Bisimulation

Lecture #22 of Model Checking

Joost-Pieter Katoen

Lehrstuhl 2: Software Modeling & Verification

E-mail: `katoen@cs.rwth-aachen.de`

June 26, 2007

Overview Lecture #22

⇒ Bisimulation equivalence

- Quotient transition system

Implementation relations

- A *binary relation* on transition systems
  - when does a transition systems correctly implements another?
- Important for system *synthesis*
  - stepwise *refinement* of a system specification TS into an "implementation" TS'
- Important for system *analysis*
  - use the implementation relation as a means for *abstraction*
  - replace $TS \models \varphi$ by $TS' \models \varphi$ where $|\,TS'\,| \ll |\,TS\,|$ such that:

$$TS \models \varphi \text{ iff } TS' \models \varphi \quad \text{or} \quad TS' \models \varphi \;\Rightarrow\; TS \models \varphi$$

⇒ Focus on state-based *bisimulation* and *simulation*

  - definition: what is bisimulation?
  - logical characterization: which logical formulas are preserved by bisimulation?

Bisimulation equivalence

Let $\mathit{TS}_i = (S_i, \mathit{Act}_i, \rightarrow_i, I_i, \mathit{AP}, L_i)$, $i{=}1, 2$, be transition systems

A *bisimulation* for $(\mathit{TS}_1, \mathit{TS}_2)$ is a binary relation $\mathcal{R} \subseteq S_1 \times S_2$ such that:

1. $\forall s_1 \in I_1 \, \exists s_2 \in I_2. \, (s_1, s_2) \in \mathcal{R}$ and $\forall s_2 \in I_2 \, \exists s_1 \in I_1. \, (s_1, s_2) \in \mathcal{R}$

2. for all states $s_1 \in S_1$, $s_2 \in S_2$ with $(s_1, s_2) \in \mathcal{R}$ it holds:

   (a) $L_1(s_1) = L_2(s_2)$

   (b) if $s_1' \in \mathit{Post}(s_1)$ then there exists $s_2' \in \mathit{Post}(s_2)$ with $(s_1', s_2') \in \mathcal{R}$

   (c) if $s_2' \in \mathit{Post}(s_2)$ then there exists $s_1' \in \mathit{Post}(s_1)$ with $(s_1', s_2') \in \mathcal{R}$

TS_1 and TS_2 are bisimilar, denoted $\mathit{TS}_1 \sim \mathit{TS}_2$, if there exists a bisimulation for $(\mathit{TS}_1, \mathit{TS}_2)$

