \triangleright g[h]\]
\[f[g][h] = f[h \triangleright g]\]
\[f \heartsuit g \land (f \Vert g) \heartsuit h \iff (f \heartsuit g \land f \heartsuit h \land g \heartsuit h)\]
\[f \heartsuit g \iff g \heartsuit f\]
\[f \heartsuit g[f]\]
\[f \heartsuit g \Rightarrow f[h] \heartsuit g[h]\]
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 $\tau$. Finally, we assume a set of durations or time-passage actions, which are just the nonnegative real numbers in $\mathbb{R}_{\geq 0}$. 
A timed transition system (TTS) is a tuple

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

where $E, H \subseteq \mathcal{V}$ are disjoint sets of external and internal variables, respectively, $\mathcal{V} = E \cup H$, $S \subseteq \text{Val}(\mathcal{V})$, and $\langle S, s^0, \text{Act}, \rightarrow^1 \cup \rightarrow^0 \rangle$ is an LTS.

We write $r \xrightarrow{a,b} s$ if $(r, a, s) \in \rightarrow^b$. The value $b$ determines whether or not a transition is committed. We often omit $b$ when it equals 0. We write $\text{LTS}(\mathcal{T})$ to denote the underlying LTS of $\mathcal{T}$. 

Axioms for TTS

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

\[
\begin{align*}
  s \xrightarrow{a,1} \land s \xrightarrow{a',b} & \implies a' \in E \lor (a' = \tau \land b) \\
  s[u] & \in S \\
  s \xrightarrow{c?,b} & \implies s[u] \xrightarrow{c?,b} \\
  s \xrightarrow{d} t & \implies t = s \oplus d
\end{align*}
\]

A state $s$ of a TTS is called committed, notation $\text{Comm}(s)$, iff it enables an outgoing committed transition.
Rules for Parallel Composition of TTSs

\[
\begin{align*}
 r^{e,b} & \xrightarrow{i} r' \\
 r \parallel s & \xrightarrow{e,b} r' \triangleright s
\end{align*}
\]

\[
\begin{align*}
 r^{\tau,b} & \xrightarrow{i} r' \quad \text{Comm}(s) \Rightarrow b \\
 r \parallel s & \xrightarrow{\tau,b} r' \triangleright s
\end{align*}
\]

\[
\begin{align*}
 r^{c!,b} & \xrightarrow{i} r' \\
 s[r'] & \xrightarrow{c?,b'} \xrightarrow{j} s' \quad i \neq j \\
 \text{Comm}(r) \lor \text{Comm}(s) & \Rightarrow b \lor b'
\end{align*}
\]

\[
\begin{align*}
 r^{\tau,b \lor b'} & \xrightarrow{i} r' \triangleleft s' \\
 r \parallel s & \xrightarrow{\tau,b \lor b'} \xrightarrow{r' \triangleleft s'}
\end{align*}
\]

\[
\begin{align*}
 r^{d} & \xrightarrow{i} r' \\
 s & \xrightarrow{d} s' \quad i \neq j \\
 r \parallel s & \xrightarrow{d} r' \parallel s'
\end{align*}
\]
Timed Step Simulation

Two TTSs $T_1$ and $T_2$ are comparable if they have the same external variables, that is $E_1 = E_2$. Given comparable TTSs $T_1$ and $T_2$, we say that a relation $R \subseteq S_1 \times S_2$ is a timed step simulation from $T_1$ to $T_2$, provided that $s_1^0 R s_2^0$ and if $s R r$ then

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

We write $T_1 \preceq T_2$ when there exists a timed step simulation from $T_1$ to $T_2$. 

Jasper Berendsen and Vaandrager
Compositionality in Real-Time Model Checking
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 $\mathcal{T}_1 \parallel \mathcal{T}_3 \preceq \mathcal{T}_2 \parallel \mathcal{T}_3$. 
Theorem

Let $\mathcal{N} = \langle A_1, \ldots, A_n \rangle$ be a network of timed automata. Then

$$\text{LTS}(\mathcal{N}) \equiv \text{LTS}((\text{TTS}(A_1) \parallel \cdots \parallel \text{TTS}(A_n))\setminus \mathcal{C}).$$