並行分散計算特論 (13)

Shoji Yuen

2012/01/10

Property of region automata

Run of $\mathcal{R}(A)$

$$r : \langle s_0, \alpha_0 \rangle \xrightarrow{a_1} \langle s_1, \alpha_1 \rangle \xrightarrow{a_2} \langle s_2, \alpha_2 \rangle \xrightarrow{a_3} \cdots$$

where $\alpha_i = \{\nu | \nu \models R_i\}$. R_i: Region

A run is *progressive* if $\forall x \in C$, there are infinitely many $i \geq 0$ such that α_i satisfies $[(x = 0) \vee (x > c_x)]$

For a progressive r in $\mathcal{R}(A)$, there exists a correspoinding run in A.

Theorem
There exists a progressive r in $\mathcal{R}(A)$ iff there exists a run in A.

Property of region automata

Run of $\mathcal{R}(A)$

$$r : \langle s_0, \alpha_0 \rangle \xrightarrow{a_1} \langle s_1, \alpha_1 \rangle \xrightarrow{a_2} \langle s_2, \alpha_2 \rangle \xrightarrow{a_3} \cdots$$

where $\alpha_i = \{\nu | \nu \models R_i\}$. R_i: Region

A run is *progressive* if $\forall x \in C$, there are infinitely many $i \geq 0$ such that α_i satisfies $[(x = 0) \vee (x > c_x)]$

For a progressive r in $\mathcal{R}(A)$, there exists a correspoinding run in A.

Theorem
There exists a progressive r in $\mathcal{R}(A)$ iff there exists a run in A.

An accepting run is progressive.

Property of region automata

Run of $\mathcal{R}(A)$

$$r : \langle s_0, \alpha_0 \rangle \xrightarrow{a_1} \langle s_1, \alpha_1 \rangle \xrightarrow{a_2} \langle s_2, \alpha_2 \rangle \xrightarrow{a_3} \cdots$$

where $\alpha_i = \{\nu | \nu \models R_i\}$. R_i: Region

A run is *progressive* if $\forall x \in C$, there are infinitely many $i \geq 0$ such that α_i satisfies $[(x = 0) \vee (x > c_x)]$

For a progressive r in $\mathcal{R}(A)$, there exists a correspoinding run in A.

Theorem
There exists a progressive r in $\mathcal{R}(A)$ iff there exists a run in A.

An accepting run is progressive.

$L_T(A) = \emptyset$ if $L(\mathcal{R}(A)) = \emptyset$

Untiming construction

Proposition

Given a TBA $A = \langle \Sigma, S, S_0, C, E, F \rangle$, there exists an (untimed) BA that accepts $Untime(L_T(A))$.

$Untime(w) = \sigma$ where $w = (\sigma, \tau)$

Let $\mathcal{R}(A) = \langle \Sigma, S_R, S_{R0}, E_R \rangle$ and $F_R = \{(s, \alpha) | s \in F\}$
The untiemd BA=$\langle \mathcal{R}(A), F_R \rangle$

For $(\sigma, \tau) \in L_T(A)$, there exists a run r over (σ, τ) where some $s \in F$ is infinitely many visted.
Then, there exists a progressive run over σ in $\mathcal{R}(A)$.
$F'' = F_R \cap \cup_{x \in C} \{(s, \alpha) | \alpha \models [x = 0 \vee x > c_x]\}$

Checking emptiness

Theorem

Given a TBA $A = \langle \Sigma, S, S_0, C, E, F \rangle$, $L(A) = \emptyset$ is checked in time $O[(|S| + |E|) \times 2^{|\delta(A)|}]$.

Theorem

The emptiness checking problem is PSPACE-complete.

Universality

$L_T(A) \stackrel{?}{=} \{(\sigma, \tau) | \sigma \in \Sigma^\omega, \tau \text{ is a time sequence}\}$

Theorem

The universality problem is undecidable.

Additions in constraints

$$
\begin{array}{rcl}
\delta^+ & ::= & x \le c \mid c \le x \\
 & & \mid x \le y + c \mid x + c \le y \\
 & & \mid \neg\delta^+ \mid \delta_1^+ \wedge \delta_2^+ \\
T & ::= & x \mid c \mid T_1 + T_2 \\
\delta^* & ::= & T_1 \le T_2 \mid \neg\delta^* \mid \delta_1^* \wedge \delta_2^*
\end{array}
$$

Proposition

The emptiness problem with constraints of δ^* is undecidable.