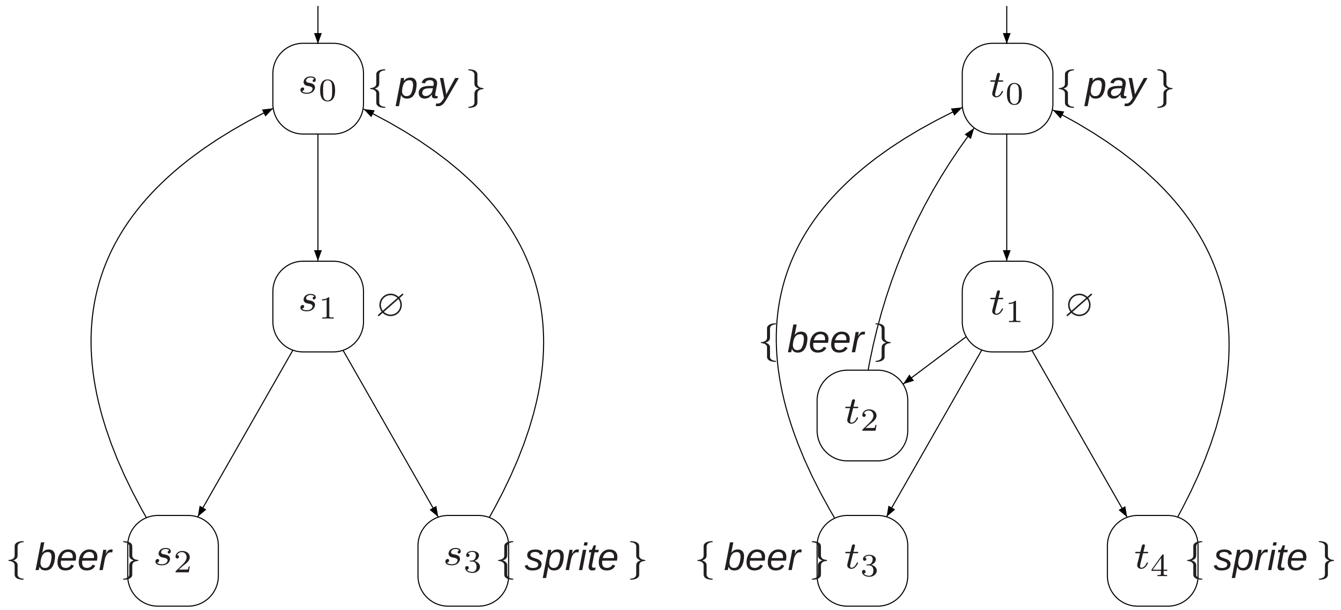
Bisimulation equivalence

$$\begin{array}{ccc} s_1 & \longrightarrow & s_1' \\ \mathcal{R} & & \\ s_2 & & \end{array} \quad \text{can be completed to} \quad \begin{array}{ccc} s_1 & \longrightarrow & s_1' \\ \mathcal{R} & & \color{red}{\mathcal{R}} \\ s_2 & \color{red}{\longrightarrow} & \color{red}{s_2'} \end{array}$$

and

$$\begin{array}{ccc} s_1 & & \\ \mathcal{R} & & \\ s_2 & \longrightarrow & s_2' \end{array} \quad \text{can be completed to} \quad \begin{array}{ccc} s_1 & \color{red}{\longrightarrow} & \color{red}{s_1'} \\ \mathcal{R} & & \color{red}{\mathcal{R}} \\ s_2 & \longrightarrow & s_2' \end{array}$$

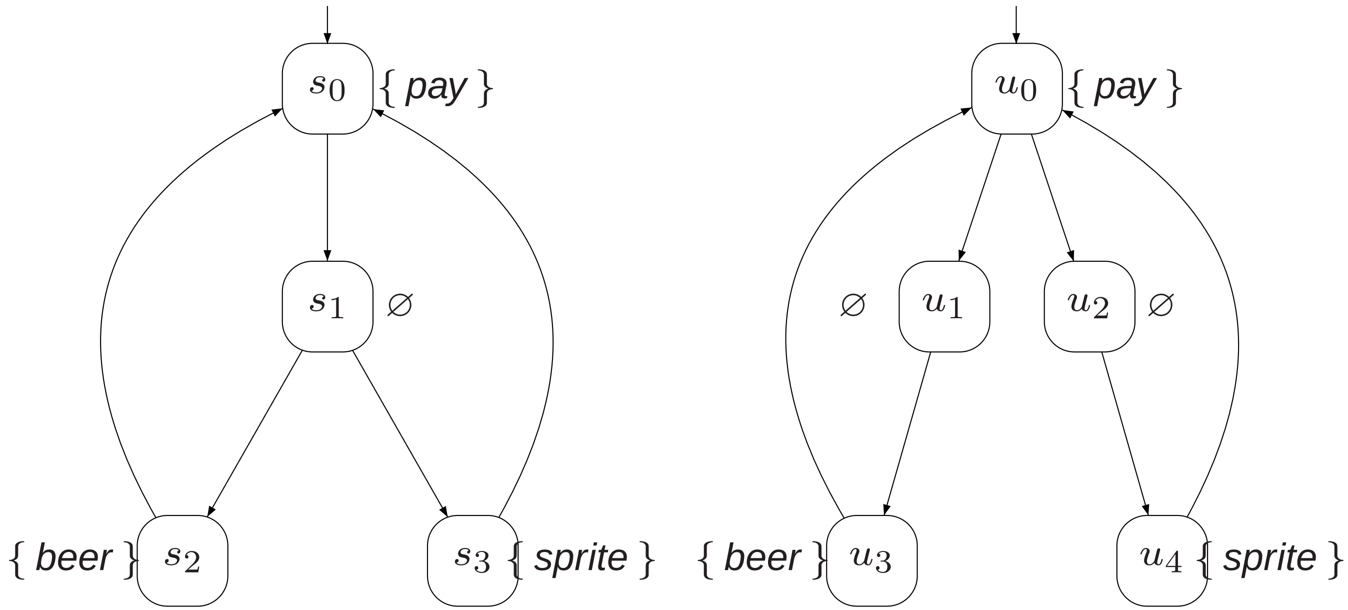
Example (1)



$$\mathcal{R} = \Big\{ (s_0, t_0), (s_1, t_1), (s_2, t_2), (s_2, t_3), (s_3, t_4) \Big\}$$

is a bisimulation for (TS_1, TS_2) where $AP = \{ pay, beer, sprite \}$

Example (2)



