Stutter Equivalences

Lecture #4–#6 of Advanced Model Checking

Joost-Pieter Katoen

Software Modeling and Verification Group

affiliated to University of Twente, Formal Methods and Tools

October 27, 2016

Content of this lecture

- Stutter trace equivalence
  - definition, properties, LTL (no next) equivalence
- Stutter bisimulation
  - definition, properties, no LTL (no next) equivalence
- Divergence sensitivity
  - divergence-sensitive bisimulation, CTL* (no next) equivalence
- Divergence-sensitive bisimulation minimisation
  - basic idea of algorithm, complexity

Content of this lecture

$\Rightarrow$ Stutter trace equivalence

- definition, properties, LTL (no next) equivalence

- Stutter bisimulation
  - definition, properties, no LTL (no next) equivalence
- Divergence sensitivity
  - divergence-sensitive bisimulation, CTL* (no next) equivalence
- Divergence-sensitive bisimulation minimisation
  - basic idea of algorithm, complexity

Motivation

- Bisimulation, simulation and trace equivalence are *strong*
  - each transition $s \rightarrow s'$ must be matched by a transition of a related state
  - for comparing models at different abstraction levels, this is too fine
  - consider e.g., modeling an abstract action by a sequence of concrete actions
- Idea: allow for sequences of "invisible" transitions
  - each transition $s \rightarrow s'$ must be matched by a path fragment of a related state
  - matching means: ending in a state related to s', and all previous states invisible
- Abstraction of such internal computations yields coarser quotients
  - but: what kind of properties are preserved?
  - but: can such quotients still be obtained efficiently?
  - but: how to treat infinite internal computations?

Motivating example

Let *TS*$_{conc}$ model the concrete program fragment

$i := y$; $z := 1$;
while $i > 1$ **do**
  $z := z * i$; $i := i - 1$;
od
$x := z$;

that computes the factorial of y iteratively.

Let *TS*$_{abs}$ be the transition system of the (abstract) program $x := y$!

Clearly, *TS*$_{abs}$ and *TS*$_{conc}$ are in some sense equivalent

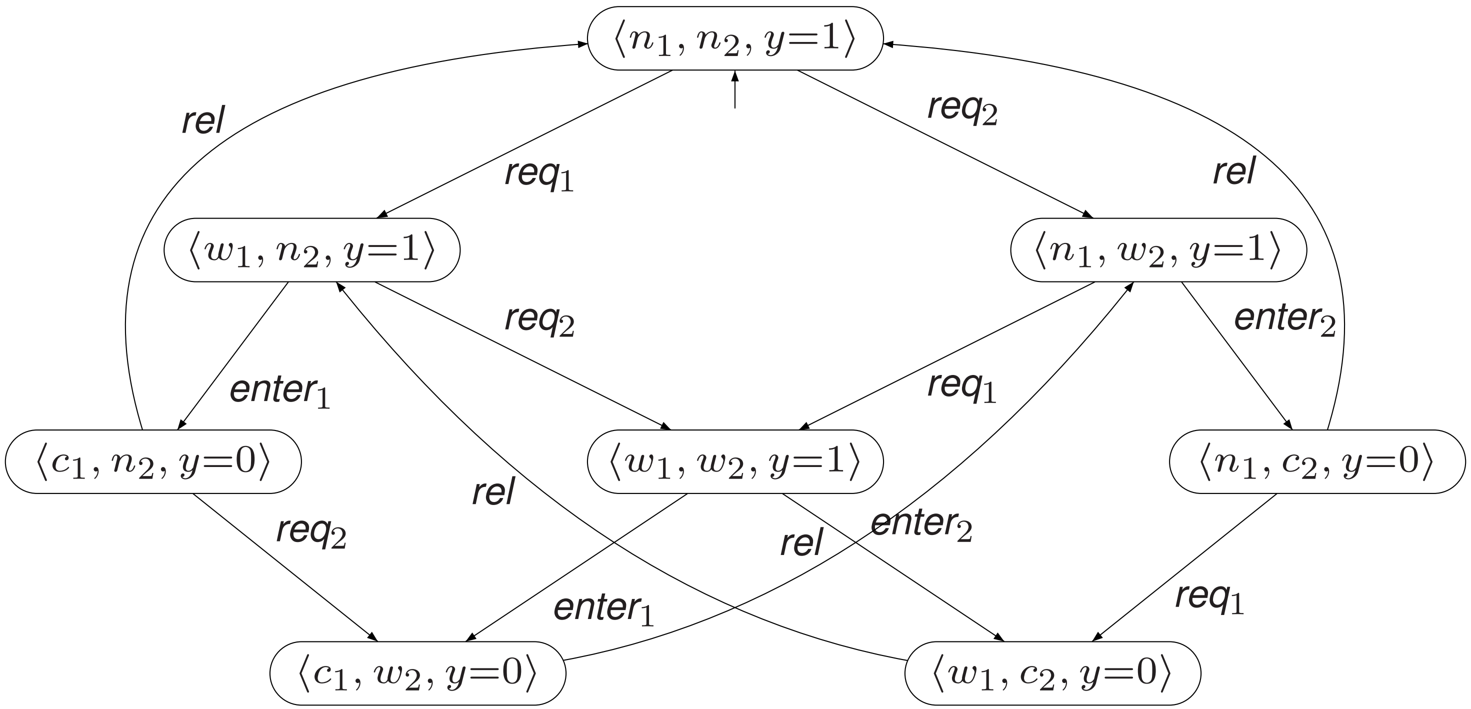
Stuttering equivalence

- $s \to s'$ in transition system *TS* is a stutter step if $L(s) = L(s')$
- Paths π_1 and π_2 are stutter equivalent, denoted $\pi_1 \triangleq \pi_2$:
  - if there exists an infinite sequence $A_0 A_1 A_2 \ldots$ with $A_i \subseteq AP$ and
  - natural numbers $n_0, n_1, n_2, \ldots, m_0, m_1, m_2, \ldots > 0$ such that:

$$
\begin{aligned}
trace(\pi_1) &= \underbrace{A_0 \ldots A_0}_{n_0\text{-times}} \underbrace{A_1 \ldots A_1}_{n_1\text{-times}} \underbrace{A_2 \ldots A_2}_{n_2\text{-times}} \ldots \\
trace(\pi_2) &= \underbrace{A_0 \ldots A_0}_{m_0\text{-times}} \underbrace{A_1 \ldots A_1}_{m_1\text{-times}} \underbrace{A_2 \ldots A_2}_{m_2\text{-times}} \ldots
\end{aligned}
$$