Equivalences

$$A_1 \sim_1 A_2 \quad \text{iff} \quad L(A_1) = L(A_2)$$

$$A_1 \sim_2 A_2 \quad \text{iff} \quad \forall A. L(A) \cap L(A_1) = \emptyset \Leftrightarrow L(A) \cap L(A_2) = \emptyset$$

Proposition

$\sim_1$ *is undecidable.*

$\sim_2$ *is decidable. co-re complete.*

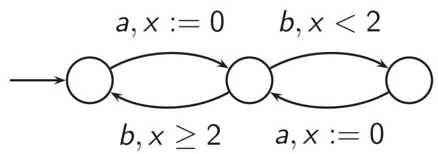
Deterministic TA

Definition

$\langle \Sigma, S, S_0, C, E \rangle$ *is* deterministic *if:*

$|S_0| = 1$

$\forall s \in S. e_1, e_2 \in E, e_1 \neq e_2$ *implies* $\delta_1 \wedge \delta_2$ *is unsatisfiable where* δ_i *is the time constraint of* e_i



Closure property of DTA

Proposition

DTMA(Deterministic Timed Müller Automata) is closed under union, intersection, *and* complementation.

$$\text{DTMA } A = \langle \Sigma, S, S_0, C, E, \mathcal{F} \rangle$$
$$A^c = \langle \Sigma, S, S_0, C, E, Loop(S, E) - \mathcal{F} \rangle$$

Proposition

DTBA(Deterministic Timed Büchi Automata) is closed under union *and* intersection. *The complementation of DTBA is accepted by some DTMA.*

Decision procedure for DTA

Theorem

$A_1 \in TA$, $A_2 \in DTA$, $L(A_1) \subseteq L(A_2)$ *is* PSPACE-complete.

Emptiness check for $L(A_1) \cap L(A_2^c)$.

Expressiveness

$$\begin{array}{cl} \text{TMA} = \text{TBA} & \text{closed under union, intersection} \\ \cup & \\ \text{DTMA} & \text{closed under union, intersection, complementation} \\ \cup & \\ \text{DTBA} & \text{closed under union, intersection} \end{array}$$

(untimed) ω-automata

$$\begin{array}{cl} \text{MA=BA=DMA} & \text{closed under union, intersection, complementation} \\ \cup & \\ \text{DBA} & \text{closed under union, intersection} \end{array}$$

Verfication Framework

Given a specification S, A is correct if $L(A) \subseteq S$

S:A timed regular langauge by DTA.

Composition

A set of timed processes (automata)

$\{P_i = (\Sigma_i, L_i)\}_i$

Parallel composition $||_i P_i$:

$$(\cup_i \Sigma_i, \{(\sigma, \tau)| \wedge_i ((\sigma, \tau) \uparrow \Sigma_i \in L_i\})$$

Exponential blowup source

How desperate to precisely check at the formal language level...

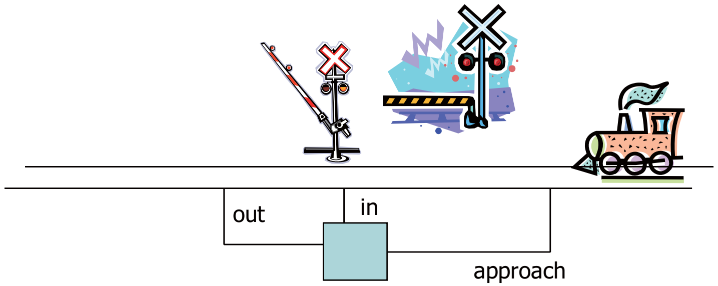
Number of regions is bounded by:

$$O[|A_s \cdot 2^{|\delta(A_s)|} \Pi_i |A_i| \cdot 2^{|\delta(A_i)|}]$$

1. the number of states in the global timed automaton
2. the product of the constatns c_x over all clocks x
3. the number of permutations over all clocks

Example

Train crossing

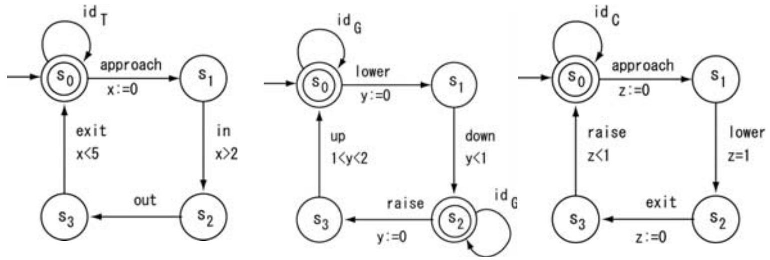


(1) Whenever the train is inside the gate, the gate should be closed.

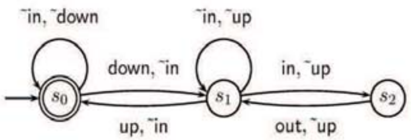
(2) The gate is never closed at a strech for more than 10min.

System description

$[TRAIN \mid\mid GATE \mid\mid CONTROLLER]$

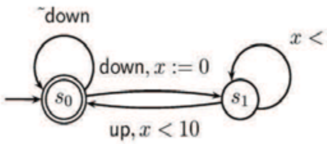


Properties



Safe

$L_T(TRAIN \mid\mid GATE \mid\mid CONTROLLER) \subseteq L_T(Safe)$



Live

$L_T(TRAIN||GATE||CONTROLLER||Live) \neq \emptyset$

More results for Decistion Proc.

Proposition

$L(A_1) \subseteq L(A_2)$ is decidable if A_2 has no ε-labelled switches and either:

(1) A_2 uses only one clock, or

(2) A_2 uses guards involving the constant 0 only

Proposition

The inclusion problem becomes undecidable if one of the following conditions holds:

Two clocks and a one-event alphabet

Two clocks and one non-zero constant in the guards

One location and a one-event alphabet

One location and one non-zero constant in the guards