Model Checking

Stutter Bisimulation

[Baier & Katoen, Chapter 7.8]

Joost-Pieter Katoen and Tim Quatmann

Software Modeling and Verification Group

RWTH Aachen, SoSe 2022

Overview

Overview

Stutter Equivalence

Definition: stutter step

Transition $s \to s'$ in transition system TS is a stutter step if $L(s) = L(s')$.

Definition: stutter equivalence

Paths π_1 and π_2 are stutter equivalent, denoted $\pi_1 \equiv_{sttrace} \pi_2$ whenever

$$trace(\pi_1) \text{ and } trace(\pi_2) \text{ are both of the form } {A_0}^+ {A_1}^+ {A_2}^+ \ldots$$

for $A_i \subseteq AP$.

For positive integers n_i and m_i:

$$trace(\pi_1) = \underbrace{A_0 \ldots A_0}_{n_0 \text{ times}} \underbrace{A_1 \ldots A_1}_{n_1 \text{ times}} \underbrace{A_2 \ldots A_2}_{n_2 \text{ times}} \ldots$$

$$trace(\pi_2) = \underbrace{A_0 \ldots A_0}_{m_0 \text{ times}} \underbrace{A_1 \ldots A_1}_{m_1 \text{ times}} \underbrace{A_2 \ldots A_2}_{m_2 \text{ times}} \ldots$$

Stutter Trace Equivalence

Definition: stutter trace equivalence

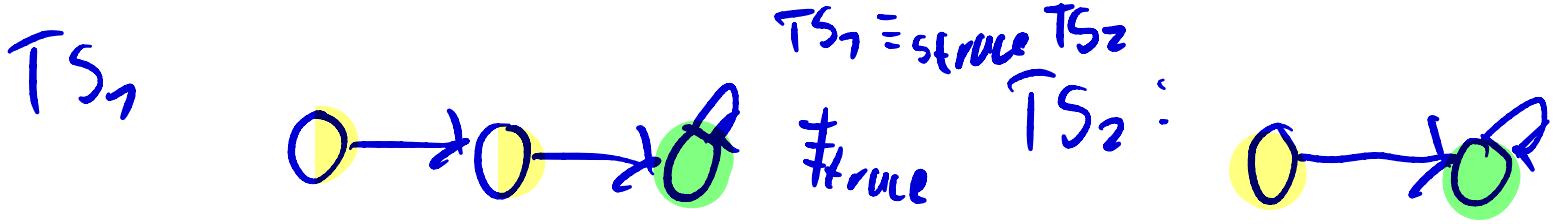
Transition systems TS_i over AP, i=1, 2, are stutter-trace equivalent:

$$TS_1 \equiv_{sttrace} TS_2 \quad \text{if and only if} \quad TS_1 \trianglelefteq TS_2 \text{ and } TS_2 \trianglelefteq TS_1$$

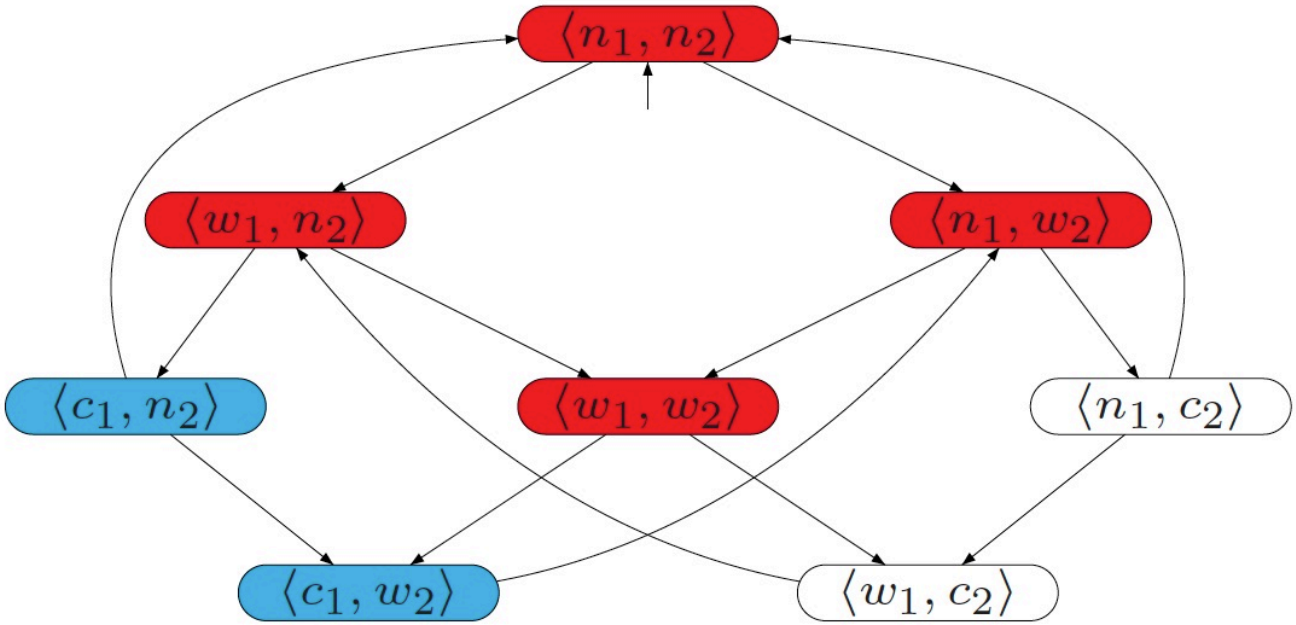
where the stutter trace inclusion relation $\trianglelefteq$ is defined by:

$$TS_1 \trianglelefteq TS_2 \quad \text{iff} \quad \forall \sigma_1 \in Traces(TS_1) \left(\exists \sigma_2 \in Traces(TS_2). \ \sigma_1 \equiv_{sttrace} \sigma_2 \right)$$

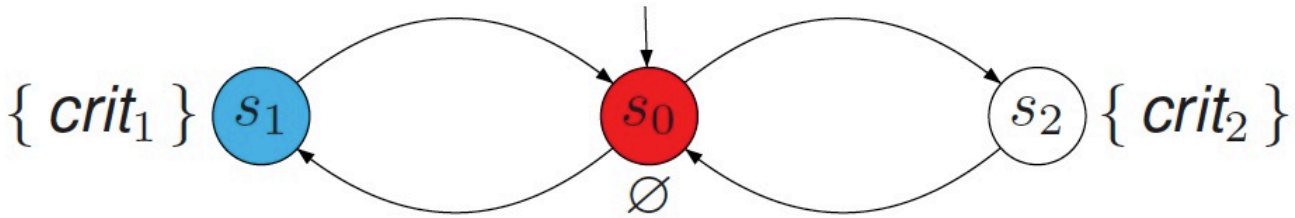
Trace-equivalent transition systems are stutter trace-equivalent, but not the converse.



Example



is stutter equivalent to:



Overview

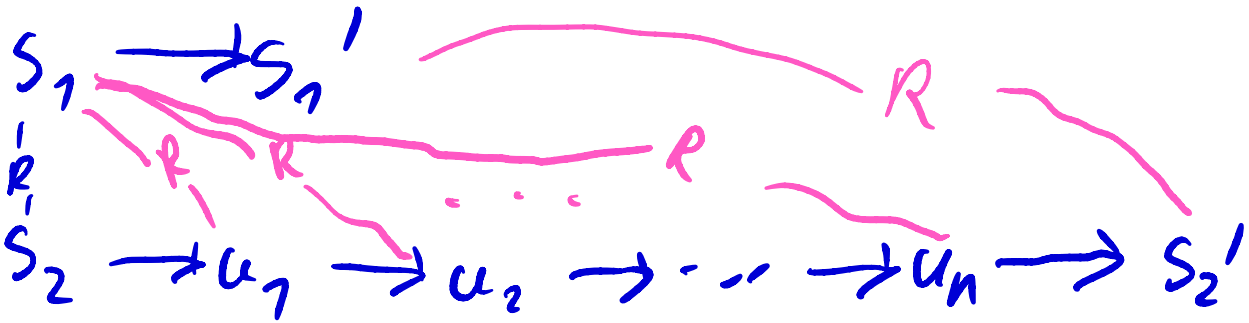
Stutter Bisimulation

Definition: stutter bisimulation

Let $TS = (S, Act, \rightarrow, I, AP, L)$ be a transition system. Symmetric relation $\mathfrak{R} \subseteq S \times S$ is a stutter-bisimulation for TS if for all $(s_1, s_2) \in \mathfrak{R}$:

1. $L(s_1) = L(s_2)$

2. if $s_1' \in Post(s_1)$ with $(s_1', s_2) \notin \mathfrak{R}$, then there exists a finite path fragment $s_2\, u_1 \ldots u_n\, s_2'$ with $n \geq 0$ and $(s_1, u_i) \in \mathfrak{R}$ and $(s_1', s_2') \in \mathfrak{R}$



Stutter Bisimulation

Definition: stutter bisimulation

Let $TS = (S, Act, \rightarrow, I, AP, L)$ be a transition system. Symmetric relation $\mathfrak{R} \subseteq S \times S$ is a stutter-bisimulation for TS if for all $(s_1, s_2) \in \mathfrak{R}$:

1. $L(s_1) = L(s_2)$
2. if $s_1' \in Post(s_1)$ with $(s_1', s_2) \notin \mathfrak{R}$, then there exists a finite path fragment $s_2\, u_1 \ldots u_n\, s_2'$ with $n \geq 0$ and $(s_1, u_i) \in \mathfrak{R}$ and $(s_1', s_2') \in \mathfrak{R}$

Definition: stutter bisimilarity

s_1, s_2 are stutter bisimilar, denoted $s_1 \approx_{TS} s_2$, if $(s_1, s_2) \in \mathfrak{R}$ for some stutter bisimulation $\mathfrak{R}$ for TS. Thus:

$$\approx_{TS} = \bigcup \{ \mathfrak{R} \mid \mathfrak{R} \text{ is a stutter bisimulation for } TS \}$$

Stutter Bisimulation

Visually

$$\begin{array}{ccc} s_1 & \approx & s_2 \\ \downarrow & & \\ s_1' & & \\ \text{(with } s_1' \not\approx s_2\text{)} & & \end{array}$$

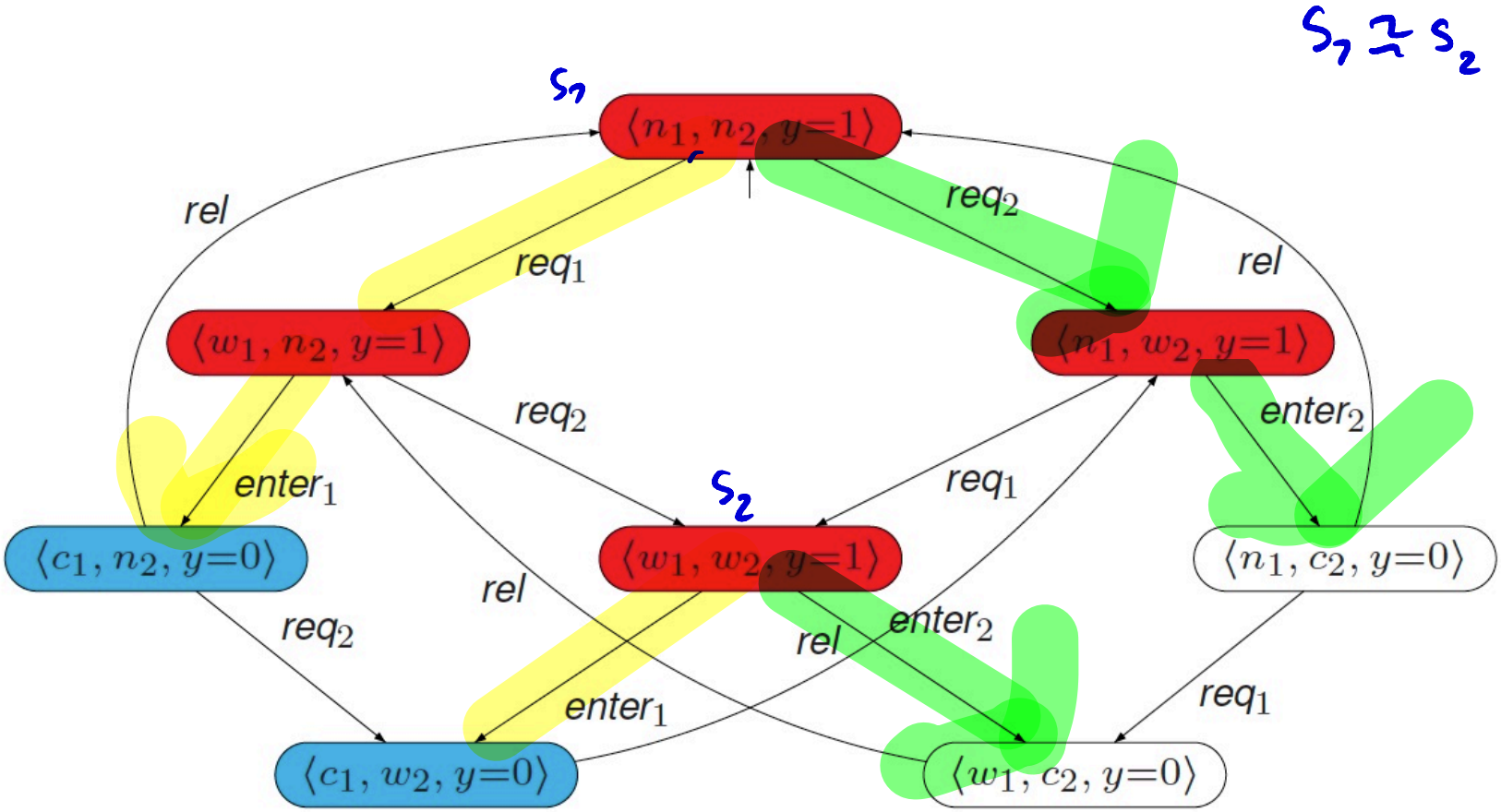
can be completed to

$$\begin{array}{ccc} s_1 & \approx & s_2 \\ & & \downarrow \\ s_1 & \approx & u_1 \\ & & \downarrow \\ s_1 & \approx & u_2 \\ & & \downarrow \\ \vdots & & \vdots \\ & & \downarrow \\ s_1 & \approx & u_n \\ \downarrow & & \downarrow \\ s_1' & \approx & s_2' \end{array}$$

and, by symmetry, the same applies to transitions of s_2

Stutter Bisimulation

Example



stutter-bisimilar states for $AP = \{ crit_1, crit_2 \}$

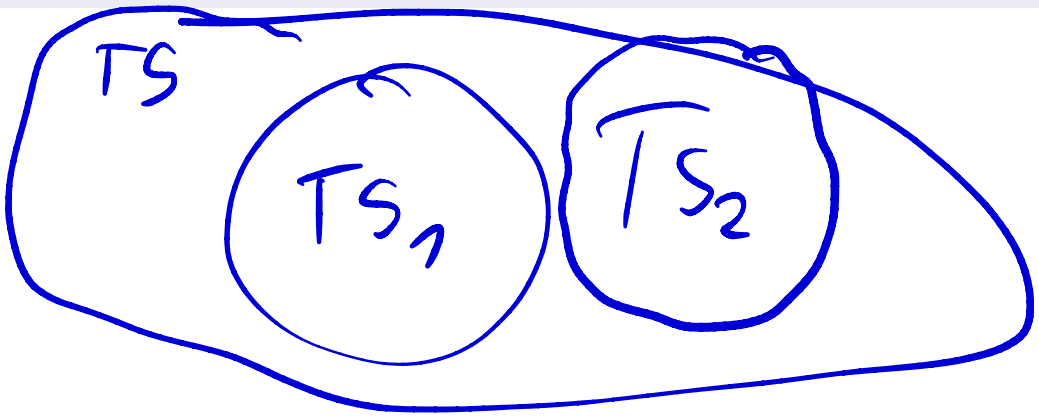
Stutter Bisimilar Transition Systems

Definition: stutter bisimilar transition systems

Let $TS_i = (S_i, Act_i, \rightarrow_i, I_i, AP, L_i)$, $i = 1, 2$, be transition systems.