$\Rightarrow$ $\pi_1 \triangleq \pi_2$ if both their traces are of the form $A_0{}^+ A_1{}^+ A_2{}^+ \ldots$ for $A_i \subseteq AP$

Semaphore-based mutual exclusion



Stutter equivalent traces

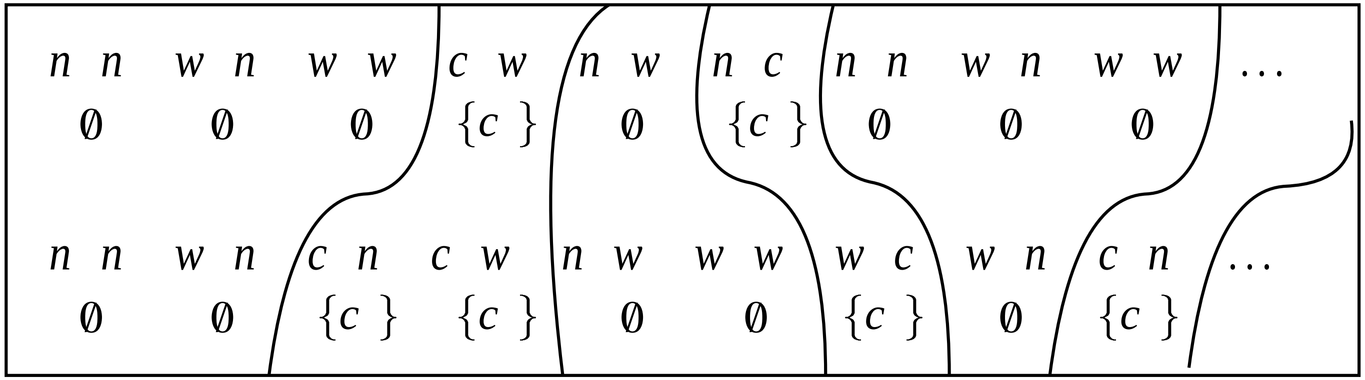
These infinite paths are stutter equivalent

$$\begin{array}{rcl} \pi_1 & = & \langle n_1, n_2 \rangle \rightarrow \langle w_1, n_2 \rangle \rightarrow \langle w_1, w_2 \rangle \rightarrow \langle c_1, w_2 \rangle \rightarrow \langle n_1, w_2 \rangle \rightarrow \\ & & \langle n_1, c_2 \rangle \rightarrow \langle n_1, n_2 \rangle \rightarrow \langle w_1, n_2 \rangle \rightarrow \langle w_1, w_2 \rangle \rightarrow \langle c_1, w_2 \rangle \rightarrow \ldots \\ \pi_2 & = & \langle n_1, n_2 \rangle \rightarrow \langle w_1, n_2 \rangle \rightarrow \langle c_1, n_2 \rangle \rightarrow \langle c_1, w_2 \rangle \rightarrow \langle n_1, w_2 \rangle \rightarrow \\ & & \langle w_1, w_2 \rangle \rightarrow \langle w_1, c_2 \rangle \rightarrow \langle w_1, n_2 \rangle \rightarrow \langle c_1, n_2 \rangle \rightarrow \ldots \end{array}$$

Hence, $\pi_1 \triangleq \pi_2$, since for $AP = \{ crit_1, crit_2 \}$:

$$\begin{array}{rcl} trace(\pi_1) & = & \varnothing^3 \{ crit_1 \} \varnothing \{ crit_2 \} \varnothing^3 \{ crit_1 \} \ldots \quad \text{and} \\ trace(\pi_2) & = & \varnothing^2 (\{ crit_1 \})^2 \varnothing^2 \{ crit_2 \} \varnothing \{ crit_1 \} \ldots \end{array}$$

