Exame de *Formal Verification of Critical Applications* 2021-2022

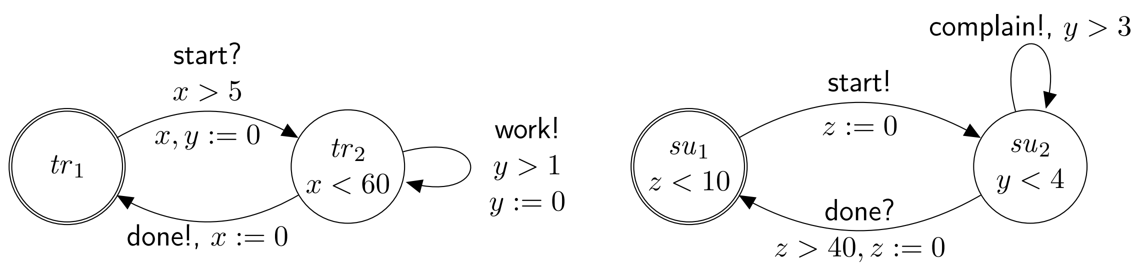
Mestrado em Engenharia de Sistemas Computacionais Críticos

Eduardo Tovar & David Pereira & José Proença

22 Julho 2022 (época de recurso) – duração: 2h

Modelação de sistemas de tempo real

Exercício 1. Considere a rede de autómatos de tempo real abaixo, que representa um *trabalhador* (esquerda) e um *supervisor* (direita) em paralelo.



1.1. Desenhe um único autómato *"trab-supervisionado"* equivalente ao produto dos dois autómatos dados.

1.2. Para cada um dos 3 autómatos (*"trabalhador"*, e *"trab-supervisionado"*), indique:

- Se tem algum comportamento **Zeno** (e se sim, indique o caminho, se não, explique informalmente).
- Se tem algum **timelock** (e se sim, indique o caminho, se não, explique informalmente).

Exercício 2. Dê exemplos de 2 autómatos de tempo real que sejam *timed-traced equivalent* mas não sejam *timed bisimilar*, e explique quais os seus traços.
(Caso não encontre 2 autómatos nestas condições, apresente um par de autómatos diferentes mas que sejam timed-bisimilar, e mostre a bisimulação que o demonstra – penalização de 50%.)

Exercício 3. Considere um sistema com 2 autómatos de tempo real em paralelo, *ArCond* e *Termostato*, com estados $\{Aquecer, Arrefecer, Desligado\}$ e $\{Quente, Bom, Frio\}$, respectivamente. Assuma ainda que o *ArCond* tem um relógio c que é colocado a zero de cada vez que o sistema chega aos estados $Aquecer$ e $Arrefecer$.

3.1. Sem modelar os autómatos, formalize as seguintes propriedades usando lógica temporal (como em UPPAAL).

(a) O *Termostato* não pode estar $Quente$ enquanto o ar condicionado está a $Aquecer$.

(b) O ar condicionado só pode ficar a $Arrefecer$ no máximo 25 unidades de tempo e ficar a $Aquecer$ no máximo 20 unidades de tempo.

3.2. Desenhe um autómato para *ArCond* sem comportamentos Zeno nem *timelocks* que obedeça às 3 propriedades abaixo.

descrita em 3.1(b) | `E<>` $Arrefecer$ | `A[] not` $Aquecer$

Lógica e verificação dedutiva

Exercício 4. Considere os triplos de Hoare apresentados abaixo e, para cada um deles, calcule a respetiva pré-condição mais fraca usando o algoritmo introduzido nas aulas. Mostre que a pré-condição mais fraca é satisfeita pela pré-condição explicitada no triplo.

1. $\{true\}$ if(x > 0) then {y := x;} else {y := -x;} $\{y \geq 0\}$
2. $\{x + y > 4\}$ x := x + 1; y := x + y; $\{y > 5\}$

Exercício 5. Considerando os mesmo triplos de Hoare apresentados na questão anterior, aplique um dos algoritmos de geração de obrigações de demonstração/prova (introduzidos nas aulas como VC e VCG) a cada um desses triplos. **Nota:** No anexo D podem encontrar um dos algoritmos introduzidos na disciplina. A utilização desse algoritmo para efeitos da resolução desta questão será sujeita a uma penalização de 25%; a utilização do algoritmo alternativo não será sujeita a qualquer penalização.

Exercício 6. Considere o triplo de Hoare apresentado abaixo.

```
{i = 7 ∧ a = 0}
while(i > 0) {
 a := a + k;
 i := i - 1;
}
{a = 7×k}
```

Da lista de opções apresentada abaixo, indique aquela que é uma invariante válida e que garante a correção do triplo. Justifique (pode justificar informalmente ou através da aplicação de um dos algoritmos para a geração de condições de verificação VC ou VCG).