TS_1 and TS_2 are stutter bisimilar, denoted $TS_1 \approx TS_2$, if there exists a stutter bisimulation $\mathfrak{R}$ on $\underbrace{TS_1 \oplus TS_2}_{\text{disjoint union}}$ such that:

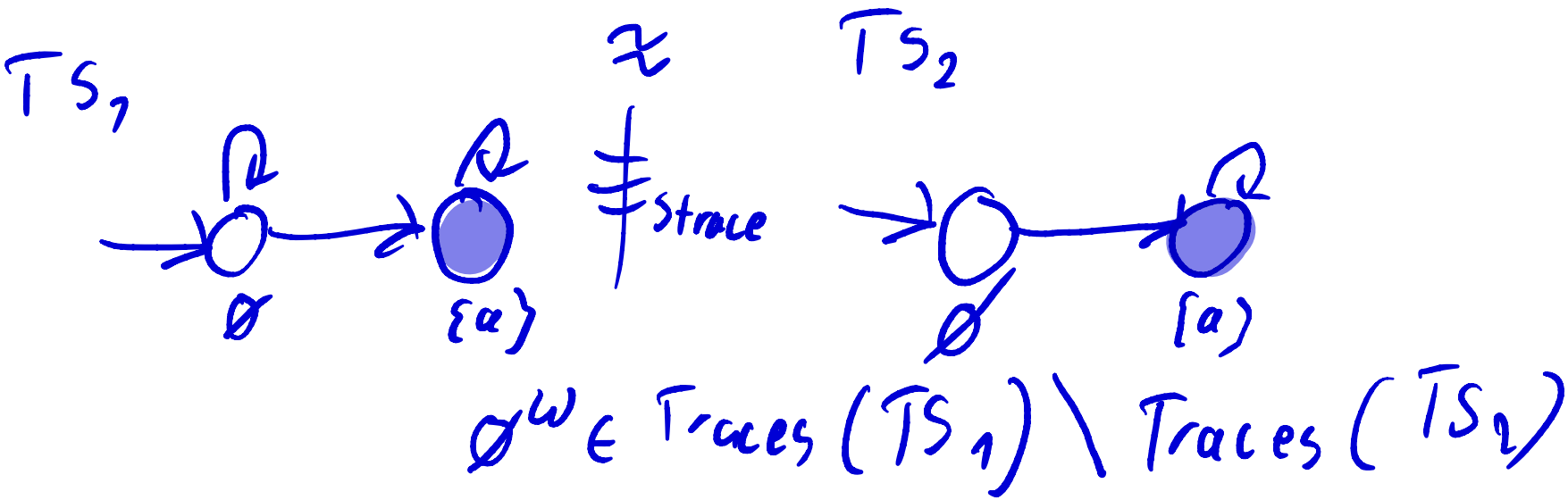
$$\forall s_1 \in I_1 . \big(\exists s_2 \in I_2 . (s_1, s_2) \in \mathfrak{R} \big) \text{ and } \forall s_2 \in I_2 . \big(\exists s_1 \in I_1 . (s_1, s_2) \in \mathfrak{R} \big)$$



Stutter Trace Equivalence and Stutter Bisimilarity

- Known fact: $TS_1 \sim TS_2$ implies $TS_1 \equiv_{trace} TS_2$
  ↳ standard bisim
- But: $TS_1 \approx TS_2$ does not imply $TS_1 \equiv_{sttrace} TS_2$

- Why? Paths that only stutter:
  - stutter bisimulation does not impose any constraint on such paths
  - but $\equiv_{sttrace}$ requires the existence of a stuttering equivalent trace



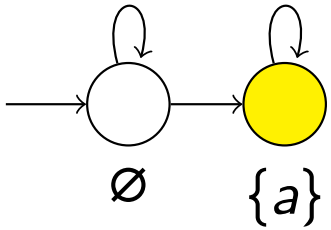
Stutter Trace Equivalence and Stutter Bisimilarity

- Known fact: $TS_1 \sim TS_2$ implies $TS_1 \equiv_{trace} TS_2$
- But: $TS_1 \approx TS_2$ does not imply $TS_1 \equiv_{sttrace} TS_2$
- Why? Paths that only stutter:
  - stutter bisimulation does not impose any constraint on such paths
  - but $\equiv_{sttrace}$ requires the existence of a stuttering equivalent trace
- So:
  - bisimilar transition systems are trace equivalent
  - but stutter-bisimilar transition systems are not always stutter trace-equivalent!

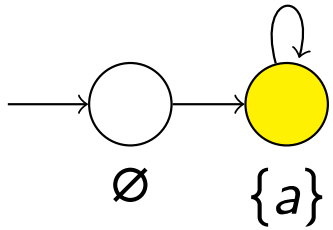
Stutter Bisimulation

Stutter Trace Equivalence and Stutter Bisimilarity

TS_1:



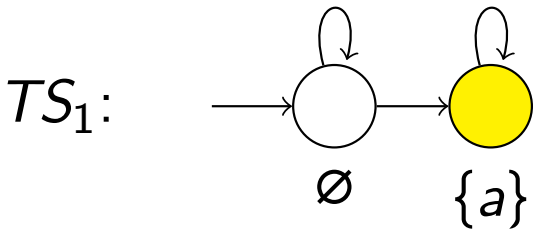
TS_2:

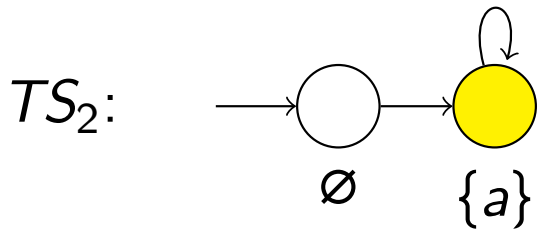


$TS_1 \approx TS_2$ but $TS_1 \not\equiv_{sttrace} TS_2$

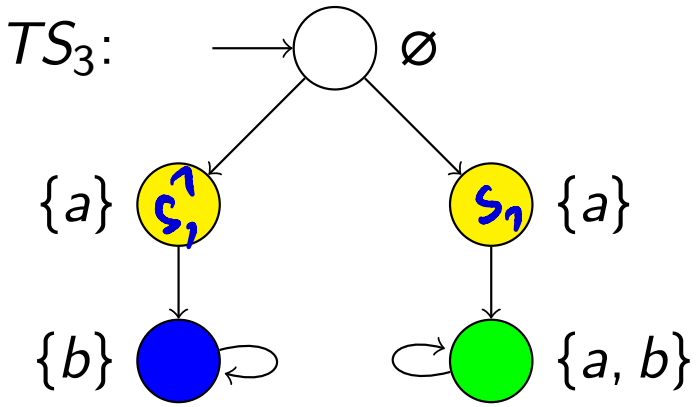
Stutter Bisimulation

Stutter Trace Equivalence and Stutter Bisimilarity



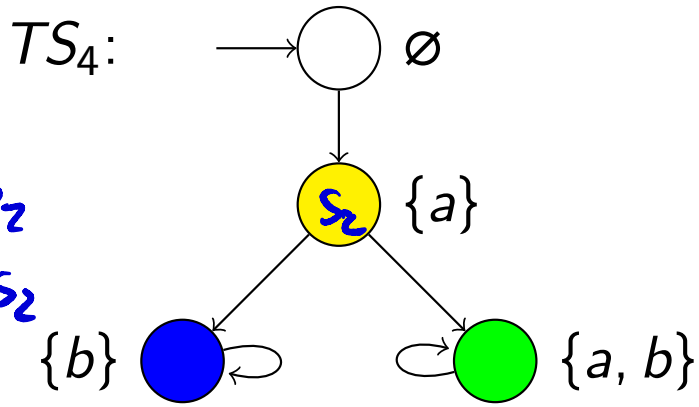


$TS_1 \approx TS_2$ but $TS_1 \not\equiv_{sttrace} TS_2$



$s_1 \not\equiv s_2$

$s_1 \not\equiv s_2$



$TS_3 \equiv_{sttrace} TS_4$ but $TS_3 \not\approx TS_4$

Stutter Bisimilarity Does Not Preserve LTL$_{\setminus\bigcirc}$

$TS_1 \approx TS_2$ but $TS_1 \not\models \Diamond a$ and $TS_2 \models \Diamond a$

Reason: presence of infinite stutter path s_0^ω in TS_1

Overview

Divergence Sensitivity

- Stutter paths are paths that only consist of stutter steps
  - no restrictions are imposed on such paths by a stutter bisimulation
- Stutter paths diverge: they never leave an equivalence class
- Remedy: only relate divergent states or non-divergent states
  - divergent state = a state that has a stutter path
  - relate states only if they either both have stutter paths or none of them
- This yields divergence-sensitive stutter bisimulation ($\approx^{div}$)

  $\Rightarrow$ $\approx^{div}$ is strictly finer than $\equiv_{sttrace}$ (and $\approx$)

Divergence Sensitivity

Definition: divergent paths

Let $\mathfrak{R}$ be an equivalence relation on $S \times S$. State s is $\mathfrak{R}$-divergent if for some infinite path $s\, s_1\, s_2 \ldots \in \mathit{Paths}(s)$ it holds $(s, s_j) \in \mathfrak{R}$ for all $j > 0$.