Pictorially



Stutter trace equivalence

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

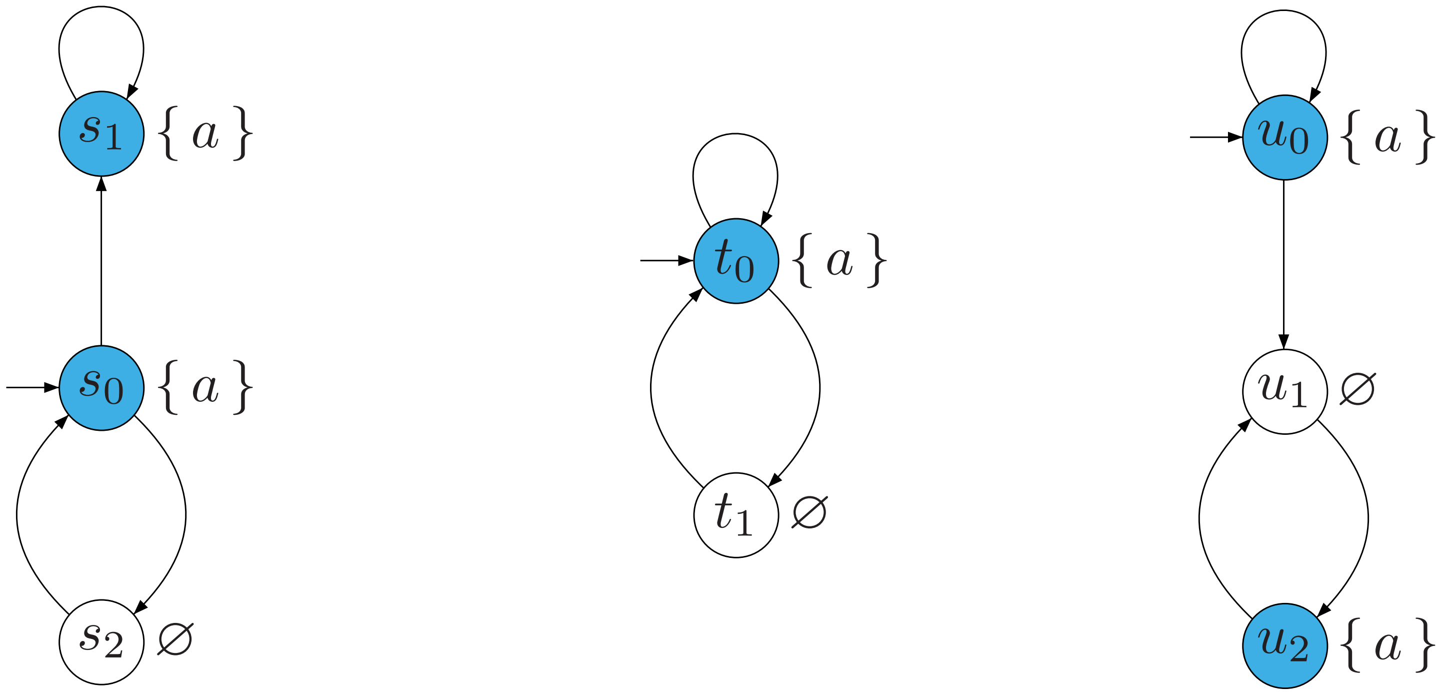
$$TS_1 \triangleq TS_2 \quad \text{if and only if} \quad TS_1 \trianglelefteq TS_2 \text{ and } TS_2 \trianglelefteq TS_1$$

where $\trianglelefteq$, pronounced *stutter trace inclusion*, is defined by:

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

$Traces(TS_1) = Traces(TS_2)$ implies $TS_1 \triangleq TS_2$, but not always the converse

Example



$TS_1 \triangleq TS_2$, $TS_1 \not\trianglelefteq TS_3$ and $TS_2 \not\trianglelefteq TS_3$, but $TS_3 \trianglelefteq TS_2$ and $TS_3 \trianglelefteq TS_1$

The $\bigcirc$ operator

Stuttering equivalence does not preserve the validity of next-formulas:

$\sigma_1 = A\,B\,B\,B\ldots$ and $\sigma_2 = A\,A\,A\,B\,B\,B\,B\ldots$ for $A, B \subseteq AP$ and $A \neq B$

Then for $b \in B \setminus A$:

$$\sigma_1 \triangleq \sigma_2 \quad \text{but} \quad \sigma_1 \models \bigcirc b \quad \text{and} \quad \sigma_2 \not\models \bigcirc b.$$

$\Rightarrow$ a logical characterization of $\triangleq$ can only be obtained by omitting $\bigcirc$

in fact, it turns out that this is the only modal operator that is not preserved by $\triangleq$!

Stutter trace and $\mathrm{LTL}_{\setminus\bigcirc}$ equivalence

For traces σ_1 and σ_2 over 2^{AP} it holds:

$$\sigma_1 \triangleq \sigma_2 \quad \Rightarrow \quad (\sigma_1 \models \varphi \text{ if and only if } \sigma_2 \models \varphi)$$

for any $\mathrm{LTL}_{\setminus\bigcirc}$ formula φ over AP

$\mathrm{LTL}_{\setminus\bigcirc}$ denotes the class of LTL formulas without the next operator $\bigcirc$

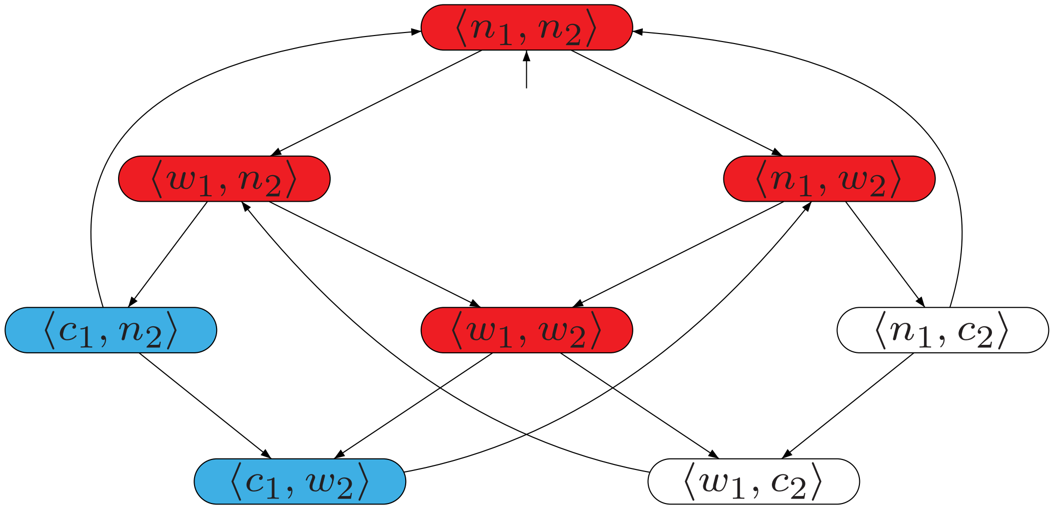
Stutter trace and LTL$_{\setminus\bigcirc}$ equivalence

For transition systems TS_1, TS_2 without terminal states:

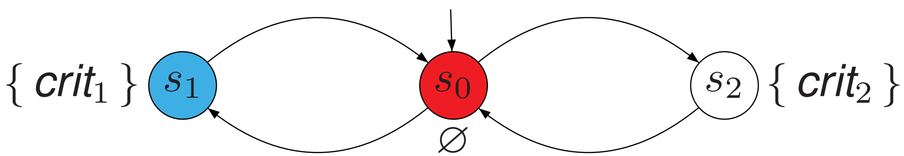
(a) $TS_1 \triangleq TS_2$ if and only if $\left(TS_1 \equiv_{LTL_{\setminus\bigcirc}} TS_2 \right)$

(b) if $TS_1 \trianglelefteq TS_2$ then for any LTL$_{\setminus\bigcirc}$ formula φ: $TS_2 \models \varphi$ implies $TS_1 \models \varphi$

Semaphore-based mutual exclusion



This transition system is stutter trace-equivalent:



Content of this lecture

- Stutter trace equivalence
  - definition, properties, LTL (no next) equivalence

$\Rightarrow$ Stutter bisimulation
  - definition, properties, no LTL (no next) equivalence

- Divergence sensitivity
  - divergence-sensitive bisimulation, CTL* (no next) equivalence

- Divergence-sensitive bisimulation minimisation
  - basic idea of algorithm, complexity

Stutter bisimulation

Let *TS* $= (S, Act, \rightarrow, I, AP, L)$ be a transition system and $\mathcal{R} \subseteq S \times S$

$\mathcal{R}$ is a *stutter-bisimulation* for *TS* if for all $(s_1, s_2) \in \mathcal{R}$:

1. $L(s_1) = L(s_2)$
2. if $s_1' \in Post(s_1)$ with $(s_1', s_2) \notin \mathcal{R}$, then there exists a finite path fragment $s_2\, u_1 \, \ldots \, u_n\, s_2'$ with $n \geqslant 0$ and $(s_1, u_i) \in \mathcal{R}$ and $(s_1', s_2') \in \mathcal{R}$
3. if $s_2' \in Post(s_2)$ with $(s_1, s_2') \notin \mathcal{R}$, then there exists a finite path fragment $s_1\, v_1 \, \ldots \, v_n\, s_1'$ with $n \geqslant 0$ and $(s_2, v_i) \in \mathcal{R}$ and $(s_1', s_2') \in \mathcal{R}$

s_1, s_2 are *stutter-bisimulation equivalent*, denoted $s_1 \approx_{TS} s_2$,
if there exists a stutter bisimulation $\mathcal{R}$ for *TS* with $(s_1, s_2) \in \mathcal{R}$

Stutter bisimulation

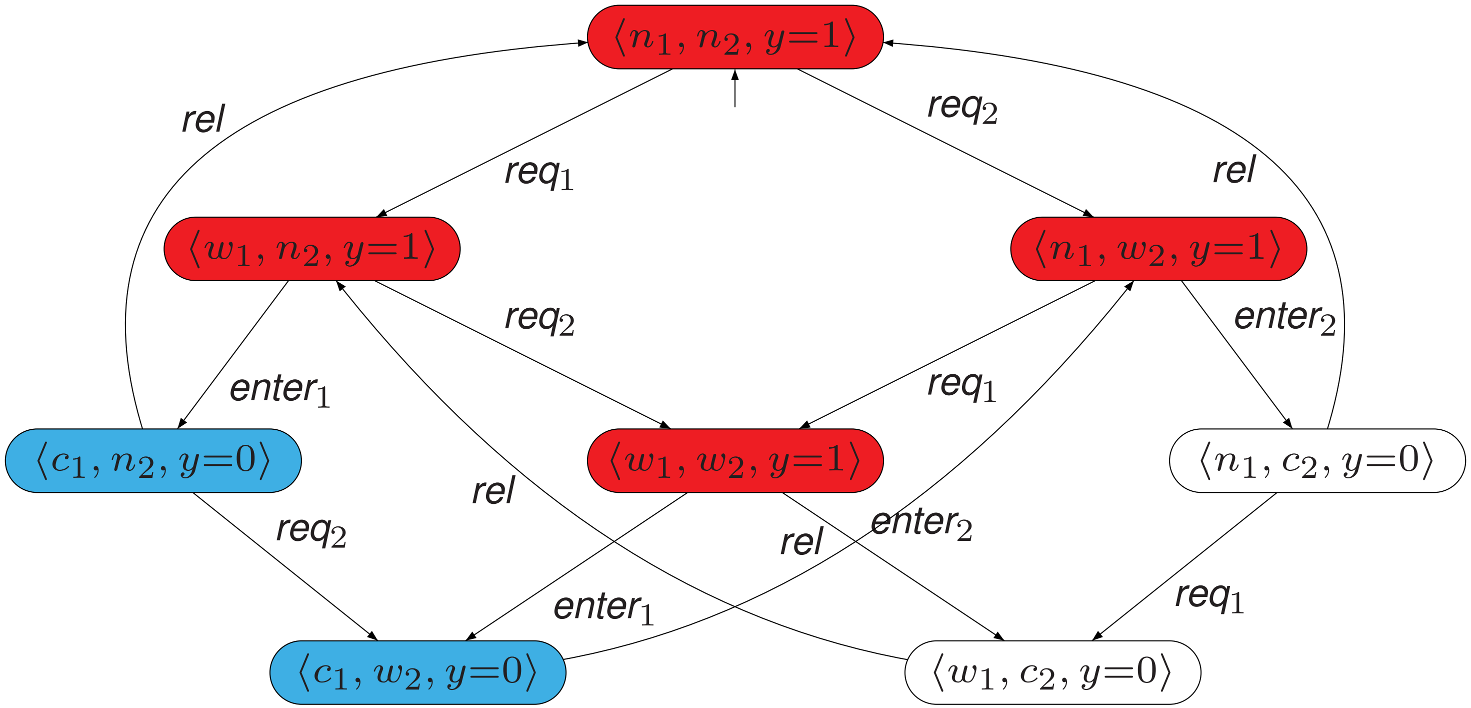
$$\begin{array}{ccc} s_1 & \approx & s_2 \\ \downarrow & & \\ s_1' & & \end{array}$$