1. $i \geq 0 \wedge k \geq 0$
2. $a = (7-i) \times k \wedge i \geq 0$
3. $a = 7\times k \wedge i \geq 0$

A Anexo: Semântica de Timed Automata

Let $ta = \langle L, L_0, Act, C, Tr, Inv\rangle$

$$\mathcal{T}(ta) \;=\; \langle S, S_0 \subseteq S, N, T\rangle$$

where

- $S = \{\langle l, \eta\rangle \in L \times (\mathbb{R}_0^+)^C \mid \eta \models Inv(l)\}$
- $S_0 = \{\langle \ell_0, \eta\rangle \mid \ell_0 \in L_0 \; e \; \eta\, x = 0$ for all $x \in C\}$
- $N = Act \cup \mathbb{R}_0^+$ (ie, transitions can be labelled by actions or delays)
- $T \subseteq S \times N \times S$ is given by:

$$\langle l, \eta\rangle \xrightarrow{a} \langle l', \eta'\rangle \quad \Leftarrow \quad \exists_{l \xrightarrow{g,a,U} l' \in Tr} \;\; \eta \models g \;\wedge\; \eta' = \eta[U] \;\wedge\; \eta' \models Inv(l')$$
$$\langle l, \eta\rangle \xrightarrow{d} \langle l, \eta + d\rangle \quad \Leftarrow \quad \exists_{d \in \mathbb{R}_0^+} \;\; \eta + d \models Inv(l)$$

A *path* of a timed automata ta is a (possibly empty) trace of $\mathcal{T}(ta)$: $\langle \ell_1, \eta_1\rangle \xrightarrow{\alpha_1} \langle \ell_2, \eta_2\rangle \xrightarrow{\cdots} \langle \ell_n, \eta_n\rangle$.

B Anexo: Timed traces and their equivalence

A timed trace over a timed LTS is a (finite or infinite) sequence $\langle t_1, a_1\rangle, \langle t_2, a_2\rangle, \cdots$ in $\mathbb{R}_0^+ \times Act$ such that there exists a path

$$\langle l_0, \eta_0\rangle \xrightarrow{d_1} \langle l_0, \eta_1\rangle \xrightarrow{a_1} \langle l_1, \eta_2\rangle \xrightarrow{d_2} \langle l_1, \eta_3\rangle \xrightarrow{a_2} \cdots$$

such that

$$t_i = t_{i-1} + d_i$$

with $t_0 = 0$ and, for all clock x, $\eta_0\, x = 0$.

Two states s_1 and s_2 of a timed LTS are timed-language equivalent if the set of finite timed traces of s_1 and s_2 coincide.

C Anexo: Timed bisimulation

A relation R is an timed simulation iff whenever $s_1 R s_2$, for any action a and delay $d \in \mathbb{R}_0^+$,

$$s_1 \xrightarrow{a} s_1' \;\Rightarrow\; \text{there is a transition} \quad s_2 \xrightarrow{a} s_2' \;\&\; s_1' R s_2'$$
$$s_1 \xrightarrow{d} s_1' \;\Rightarrow\; \text{there is a transition} \quad s_2 \xrightarrow{d} s_2' \;\&\; s_1' R s_2'$$

And it is an timed bisimulation if its converse is also an untimed simulation.

D Anexo: Geração de condições de demonstração/prova

$$
\begin{array}{rcl}
VC(\{P\}\,skip\,\{Q\}) & = & \{P \rightarrow Q\} \\
VC(\{P\}\,x := E\,\{Q\}) & = & \{P \rightarrow Q[E \mapsto x]\} \\
VC(\{P\}C_1; C_2\{Q\}) & = & VC(\{P\}\,C_1\,\{wprec(C_2; Q)\}) \\
 & & \cup \\
 & & VC(\{wprec(C_2, Q)\}\,C_2\,\{Q\}) \\
VC(\{P\}\,if(B)\,then\,C_1\,else\,C_2\,\{Q\}) & = & VC(\{P \wedge B\}\,C_1\,\{Q\}) \\
 & & \cup \\
 & & VC(\{P \wedge \neg B\}\,C_2\,\{Q\}) \\
VC(\{P\}\,while(B)\,\{I\}\,C\{\}) & = & \{P \rightarrow I, I \wedge \neg B \rightarrow Q\} \\
 & & \cup \\
 & & VC(\{P \wedge \neg B\}\,C\,\{Q\})
\end{array}
$$