$\pi =$

i.e. π is $\mathfrak{R}$-divergent

s is $\mathfrak{R}$-divergent if some infinite path from s only visits $[s]_{\mathfrak{R}}$.

Divergence Sensitivity

Definition: divergent paths

Let $\mathfrak{R}$ be an equivalence relation on $S \times S$. State s is $\mathfrak{R}$-divergent if for some infinite path $s\, s_1\, s_2 \ldots \in \mathit{Paths}(s)$ it holds $(s, s_j) \in \mathfrak{R}$ for all $j > 0$.

s is $\mathfrak{R}$-divergent if some infinite path from s only visits $[s]_{\mathfrak{R}}$.

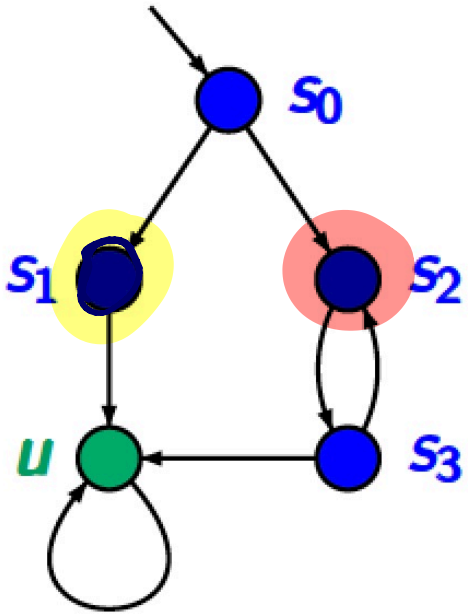
Definition: divergent-sensitive relation

Relation $\mathfrak{R}$ is divergent sensitive if for any $(s_1, s_2) \in \mathfrak{R}$:

s_1 is $\mathfrak{R}$-divergent implies s_2 is $\mathfrak{R}$-divergent.

Divergence-sensitive relations do not pair divergent with non-divergent states.

Examples



● $\widehat{=} \{a\}$

● $\widehat{=} \varnothing$

$AP = \{a\}$

divergent states:
$\{u, s_0, s_2, s_3\}$

R:	$\{s_0\}\ \{s_1\}\ \{s_2, s_3\}\ \{u\}$	divergence-sensitive
R_2:	$\{s_0, s_1, s_2, s_3\}\ \{u\}$	**not** divergence-sensitive
R_3:	$\{s_1\}\ \{s_0, s_2, s_3\}\ \{u\}$	divergence-sensitive
R_4:	$\{s_1\}\ \{s_2\}\ \{s_0, s_3\}\ \{u\}$	divergence sensitive

Divergence-Sensitive Stutter Bisimilarity

s_1, s_2 are divergent-sensitive stutter-bisimilar, denoted $s_1 \approx^{div}_{TS} s_2$, if $(s_1, s_2) \in \mathfrak{R}$ for some $\mathfrak{R}$-divergent-sensitive stutter bisimulation $\mathfrak{R}$ on TS.

$$\approx^{div}_{TS} = \bigcup \{ \mathfrak{R} \mid \mathfrak{R} \text{ is a divergent-sensitive stutter bisimulation for } TS \}$$

① R is a stutter bisim.
and ② R is divergent-sensitive

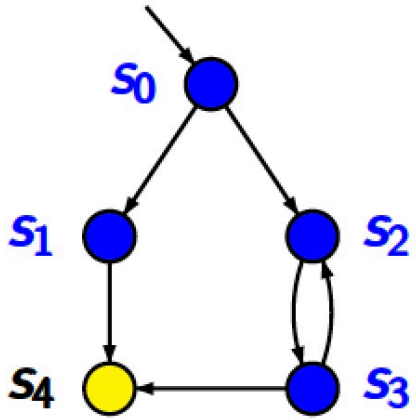
Divergence-Sensitive Stutter Bisimilarity

s_1, s_2 are divergent-sensitive stutter-bisimilar, denoted $s_1 \approx_{TS}^{div} s_2$, if $(s_1, s_2) \in \mathfrak{R}$ for some $\mathfrak{R}$-divergent-sensitive stutter bisimulation $\mathfrak{R}$ on TS.

$$\approx_{TS}^{div} = \bigcup \{ \mathfrak{R} \mid \mathfrak{R} \text{ is a divergent-sensitive stutter bisimulation for } TS \}$$

The relation $\approx_{TS}^{div}$ is an equivalence and the coarsest divergence-sensitive stutter bisimulation for TS.

Example

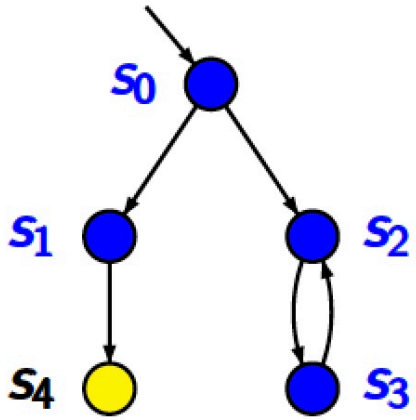


stutter bis. equivalence classes:

$$S/\approx_T = \{\{s_0, s_1, s_2, s_3\}, \{s_4\}\}$$

stutter bis. equiv. classes with **div.**:

$$S/\approx_T^{div} = \{\{s_0\}, \{s_1\}, \{s_2, s_3\}, \{s_4\}\}$$



stutter bis. equivalence classes

$$S/\approx_T = \{\{s_0\}, \{s_1\}, \{s_2, s_3\}, \{s_4\}\}$$

stutter bis. equiv. classes with **div.**:

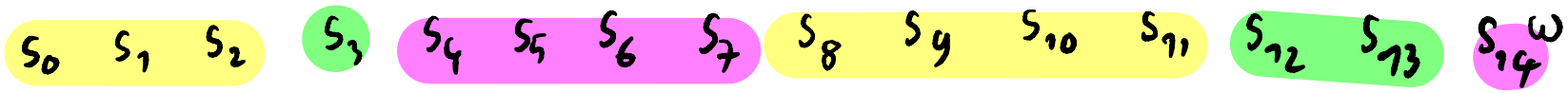
$$S/\approx_T^{div} = \{\{s_0\}, \{s_1\}, \{s_2, s_3\}, \{s_4\}\}$$

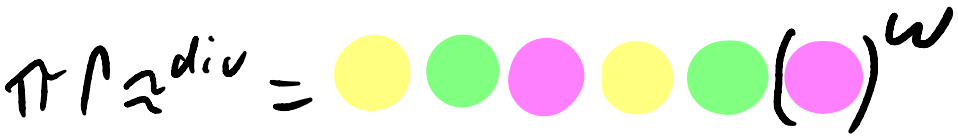
Divergence Stutter-Bisimilar Paths

For infinite path fragment $\pi = s_0 s_1 s_2 s_3 \ldots \in S^\omega$ let $\pi \restriction \approx^{div} \in (S/\approx^{div})^\omega$ be

$$\pi \restriction \approx^{div} = \begin{cases} ([s_0]_{\approx^{div}})^\omega & \text{if } \pi \text{ diverges} \\ [s_0]_{\approx^{div}} \, \pi[1..] \restriction \approx^{div} & \text{if } s_0 \not\approx^{div} s_1 \\ \pi[1..] \restriction \approx^{div} & \text{otherwise} \end{cases}$$

Example: $S/\approx^{div} = \{$  $\}$

$\pi =$ 

$\pi \restriction \approx^{div} =$ 

Divergence Stutter-Bisimilar Paths

For infinite path fragment $\pi = s_0 s_1 s_2 s_3 \ldots \in S^\omega$ let $\pi \restriction\approx^{div} \in (S/\approx^{div})^\omega$ be

$$\pi \restriction\approx^{div} = \begin{cases} ([s_0]_{\approx^{div}})^\omega & \text{if } \pi \text{ diverges} \\ [s_0]_{\approx^{div}} \, \pi[1..] \restriction\approx^{div} & \text{if } s_0 \not\approx^{div} s_1 \\ \pi[1..] \restriction\approx^{div} & \text{otherwise} \end{cases}$$