(with $s_1' \not\approx s_2$)

can be completed to

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

Semaphore-based mutual exclusion



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

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 $\mathcal{R}$ on $TS_1 \oplus TS_2$ such that:

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

Stutter bisimulation quotient

Let *TS* $=$ $(S, \mathit{Act}, \rightarrow, I, \mathit{AP}, L)$ and stutter bisimulation $\mathcal{R} \subseteq S \times S$ be an *equivalence*

The *quotient* of *TS* under $\mathcal{R}$ is defined by:

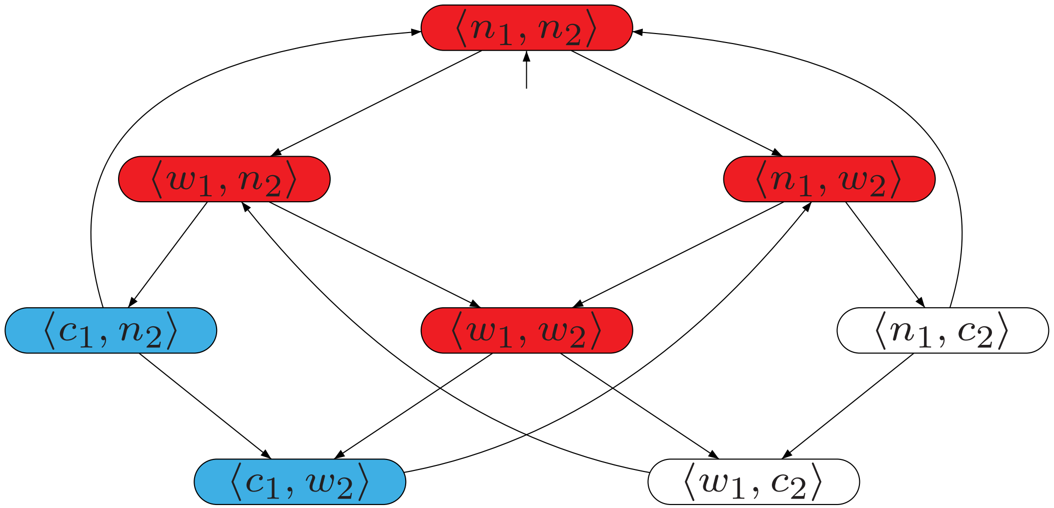
$$\mathit{TS}/\mathcal{R} \;=\; (S', \{\,\tau\,\}, \rightarrow', I', \mathit{AP}, L')$$

where

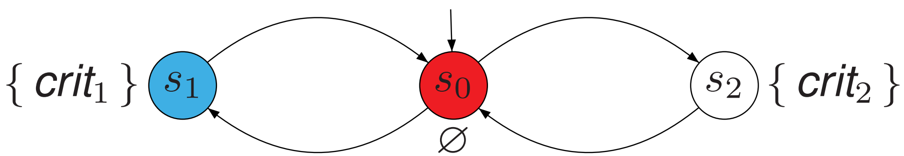
- $S' = S/\mathcal{R} \;=\; \{\, [s]_{\mathcal{R}} \mid s \in S \,\}$ with $[s]_{\mathcal{R}} \;=\; \{\, s' \in S \mid (s, s') \in \mathcal{R} \,\}$
- $I' = \{\, [s]_{\mathcal{R}} \mid s \in I \,\}$
- $L'([s]_{\mathcal{R}}) = L(s)$
- $\rightarrow'$ is defined by: $$\frac{s \xrightarrow{\alpha} s' \text{ and } (s, s') \notin \mathcal{R}}{[s]_{\mathcal{R}} \xrightarrow{\tau}{}' [s']_{\mathcal{R}}}$$

note that (a) no self-loops occur in $\mathit{TS}/\approx_{\mathit{TS}}$ and (b) $\mathit{TS} \approx \mathit{TS}/\approx_{\mathit{TS}}$

Semaphore-based mutual exclusion



The stutter-bisimulation quotient:

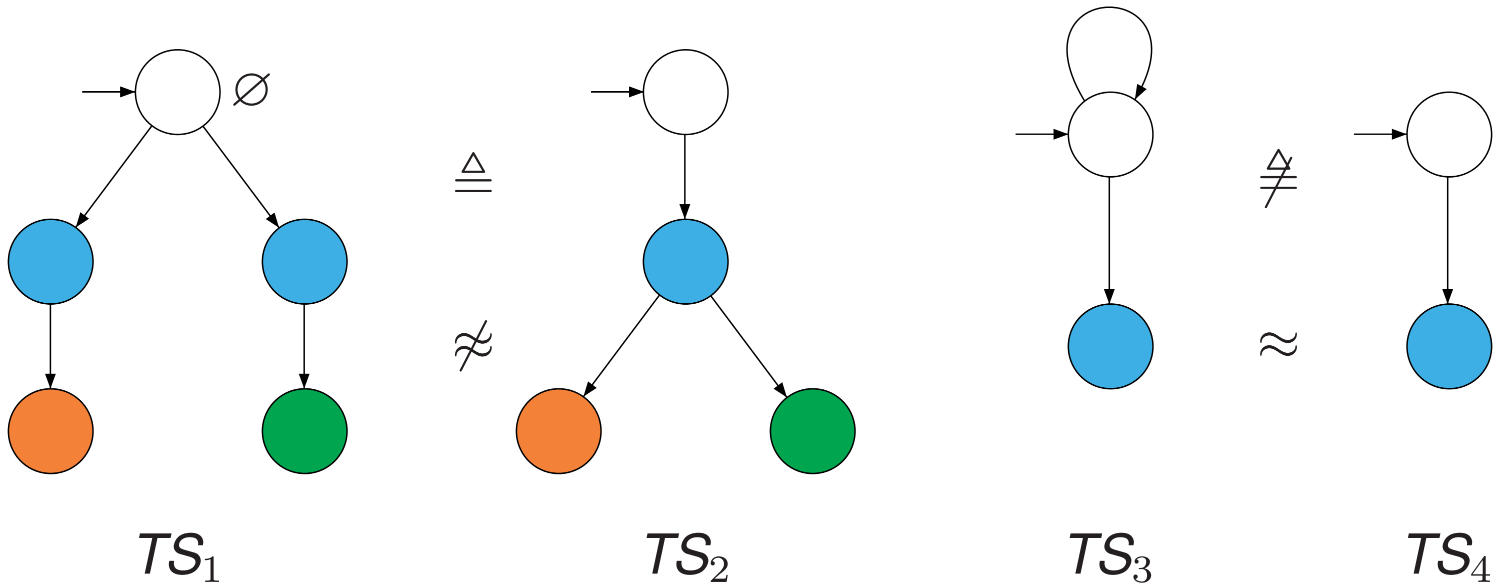


Stutter trace and stutter bisimulation

For transition systems TS_1 and TS_2 over AP:

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

Stutter trace and stutter bisimulation are incomparable



Stutter bisimulation does not preserve $\text{LTL}_{\setminus\bigcirc}$

TS$_{left}$ $\approx$ *TS*$_{right}$ but *TS*$_{left}$ $\not\models \Diamond a$ and *TS*$_{right}$ $\models \Diamond a$

reason: presence of infinite stutter paths in *TS*$_{left}$

Content of this lecture

- Stutter trace equivalence
  - definition, properties, LTL (no next) equivalence
- Stutter bisimulation
  - definition, properties, no LTL (no next) equivalence

$\Rightarrow$ Divergence sensitivity
  - divergence-sensitive bisimulation, CTL* (no next) equivalence

- Divergence-sensitive bisimulation minimisation
  - basic idea of algorithm, complexity

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

  $\Rightarrow$ 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 $\triangleq$ (and $\approx$)

Outlook

formal relation	trace equivalence	bisimulation	simulation
complexity logical fragment preservation	PSPACE-complete LTL strong	PTIME CTL^* strong match	PTIME $\forall\mathrm{CTL}^*$ weak match

formal relation	stutter trace equivalence	divergence-sensitive stutter bisimulation
complexity logical fragment preservation	PSPACE-complete $\mathrm{LTL}_{\setminus\bigcirc}$ strong	PTIME $\mathrm{CTL}^*_{\setminus\bigcirc}$ strong match

Divergence sensitivity

Let *TS* be a transition system and $\mathcal{R}$ an equivalence relation on S

- s is *$\mathcal{R}$-divergent* if there exists an infinite path fragment $s\, s_1\, s_2 \ldots \in$ *Paths*(s) such that $(s, s_j) \in \mathcal{R}$ for all $j > 0$
  - s is $\mathcal{R}$-divergent if there is an infinite path starting in s that only visits $[s]_{\mathcal{R}}$
- $\mathcal{R}$ is *divergence sensitive* if for any $(s_1, s_2) \in \mathcal{R}$:

  s_1 is $\mathcal{R}$-divergent implies s_2 is $\mathcal{R}$-divergent

  - $\mathcal{R}$ is divergence-sensitive if in any $[s]_{\mathcal{R}}$ either all or none states are $\mathcal{R}$-divergent

Divergent-sensitive stutter bisimulation

s_1, s_2 are *divergent-sensitive stutter-bisimilar*, denoted $s_1 \approx_{TS}^{div} s_2$, if:

$\exists$ divergent-sensitive stutter bisimulation $\mathcal{R}$ on *TS* such that $(s_1, s_2) \in \mathcal{R}$

$\approx_{TS}^{div}$ is an equivalence, the coarsest divergence-sensitive stutter bisimulation for *TS*

and the union of all divergence-sensitive stutter bisimulations for *TS*

Quotient transition system under $\approx^{div}$

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

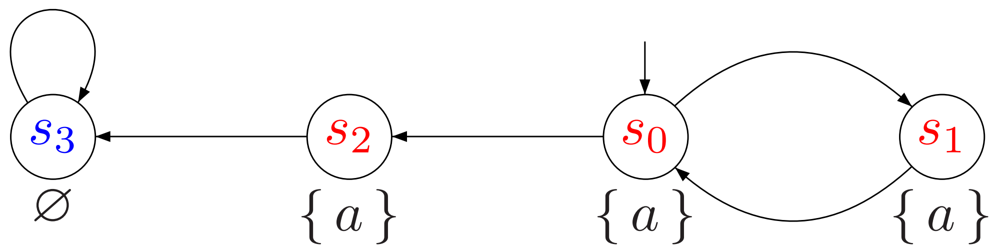
where

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

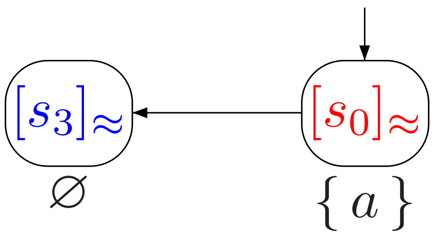
$$\frac{s \xrightarrow{\alpha} s' \;\wedge\; s \not\approx^{div} s'}{[s]_{div} \xrightarrow{\tau}{}'_{div} [s']_{div}} \quad \text{and} \quad \frac{s \text{ is } \approx^{div}\text{-divergent}}{[s]_{div} \xrightarrow{\tau}{}'_{div} [s]_{div}}$$