$TS_1 \not\sim TS_3$ for $AP = \{ \textit{pay}, \textit{beer}, \textit{sprite} \}$

But: $\{ (s_0, u_0), (s_1, u_1), (s_1, u_2), (s_2, u_3), (s_2, u_4), (s_3, u_3), (s_3, u_4) \}$

is a bisimulation for (TS_1, TS_3) for $AP = \{ \textit{pay}, \textit{drink} \}$

$\sim$ is an equivalence

For any transition systems *TS*, TS_1, TS_2 and TS_3 over *AP*:

$TS \sim TS$ (reflexivity)

$TS_1 \sim TS_2$ implies $TS_2 \sim TS_1$ (symmetry)

$TS_1 \sim TS_2$ and $TS_2 \sim TS_3$ implies $TS_1 \sim TS_3$ (transitivity)

Bisimulation on paths

Whenever we have:

$$\begin{array}{ccccccccc} s_0 & \longrightarrow & s_1 & \longrightarrow & s_2 & \longrightarrow & s_3 & \longrightarrow & s_4 \ldots\ldots \\ \mathcal{R} & & & & & & & & \\ t_0 & & & & & & & & \end{array}$$

this can be completed to

$$\begin{array}{ccccccccc} s_0 & \longrightarrow & s_1 & \longrightarrow & s_2 & \longrightarrow & s_3 & \longrightarrow & s_4 \ldots\ldots \\ \mathcal{R} & & \mathcal{R} & & \mathcal{R} & & \mathcal{R} & & \mathcal{R} \\ t_0 & \longrightarrow & t_1 & \longrightarrow & t_2 & \longrightarrow & t_3 & \longrightarrow & t_4 \ldots\ldots \end{array}$$

proof: by induction on index i of state s_i

Bisimulation vs. trace equivalence

$$TS_1 \sim TS_2 \quad \text{implies} \quad Traces(TS_1) = Traces(TS_2)$$

bisimilar transition systems thus satisfy the same LT properties!

Overview Lecture #22

- Bisimulation equivalence

⇒ Quotient transition system

Bisimulation on states

$\mathcal{R} \subseteq S \times S$ is a *bisimulation* on *TS* if for any $(s_1, s_2) \in \mathcal{R}$:

- $L(s_1) = L(s_2)$
- if $s_1' \in \textit{Post}(s_1)$ then there exists an $s_2' \in \textit{Post}(s_2)$ with $(s_1', s_2') \in \mathcal{R}$
- if $s_2' \in \textit{Post}(s_2)$ then there exists an $s_1' \in \textit{Post}(s_1)$ with $(s_1', s_2') \in \mathcal{R}$

s_1 and s_2 are *bisimilar*, $s_1 \sim_{\textit{TS}} s_2$, if $(s_1, s_2) \in \mathcal{R}$ for some bisimulation $\mathcal{R}$ for *TS*

$s_1 \;\sim_{\textit{TS}}\; s_2$ if and only if $\textit{TS}_{s_1} \;\sim\; \textit{TS}_{s_2}$

Coarsest bisimulation

$\sim_{TS}$ is an equivalence and the coarsest bisimulation for TS

Quotient transition system

For *TS* $= (S, Act, \rightarrow, I, AP, L)$ and bisimulation $\sim_{TS} \subseteq S \times S$ on *TS* let