Definition: divergence stutter-bisimilar paths

Two infinite path fragments π_1 and π_2 are divergence stutter-bisimilar, denoted $\pi_1 \approx^{div} \pi_2$, iff $\pi_1 \restriction\approx^{div} = \pi_2 \restriction\approx^{div}$

$s_1 \approx^{div}_{TS} s_2$ implies $\forall \pi_1 \in Paths(s_1).\ \exists \pi_2 \in Paths(s_2).\ \pi_1 \approx^{div} \pi_2$

$s_1 \approx^{div}_{TS} s_2$ implies $s_1 \equiv_{sttrace} s_2$

Quotient Transition System Under $\approx^{div}$

Definition: quotient transition system under $\approx^{div}$

The quotient of TS under relation $\approx^{div}$ is defined as:

$$TS/\approx^{div} = (S/\approx^{div}, \{\tau\}, \rightarrow', I', AP, L')$$

eq. classes of $\approx^{div}$

where

- I' and L' are defined as usual (for eq. classes $[s]$ under $\approx^{div}$)
- transition relation $\rightarrow'$ is defined by:

$$\frac{s \xrightarrow{\alpha} s' \ \wedge \ s \not\approx^{div} s'}{[s] \xrightarrow{\tau}{}' [s']} \qquad \text{and} \qquad \frac{s \text{ is } \approx^{div}\text{-divergent}}{[s] \xrightarrow{\tau}{}' [s]}$$

Quotient Transition System Under $\approx^{div}$

Definition: quotient transition system under $\approx^{div}$

The quotient of TS under relation $\approx^{div}$ is defined as:

$$TS/\approx^{div} = (S/\approx^{div}, \{\tau\}, \rightarrow', I', AP, L')$$

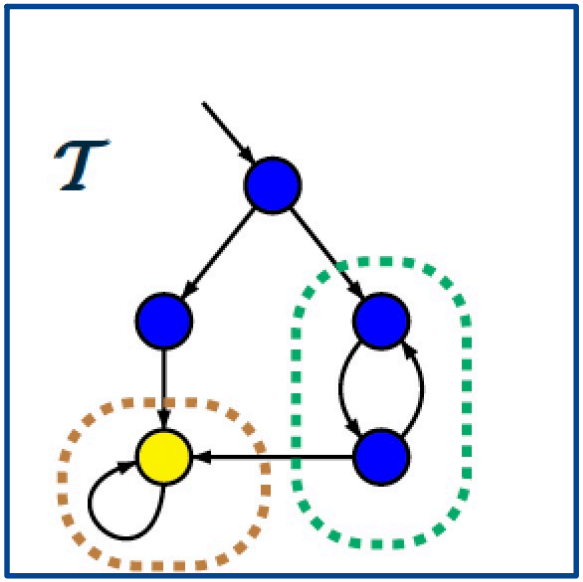
where

- I' and L' are defined as usual (for eq. classes $[s]$ under $\approx^{div}$)
- transition relation $\rightarrow'$ is defined by:

$$\frac{s \xrightarrow{\alpha} s' \ \wedge \ s \not\approx^{div} s'}{[s] \xrightarrow{\tau}{}' [s']} \qquad \text{and} \qquad \frac{s \text{ is } \approx^{div}\text{-divergent}}{[s] \xrightarrow{\tau}{}' [s]}$$

For every transition system TS it holds: $TS \approx^{div} TS/\approx^{div}$.

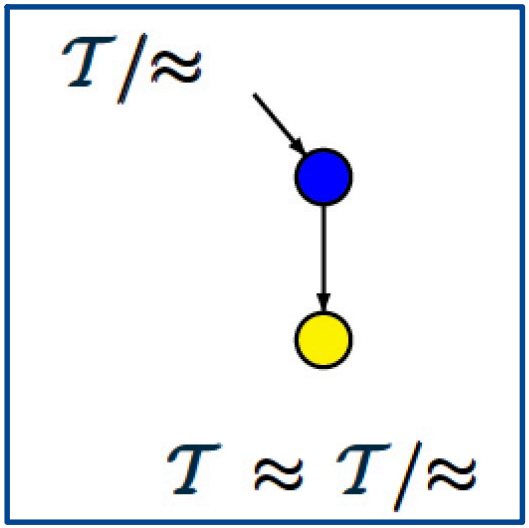
Example



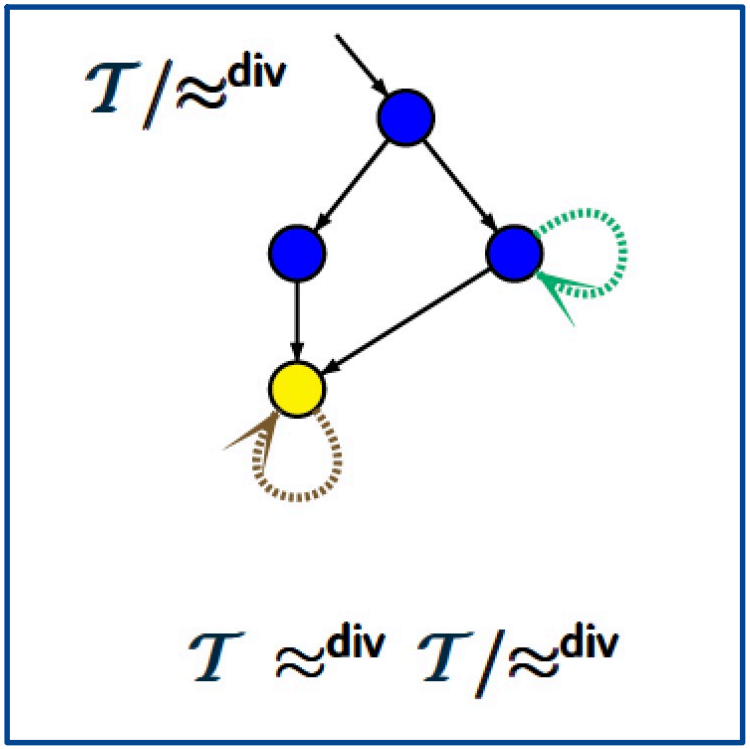
$\not\approx_{div}$

$\widehat{\bullet} \triangleq \{a\}$

$\widehat{\bullet} \triangleq \emptyset$



$\not\approx$



Stutter Bisimilarity And $CTL_{\setminus\bigcirc}$

Theorem: Stutter Bisimilarity, $CTL_{\setminus\bigcirc}$ and $CTL^*_{\setminus\bigcirc}$

Let TS be a finitely branching[1] transition system and s, s' states in TS. The following three statements are equivalent:

1. $s \approx^{div}_{TS} s'$
2. s and s' are $CTL_{\setminus\bigcirc}$-equivalent, i.e., $s \equiv_{CTL_{\setminus\bigcirc}} s'$
3. s and s' are $CTL^*_{\setminus\bigcirc}$-equivalent, i.e., $s \equiv_{CTL^*_{\setminus\bigcirc}} s'$.

Proof.

This is proven in three steps in a similar way as for the correspondence between bisimilarity, CTL- and CTL*-equivalence. □

[1] Every state has only finitely many direct successors.

Show "2. $\Rightarrow$ 1.": $s_1 \equiv_{CTL\setminus O} s_2$ implies $s_1 \approx^{div} s_2$

Proof: $R = \{(s,s') \mid s \equiv_{CTL\setminus O} s'\}$

claim: R is ① a stutter bisimulation and ② divergent sensitive

① For each $C \in S/R$ let $CTL\setminus O$-formula Φ_C be defined by $\Phi_C := \bigwedge_{D \in S/R, D \neq C} \Phi_{C,D}$

where $\Phi_{C,D}$ with $sat(\Phi_{C,D}) \supseteq C$ and $sat(\Phi_{C,D}) \cap D = \emptyset$

exists because $s \not\equiv_{CTL\setminus O} s'$ then there is a disting. formula

Thus $sat(\Phi_C) = C$

To prove: for all $s_1' \in Post(s_1)$ with $(s_1, s_1') \notin R$ there is a finite path $s_2\, u_1 \dots u_n\, s_2'$ with $(s_1', s_2') \in R$, $\forall i: (s_2, u_i) \in R$

Let $B \in S/R$ $B = [s_1]_R = [s_2]_R$

Let $C = [s_1']_R \neq B$. Then: $s_1 \models \exists(\Phi_B \cup \Phi_C)$

since $(s_1, s_2) \in R$ and $\exists \Phi_B \cup \Phi_C \in CTL\setminus O$ it follows $s_2 \models \exists(\Phi_B \cup \Phi_C)$. Thus there is a finite path fragment $s_2\, u_1 \dots u_n\, s_2'$ with $s_2' \not\models \Phi_C$ and $s_2, u_1, \dots, u_n \models \Phi_B$

(2) assume $(s_1, s_2) \in R$ and s_1 is R-divergent

To show: s_2 is R-divergent

There is $\pi_1 = s_1, s_{1,1}, s_{1,2}, s_{1,3} \ldots \in Paths(s_1)$
s.t. π_1 is R-divergent.