note that $TS \approx^{div} TS/\approx^{div}$

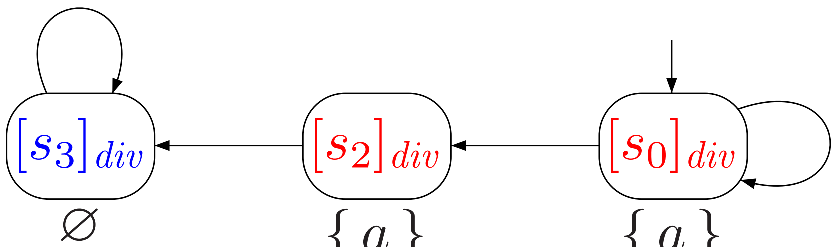
Example



transition system *TS*



transition system $TS/\approx$



transition system $TS/\approx^{div}$

Summary

stutter trace inclusion:

$$TS_1 \trianglelefteq TS_2 \quad \text{iff} \quad \forall \sigma_1 \in \mathit{Traces}(TS_1)\ \exists \sigma_2 \in \mathit{Traces}(TS_2).\ \sigma_1 \triangleq \sigma_2$$

stutter trace equivalence:

$$TS_1 \triangleq TS_2 \quad \text{iff} \quad TS_1 \trianglelefteq TS_2 \text{ and } TS_2 \trianglelefteq TS_1$$

stutter bisimulation equivalence:

$$TS_1 \approx TS_2 \quad \text{iff} \quad \text{there exists a stutter bisimulation for } (TS_1, TS_2)$$

stutter bisimulation equivalence with divergence:

$$TS_1 \approx^{\mathit{div}} TS_2 \quad \text{iff} \quad \text{there exists a divergence-sensitive stutter bisimulation for } (TS_1, TS_2)$$

$\mathrm{CTL}^*_{\setminus \bigcirc}$ and $\mathrm{CTL}_{\setminus \bigcirc}$ equivalence vs $\approx^{div}$

For finite transition system *TS* without terminal states, and s_1, s_2 in *TS*:

$$s_1 \approx^{div}_{TS} s_2 \quad \text{iff} \quad s_1 \equiv_{CTL^*_{\setminus \bigcirc}} s_2 \quad \text{iff} \quad s_1 \equiv_{CTL_{\setminus \bigcirc}} s_2$$

$\mathrm{CTL}^*_{\setminus\bigcirc}$ and $\mathrm{CTL}_{\setminus\bigcirc}$ equivalence vs $\approx^{div}$

For finite transition system *TS* without terminal states, and s_1, s_2 in *TS*:

$$s_1 \approx^{div}_{TS} s_2 \quad \text{iff} \quad s_1 \equiv_{CTL^*_{\setminus\bigcirc}} s_2 \quad \text{iff} \quad s_1 \equiv_{CTL_{\setminus\bigcirc}} s_2 \quad \cancel{\text{iff}} \quad s_1 \equiv_{CTL_{\setminus\bigcirc,U}} s_2$$

Equivalences and logical equivalence

$$
\begin{tikzcd}
\text{CTL}^* \text{ equivalence} \arrow[r] \arrow[d, leftrightarrow] & \text{LTL equivalence} & \\
\begin{array}{c}\text{bisimulation equivalence}\\ \mathit{TS}_1 \sim \mathit{TS}_2\end{array} \arrow[r] \arrow[d] & \begin{array}{c}\text{trace equivalence}\\ \mathit{Traces}(T_1) = \mathit{Traces}(\mathit{TS}_2)\end{array} \arrow[u] \arrow[r] \arrow[d] & \begin{array}{c}\text{trace inclusion}\\ \mathit{Traces}(T_1) \subseteq \mathit{Traces}(\mathit{TS}_2)\end{array} \arrow[d] \\
\begin{array}{c}\text{divergence sensitive}\\ \text{stutter bisimulation equivalence}\\ \mathit{TS}_1 \approx^{\mathit{div}} \mathit{TS}_2\end{array} \arrow[r] \arrow[d, leftrightarrow] & \begin{array}{c}\text{stutter trace-equivalence}\\ \mathit{TS}_1 \triangleq \mathit{TS}_2\end{array} \arrow[r] \arrow[d] & \begin{array}{c}\text{stutter trace inclusion}\\ \mathit{TS}_1 \trianglelefteq \mathit{TS}_2\end{array} \\
\text{CTL}^*_{\setminus \bigcirc} \text{ equivalence} \arrow[r] & \text{LTL}_{\setminus \bigcirc} \text{ equivalence} &
\end{tikzcd}
$$

Content of this lecture

- Stutter trace equivalence
  - definition, properties, LTL (no next) equivalence
- Stutter bisimulation
  - definition, properties, no LTL (no next) equivalence
- Divergence sensitivity
  - divergence-sensitive bisimulation, CTL* (no next) equivalence

$\Rightarrow$ Divergence-sensitive bisimulation minimisation

  - basic idea of algorithm, complexity

Quotienting: Motivation

- Quotienting wrt. $\approx^{div}$ allows to *abstract from stutter steps*
  - in particular $TS \approx^{div} TS/\approx^{div}$
  - typically we have $|TS| \gg |TS/\approx^{div}|$

- $TS_1 \approx^{div} TS_2$ if and only if ($TS_1 \models \Phi$ iff $TS_2 \models \Phi$)
  - for any $\text{CTL}^*_{\setminus\bigcirc}$ (or $\text{CTL}_{\setminus\bigcirc}$) formula Φ

$\Rightarrow$ To check $TS \models \Phi$, if suffices to check whether $TS/\approx^{div} \models \Phi$

- quotienting with respect to $\approx^{div}$ is a useful preprocessing step of model checking

Quotienting: A two-phase approach

[Groote and Vaandrager, 1990]

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

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

Partition-refinement

from now on, we assume that *TS* is finite

- Iteratively compute a partition of S
- Initially: Π_0 equals $\Pi_{AP} = \{ (s,t) \in S \times S \mid L(s) = L(t) \}$ as before
- Repeat until no change: $\boxed{\Pi_{i+1} := \mathit{Refine}_{\approx}(\Pi_i)}$
  - loop invariant: Π_i is coarser than $S/\approx$ and finer than $\{ S \}$
- Return Π_i
  - termination: $\mathcal{R}_{\Pi_0} \supsetneqq \mathcal{R}_{\Pi_1} \supsetneqq \mathcal{R}_{\Pi_2} \supsetneqq \ldots \supsetneqq \mathcal{R}_{\Pi_i} = \approx_{TS}$
  - time complexity: maximally $|\, S \,|$ iterations needed