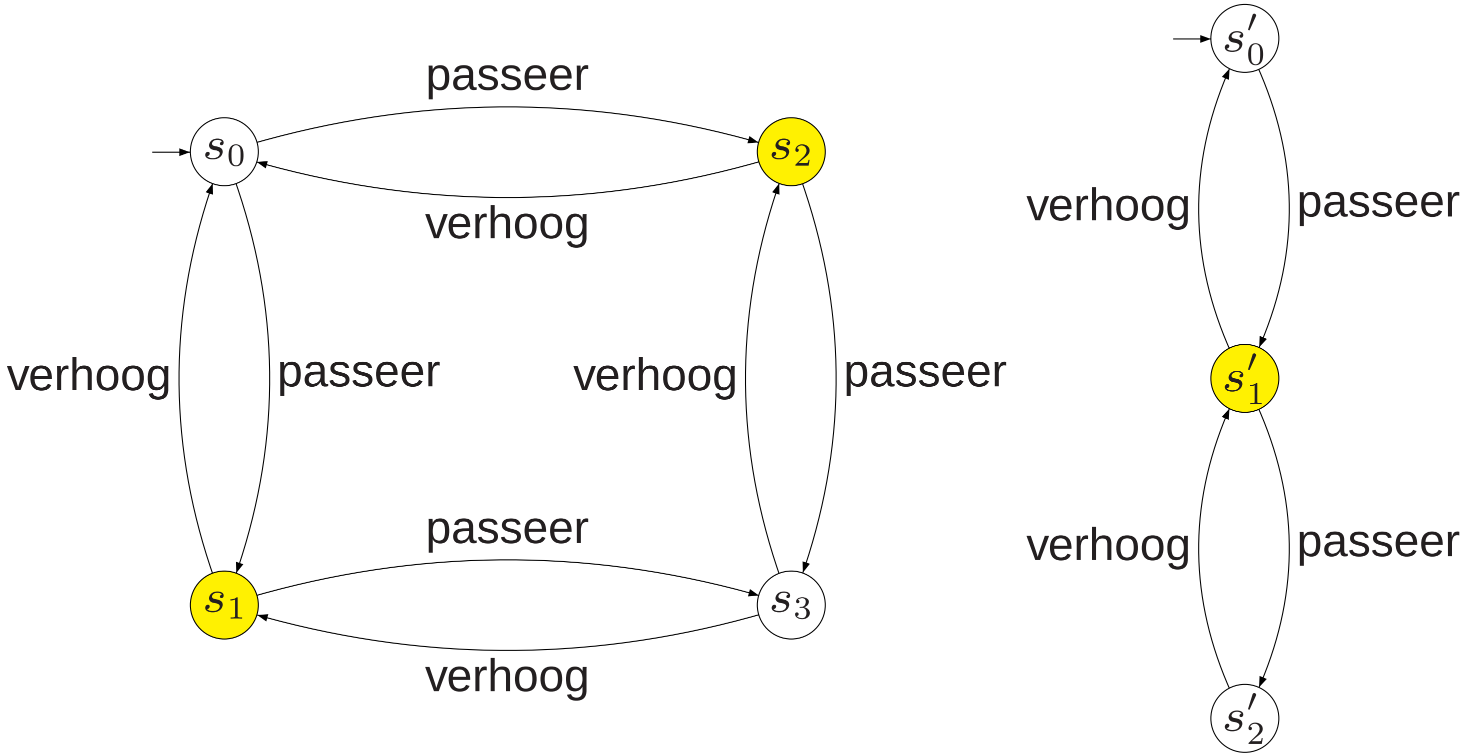
$$TS/\sim_{TS} = (S', \{\tau\}, \rightarrow', I', AP, L'), \quad \text{the } \textit{quotient} \text{ of } TS \text{ under } \sim_{TS}$$

where

- $S' = S/\sim_{TS} = \{ [s]_\sim \mid s \in S \}$ with $[s]_\sim = \{ s' \in S \mid s \sim_{TS} s' \}$
- $\rightarrow'$ is defined by: $$\frac{s \xrightarrow{\alpha} s'}{[s]_\sim \xrightarrow{\tau}{}' [s']_\sim}$$
- $I' = \{ [s]_\sim \mid s \in I \}$
- $L'([s]_\sim) = L(s)$

note that $TS \sim TS/\sim_{TS}$ Why?