Let $C = [s_1]_R$. Then $s_{1,i} \models \Phi_C \quad \forall i$

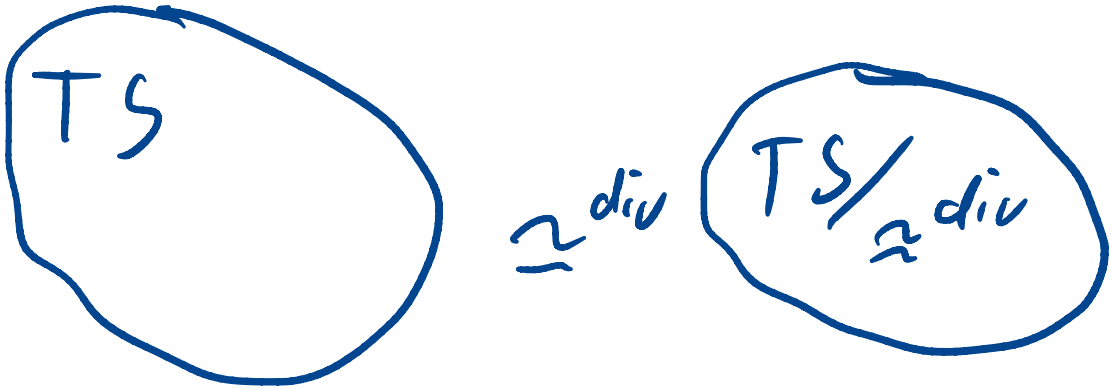
Thus $s_1 \models \underbrace{\exists \Box \Phi_C}_{\in CTL/O}$

since $(s_1, s_2) \in R$ it follows $s_2 \models \exists \Box \Phi_C$

thus s_2 is R-divergent. □

For any transition systems TS and TS' (over AP):

$$TS \approx^{div} TS' \quad \text{iff} \quad TS \equiv_{CTL_{\setminus\bigcirc}} TS' \quad \text{iff} \quad TS \equiv_{CTL^*_{\setminus\bigcirc}} TS'.$$



$TS \models \Phi \iff TS/_{\simeq^{div}} \models \Phi$ ($\Phi \in CTL^*_{\setminus\bigcirc}$)

$|TS| \gg |TS/_{\simeq}| \gg |TS/_{\simeq^{div}}|$

Overview

$\approx^{div}$-Quotienting Algorithm

Jan Friso Groote (1965 –)

Frits Vaandrager (1962 –)

$\approx^{div}$-Quotienting Algorithm

Algorithm with two components:

$\approx^{div}$-Quotienting Algorithm

Algorithm with two components:

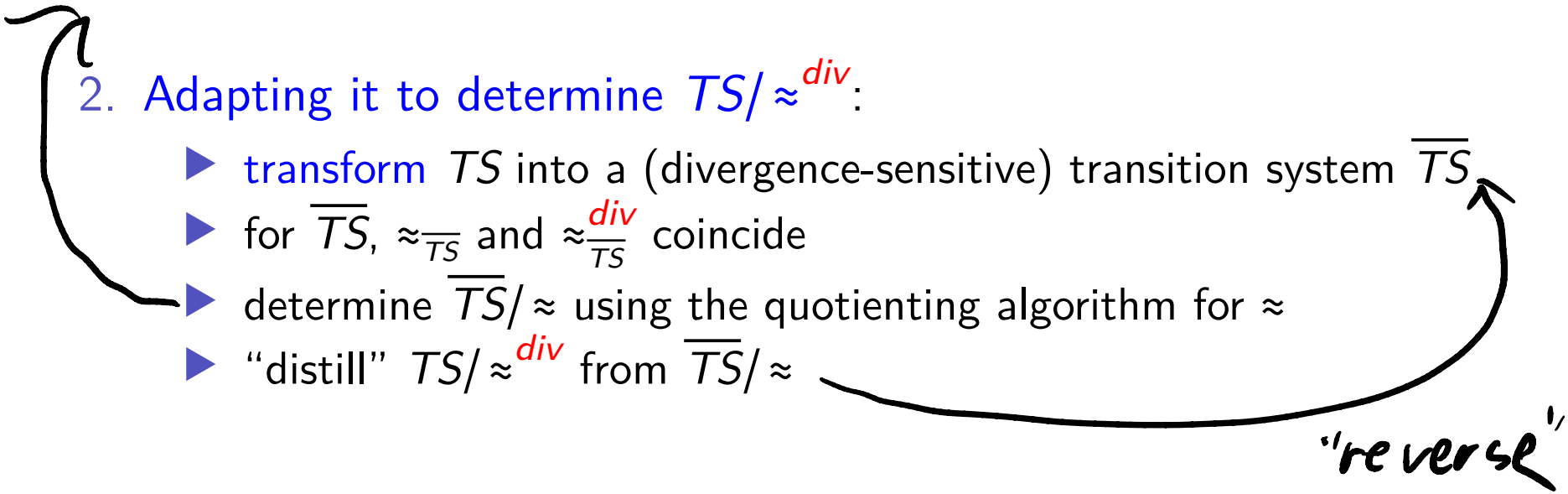
1. A quotienting algorithm to determine $\overline{TS}/\approx$:
   - remove stutter cycles from $\overline{TS}$
   - a refine operator to efficiently split (blocks of) partitions
   - exploit partition-refinement (as for bisimilarity ~)

$\approx^{div}$-Quotienting Algorithm

Algorithm with two components:

1. A quotienting algorithm to determine $\overline{TS}/\approx$:
   - remove stutter cycles from $\overline{TS}$
   - a refine operator to efficiently split (blocks of) partitions
   - exploit partition-refinement (as for bisimilarity ~)

2. Adapting it to determine $TS/\approx^{div}$:
   - transform TS into a (divergence-sensitive) transition system $\overline{TS}$
   - for $\overline{TS}$, $\approx_{\overline{TS}}$ and $\approx^{div}_{\overline{TS}}$ coincide
   - determine $\overline{TS}/\approx$ using the quotienting algorithm for $\approx$
   - "distill" $TS/\approx^{div}$ from $\overline{TS}/\approx$

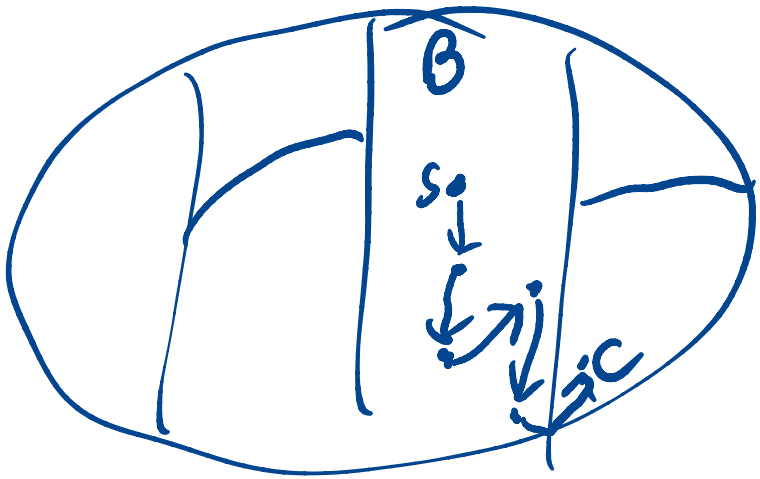


Constrained Predecessors

For partition Π of S and blocks B, C in Π we define:

$$Pre^*_\Pi(B, C) := \left\{ s \in B \mid \exists \overbrace{s_0}^{=s} \underbrace{s_1 \ldots s_{n-1}}_{\in B} \underbrace{s_n}_{\in C} \in Paths^*(s) \cap \left(B^+ . C\right) \right\}$$

That is: state s can reach C via a path that is entirely in B $(= [s]_\Pi)$



Splitters

- Let Π be a partition of S and C a super-block of Π
- C is a Π-splitter of Π for $B \in \Pi$ if:

$$B \neq C \quad \text{and} \quad B \cap Pre(C) \neq \emptyset \quad \text{and} \quad B \setminus Pre^*_\Pi(B, C) \neq \emptyset$$

- Partition Π is stable w.r.t. C if there is no $B \in \Pi$ such that C is a Π-splitter for B
- Π is stable if Π is stable w.r.t. C for all blocks $C \in \Pi$

Coarsest Partition

$S/\approx$ is the coarsest partition Π of S such that:

1. Π is finer than the initial partition Π_{AP}, and
2. for all $B, C \in \Pi$ it holds:

$$B \cap Pre(C) = \varnothing \quad \text{or} \quad B \subseteq Pre^*_\Pi(B, C).$$

Refinement

Definition: refinement operator for $\approx$

For Π a partition for S and C a block of Π:

$$Refine_{\approx}(\Pi, C) = \bigcup_{B \in \Pi} Refine_{\approx}(B, C)$$

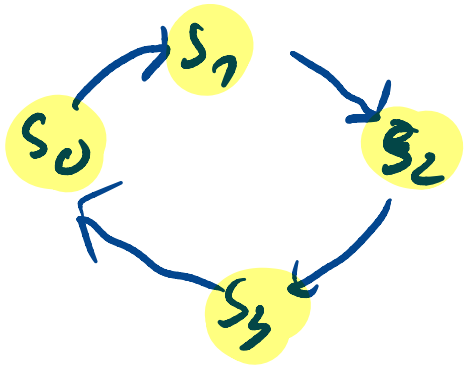
where

$$Refine_{\approx}(B, C) = \{B \cap Pre^{*}_{\Pi}(B, C),\ B \setminus Pre^{*}_{\Pi}(B, C)\} \setminus \{\varnothing\}$$

Stutter Cycles

Definition: stutter cycle

$s_0\, s_1\, \ldots\, \underbrace{s_n}_{= s_0}$ is a stutter cycle if it only consists of stutter steps.

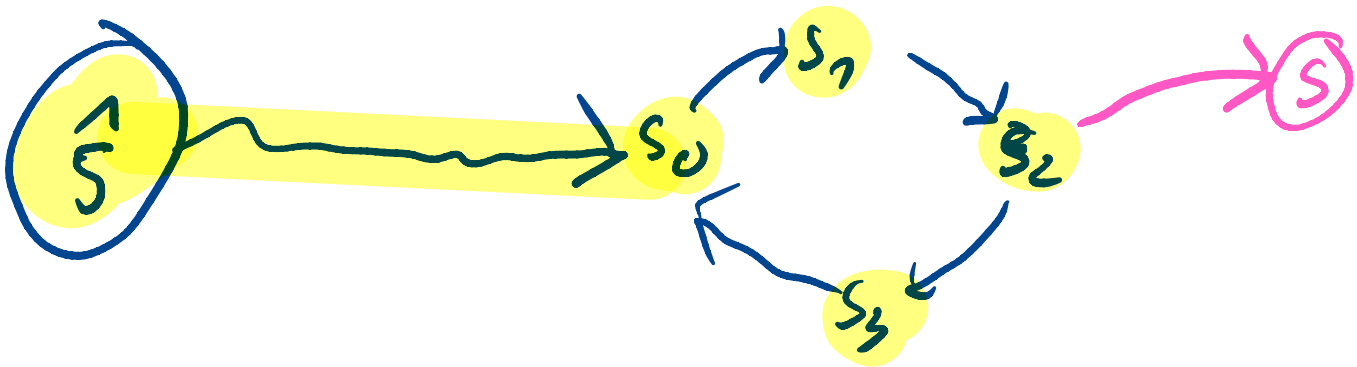


Stutter Cycles

Definition: stutter cycle

$s_0\, s_1\, \ldots \underbrace{s_n}_{= s_0}$ is a stutter cycle if it only consists of stutter steps.

For stutter cycle $s_0 \ldots s_n$ in TS it holds: $s_0 \approx^{div}_{TS} s_1 \approx^{div}_{TS} \ldots \approx^{div}_{TS} s_n$.



Stutter Cycles

Definition: stutter cycle

$s_0\, s_1\, \ldots \underbrace{s_n}_{= s_0}$ is a stutter cycle if it only consists of stutter steps.

For stutter cycle $s_0 \ldots s_n$ in TS it holds: $s_0 \approx^{div}_{TS} s_1 \approx^{div}_{TS} \ldots \approx^{div}_{TS} s_n$.

Corollary

For finite TS and state s in TS:

$$s \text{ is } \approx^{div}\text{-divergent}$$

if and only if

a stutter cycle is reachable from s via a path in $[s]_{div}$.

Removing Stutter Cycles

1. Determine the SCCs in TS that only contain stutter steps
   - use DFS to find these strongly connected components (SCCs)

2. Collapse every stutter SCC into a single state
   - $C \to' C'$ with $C \neq C'$ whenever $s \to s'$ in TS with $s \in C$ and $s' \in C'$

3. Resulting TS' has no stutter cycles
   - $s_1 \approx_{TS} s_2$ if and only if $\underbrace{C_1}_{s_1 \in C_1} \approx_{TS'} \underbrace{C_2}_{s_2 \in C_2}$

From now on, assume transition systems have no stutter cycles.

Efficient Refinement

- C is a Π-splitter for B if and only if:

$$B \neq C \quad \text{and} \quad B \cap Pre(C) \neq \varnothing \quad \text{and} \quad B \setminus Pre^*_\Pi(B, C) \neq \varnothing$$

- How to avoid the (expensive) computation of $Pre^*_\Pi(C)$ for $C \in \Pi$?

Efficient Refinement

- C is a Π-splitter for B if and only if:

$$B \neq C \quad \text{and} \quad B \cap Pre(C) \neq \varnothing \quad \text{and} \quad B \setminus Pre^*_\Pi(B, C) \neq \varnothing$$

- How to avoid the (expensive) computation of $Pre^*_\Pi(C)$ for $C \in \Pi$?

- No stutter cycles $\Rightarrow$ block $B \in \Pi$ has at least one exit state
  - exit state = a state with only direct successors outside B:
    $Bottom(B) = \{s \in B \mid Post(s) \cap B = \varnothing\}$ $\neq \phi$

Efficient Refinement

- C is a Π-splitter for B if and only if:

  $$B \neq C \quad \text{and} \quad B \cap \mathit{Pre}(C) \neq \varnothing \quad \text{and} \quad B \setminus \mathit{Pre}^*_\Pi(B, C) \neq \varnothing$$

- How to avoid the (expensive) computation of $\mathit{Pre}^*_\Pi(C)$ for $C \in \Pi$?

- No stutter cycles $\Rightarrow$ block $B \in \Pi$ has at least one exit state
  - exit state = a state with only direct successors outside B:
    $\mathit{Bottom}(B) = \{s \in B \mid \mathit{Post}(s) \cap B = \varnothing\}$

- For finite TS without stutter cycles, C is a Π-splitter for B iff:

  $$B \neq C \quad \text{and} \quad B \cap \mathit{Pre}(C) \neq \varnothing \quad \text{and} \quad \mathit{Bottom}(B) \setminus \mathit{Pre}(C) \neq \varnothing$$

Time Complexity of Computing $TS/\approx$

The partition-refinement algorithm to compute $TS/\approx$ has a worst-case time complexity in $O(N \cdot M)$ where N and M are the number of states and transitions, respectively, in TS.