A ternary semaphore and its quotient



The Bakery algorithm

Process 1:

$\ldots\ldots$

while true $\{$

$\ldots\ldots$

n_1 : $x_1 := x_2 + 1;$

w_1 : $\textbf{wait until}(x_2 = 0 \,||\, x_1 < x_2)\,\{$

c_1 : $\ldots$ critical section $\ldots\}$

$x_1 := 0;$

$\ldots\ldots$

$\}$

Process 2:

$\ldots\ldots$

while true $\{$

$\ldots\ldots$

n_2 : $x_2 := x_1 + 1;$

w_2 : $\textbf{wait until}(x_1 = 0 \,||\, x_2 < x_1)\,\{$

c_2 : $\ldots$ critical section $\ldots\}$

$x_2 := 0;$

$\ldots\ldots$

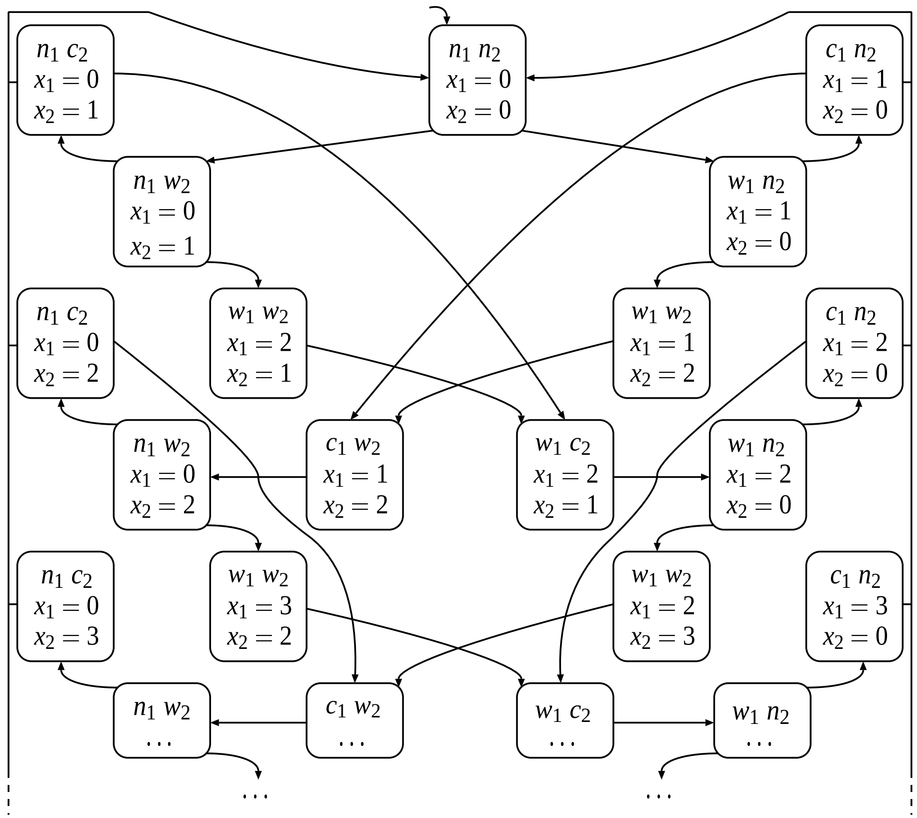
$\}$

this algorithm can be applied to arbitrary many processes

Example path fragment

process P_1	process P_2	x_1	x_2	effect
n_1	n_2	0	0	P_1 requests access to critical section
w_1	n_2	1	0	P_2 requests access to critical section
w_1	w_2	1	2	P_1 enters the critical section
c_1	w_2	1	2	P_1 leaves the critical section
n_1	w_2	0	2	P_1 requests access to critical section
w_1	w_2	3	2	P_2 enters the critical section
w_1	c_2	3	2	P_2 leaves the critical section
w_1	n_2	3	0	P_2 requests access to critical section
w_1	w_2	3	4	P_2 enters the critical section
...	...	..	..	...

Bakery algorithm transition system



infinite state space due to possible unbounded increase of counters

Data abstraction

Function f maps a reachable state of TS_{Bak} onto an abstract one in TS_{Bak}^{abs}

Let $s = \langle \ell_1, \ell_2, x_1 = b_1, x_2 = b_2 \rangle$ be a state of TS_{Bak} with $\ell_i \in \{ n_i, w_i, c_i \}$ and $b_i \in \mathbb{N}$