Quotienting Algorithm for $\approx^{div}$

1. A quotienting algorithm to determine $\overline{TS}/\approx$:
   - remove stutter cycles from $\overline{TS}$
   - a refine operator to efficiently split (blocks of) partitions
   - exploit partition-refinement (as for bisimilarity $\sim$)

2. Adapting it to determine $TS/\approx^{div}$:
   - transform TS into a (divergence-sensitive) transition system $\overline{TS}$
   - for $\overline{TS}$, $\approx_{\overline{TS}}$ and $\approx^{div}_{\overline{TS}}$ coincide
   - determine $\overline{TS}/\approx$ using the quotienting algorithm for $\approx$
   - "distill" $TS/\approx^{div}$ from $\overline{TS}/\approx$

Divergence Expansion

Definition: divergence expansion

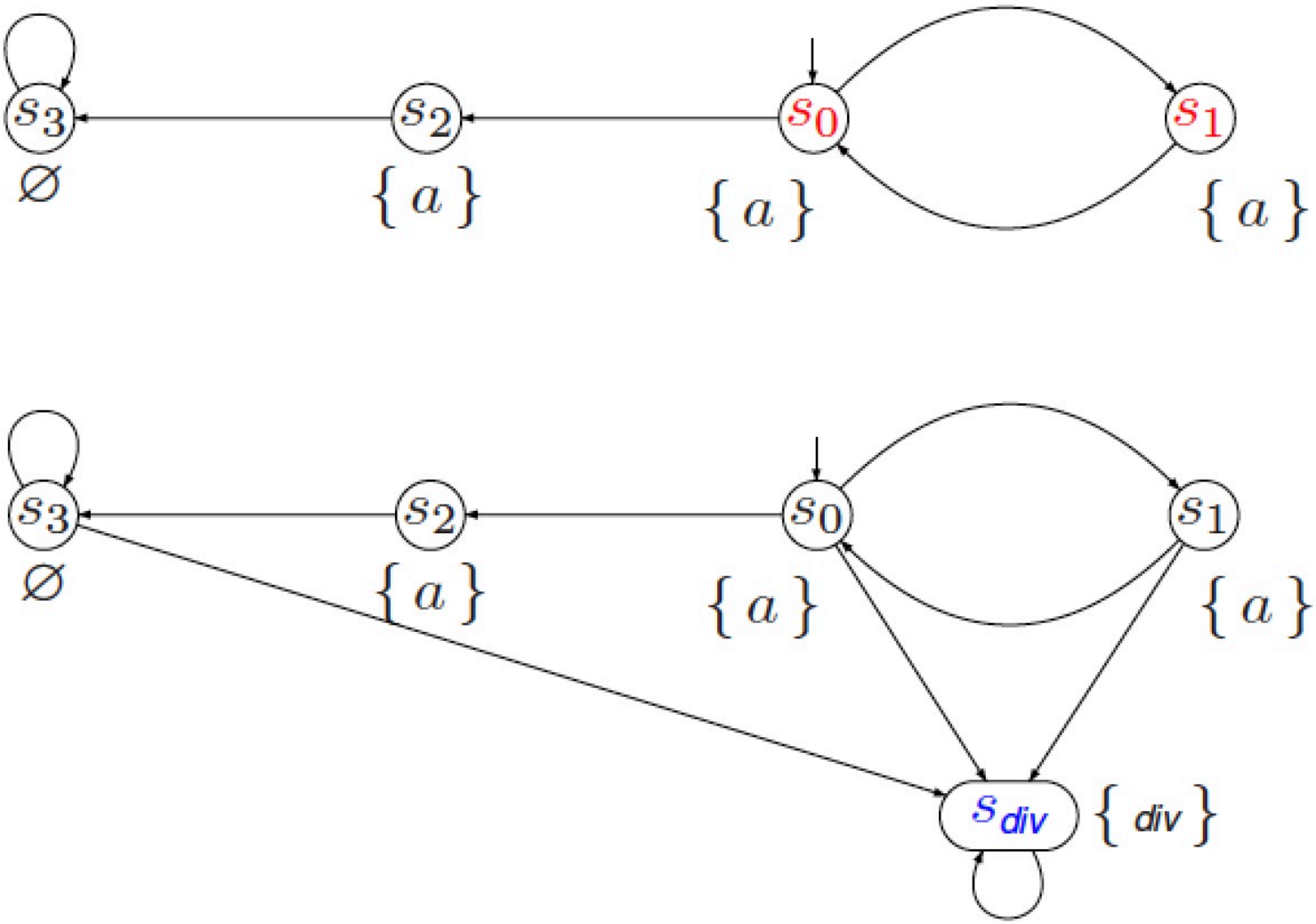
The divergence-sensitive expansion of $TS = (S, Act, \rightarrow, I, AP, L)$ is:

$$\overline{TS} \;=\; \big(S \cup \{ s_{div} \}, Act \cup \{ \tau \}, \rightarrow', I, AP \uplus \{ div \}, \overline{L}\big) \quad \text{where}$$

- the transition relation of TS is extended by:
  - $s_{div} \xrightarrow{\tau} s_{div}$ and
  - $s \xrightarrow{\tau} s_{div}$ for every state $s \in S$ on a stutter cycle in TS
- $\overline{L}(s) = L(s)$ if $s \in S$ and $\overline{L}(s_{div}) = \{ div \}$.

$s_{div} \not\approx s$ for any $s \in S$ and s_{div} can only be reached from a $\approx^{div}$-divergent state

Example

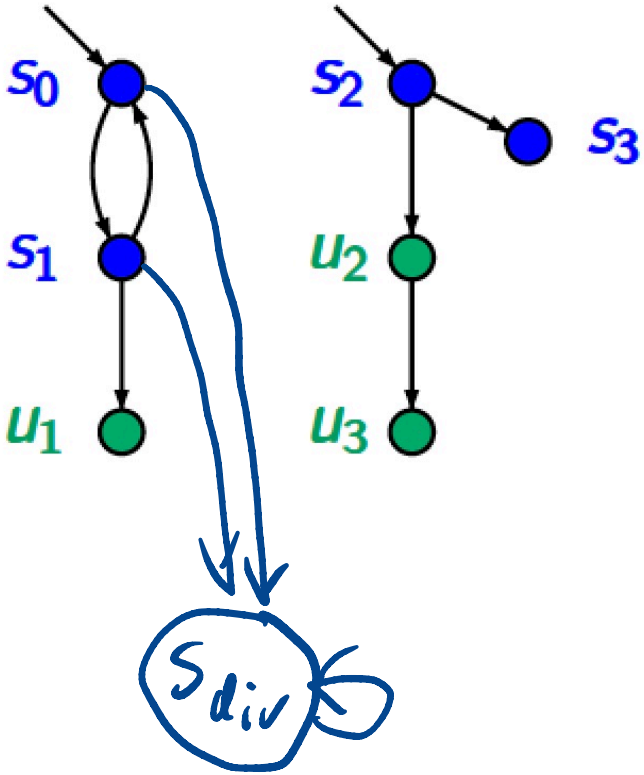


Property of Divergence Expansion

Two states in $\overline{TS}$ are divergence-sensitive stutter bisimilar if and only if they are stutter bisimilar.

$\overline{TS}$ is divergent-sensitive, i.e., if $s \approx s'$ and s is $\approx$-divergent, then s' is $\approx$-divergent too.

Example



divergence-sensitive

stutter equivalence classes:

$\{s_0, s_1\}$ $\{s_2\}$ $\{s_3\}$ $\{u_1, u_2, u_3\}$

s_0 and s_1 are $\approx_T$-divergent

u_1, u_2, u_3 not $\approx_T$-divergent

Recipe for Computing $TS/\approx^{div}$

1. Construct the divergence-sensitive expansion $\overline{TS}$ as follows:
   - determine the SCCs in $G_{stutter}(TS)$, insert $s_{div} \to s_{div}$ and
   - $s \to s_{div}$ for any state s in a non-trivial SCC of $G_{stutter}$
2. Apply partition-refinement to $\overline{TS}$ to obtain $S/\approx_{\overline{TS}} = S/\approx^{div}_{\overline{TS}}$
3. Construct $\overline{TS}/\approx$ as follows:
   - any $C \in S/\approx^{div}$ that contains an initial state of TS is an initial state
   - the labeling of $C \in S/\approx^{div}$ equals the labeling of any $s \in C$
   - every transition $s \to s'$ with $s \not\approx^{div}_{TS} s'$ yields $C_s \to C_{s'}$
4. "Distill" $TS/\approx^{div}$ from $\overline{TS}/\approx$ as follows:
   - replace transition $s \to s_{div}$ in $\overline{TS}$ by the self-loop $[s]_{\approx^{div}} \to [s]_{\approx^{div}}$
   - delete state s_{div}

Time Complexity

The quotient transition system $TS/\approx^{div}$ can be determined with a worst-case time complexity in $O(M \cdot N)$

Recently, this has been improved to $O(M \cdot \log N)$.

└ 2018

Overview

Overview

	trace equivalence	bisimulation	simulation
complexity	PSPACE-complete	$O(M \cdot \log N)$	$O(M \cdot N)$
logical fragment	LTL	CTL*	$\forall$CTL*
preservation	strong match	strong match	weak match

	stutter trace equivalence	divergence-sensitive stutter bisimulation
complexity	PSPACE-complete	$O(M \cdot N)$
logical fragment	LTL$_{\setminus\bigcirc}$	CTL$^*_{\setminus\bigcirc}$
preservation	strong match	strong match

Next Lecture

Thursday June 30, 12:30