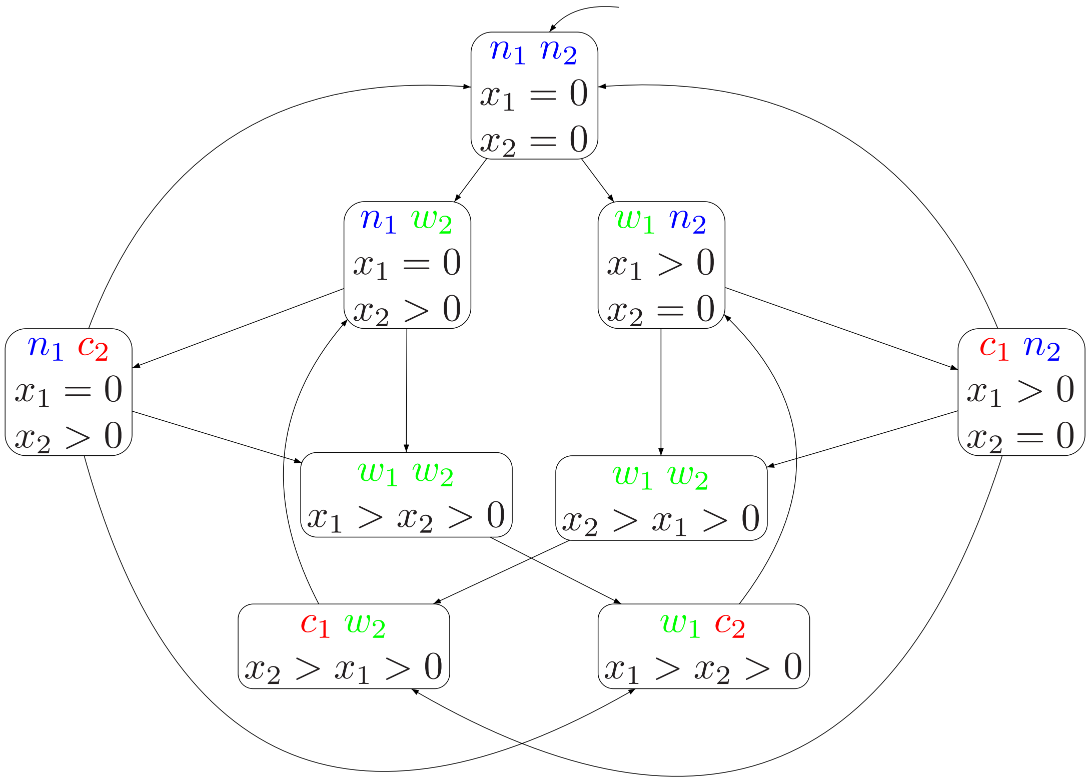
Then:

$$f(s) = \begin{cases} \langle \ell_1, \ell_2, x_1 = 0, x_2 = 0 \rangle & \text{if } b_1 = b_2 = 0 \\ \langle \ell_1, \ell_2, x_1 = 0, x_2 > 0 \rangle & \text{if } b_1 = 0 \text{ and } b_2 > 0 \\ \langle \ell_1, \ell_2, x_1 > 0, x_2 = 0 \rangle & \text{if } b_1 > 0 \text{ and } b_2 = 0 \\ \langle \ell_1, \ell_2, x_1 > x_2 > 0 \rangle & \text{if } b_1 > b_2 > 0 \\ \langle \ell_1, \ell_2, x_2 > x_1 > 0 \rangle & \text{if } b_2 > b_1 > 0 \end{cases}$$

It follows: $\mathcal{R} = \{ (s, f(s)) \mid s \in S \}$ is a bisimulation for $(TS_{Bak}, TS_{Bak}^{abs})$

for any subset of $AP = \{ \mathit{noncrit}_i, \mathit{wait}_i, \mathit{crit}_i \mid i = 1, 2 \}$

Bisimulation quotient



$$TS_{Bak}^{abs} \;=\; TS_{Bak} / \sim \quad \text{for} \quad AP = \{\, crit_1, crit_2 \,\}$$

Remarks

- Data abstraction yields a bisimulation relation
  - in this example; typically a simulation relation is obtained
- $TS^{abs}_{Bak} \models \varphi$ with, e.g.,:
  - $\Box(\neg crit_1 \vee \neg crit_2)$ and $(\Box\Diamond wait_1 \Rightarrow \Box\Diamond crit_1) \wedge (\Box\Diamond wait_2 \Rightarrow \Box\Diamond crit_2)$
- Since $TS^{abs}_{Bak} \sim TS_{Bak}$, it follows $TS_{Bak} \models \varphi$
- Note: $Traces(TS^{abs}_{Bak}) = Traces(TS_{Bak})$
  - but checking trace equivalence is PSPACE-complete
  - while checking bisimulation equivalence is in poly-time