Theorem

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

(i) Π is finer than the initial partition Π_{AP}, and

(ii) $B \cap \mathit{Pre}^*_\Pi(C) = \varnothing$ or $B \subseteq \mathit{Pre}^*_\Pi(C)$ for all $B, C \in \Pi$

for partition Π of S and blocks B, C in Π we have:

$s \in \mathit{Pre}^*_\Pi(C)$ whenever $\underbrace{s = s_1\, s_2 \ldots s_{n-1}}_{\in B}\ \underbrace{s_n}_{\in C} \in \mathit{Paths}(s)$

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

The refinement operator

- Let: $\mathit{Refine}_{\approx}(\Pi, C) \; = \; \bigcup_{B \in \Pi} \; \mathit{Refine}_{\approx}(B, C)$ for C a block in Π
  - where $\mathit{Refine}_{\approx}(B, C) \; = \; \left\{ B \cap \mathit{Pre}^*_{\Pi}(C), \; B \setminus \mathit{Pre}^*_{\Pi}(C) \right\} \setminus \{ \varnothing \}$
- Basic properties:
  - for Π finer than Π_{AP} and coarser than $S/\approx$:

    $\mathit{Refine}_{\approx}(\Pi, C)$ is finer than Π and $\mathit{Refine}_{\approx}(\Pi, C)$ is coarser than $S/\approx$
  - Π is strictly coarser than $S/\approx$ if and only if there exists a *splitter* for Π

what is an appropriate splitter for $\approx$?

Splitter for $\approx$

Let Π be a partition of S and let C, $B \in \Pi$.

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

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

2. Π is C-*stable* if there is no $B \in \Pi$ such that C is a Π-splitter for B

3. Π is *stable* if Π is C-stable for all blocks $C \in \Pi$

Partition-refinement

Input: finite transition system *TS* with state space S
Output: stutter-bisimulation quotient space $S/\approx$

$\Pi := \Pi_{AP}$; (* as before *)
while ($\exists B, C \in \Pi.\ C$ is a Π-splitter for B) **do**
  choose such $B, C \in \Pi$;
  $\Pi := (\Pi \setminus \{ B \}) \cup \{ \underbrace{B \cap \mathit{Pre}^*_\Pi(C)}_{B_1}, \underbrace{B \setminus \mathit{Pre}^*_\Pi(C)}_{B_2} \} \setminus \{ \varnothing \}$; (* refine Π *)

od
return Π

Stutter cycles

- $s_0\, s_1\, \ldots \underbrace{s_n}_{= s_0}$ is a *stutter cycle* if $s_i\, s_{i+1}$ is a stutter step

- For stutter cycle $s_0\, s_1\, s_2\, \ldots s_n$ in transition system *TS*:

$$s_0 \approx^{\mathit{div}}_{\mathit{TS}} s_1 \approx^{\mathit{div}}_{\mathit{TS}} \ldots \approx^{\mathit{div}}_{\mathit{TS}} s_n$$

- Corollary: for finite *TS* and state s in *TS*:

s is $\approx^{\mathit{div}}$ –divergent if and only if

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

Removal of stutter cycles: How?

1. Determine the SCCs in $G(\mathit{TS})$ that only contain stutter steps
   - use depth-first search to find these strongly connected components (SCCs)

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

$\Rightarrow$ Resulting TS' has no stutter cycles

- $s_1 \approx_{\mathit{TS}} s_2$ if and only if $\underbrace{C_1}_{s_1 \in C_1} \approx_{\mathit{TS}'} \underbrace{C_2}_{s_2 \in C_2}$

from now on, assume transition systems have no stutter cycles

A “local” splitter characterization

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

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

- How to avoid the 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:

$$\mathit{Bottom}(B) = \left\{ s \in B \mid \mathit{Post}(s) \cap B = \varnothing \right\}$$

- 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

$\boxed{\begin{array}{c}\text{The partition-refinement algorithm to compute } \mathit{TS}/\approx \\ \text{has a worst-case time complexity in } \mathcal{O}\Big(|S| \cdot (|\mathit{AP}| + M)\Big)\end{array}}$

Approach

1. A quotienting algorithm to determine $TS/\approx$:

   - remove *stutter cycles* from TS
   - a refine operator to *efficiently split* (blocks of) partitions
   - exploit partition-refinement (as for bisimulation $\sim$)

$\Rightarrow$ A quotienting algorithm to determine $TS/\approx^{div}$:

- *transform* TS into a (divergence-sensitive) transition system $\overline{TS}$
- $\overline{TS}$ is divergent-sensitive, i.e., $\approx_{\overline{TS}}$ 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

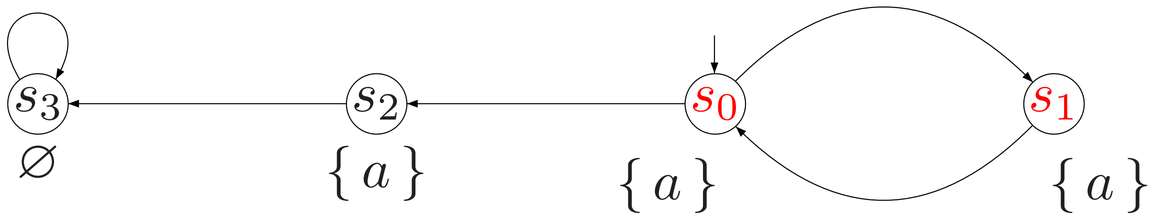
Divergence-sensitive expansion of finite $TS = (S, Act, \rightarrow, I, AP, L)$ is:

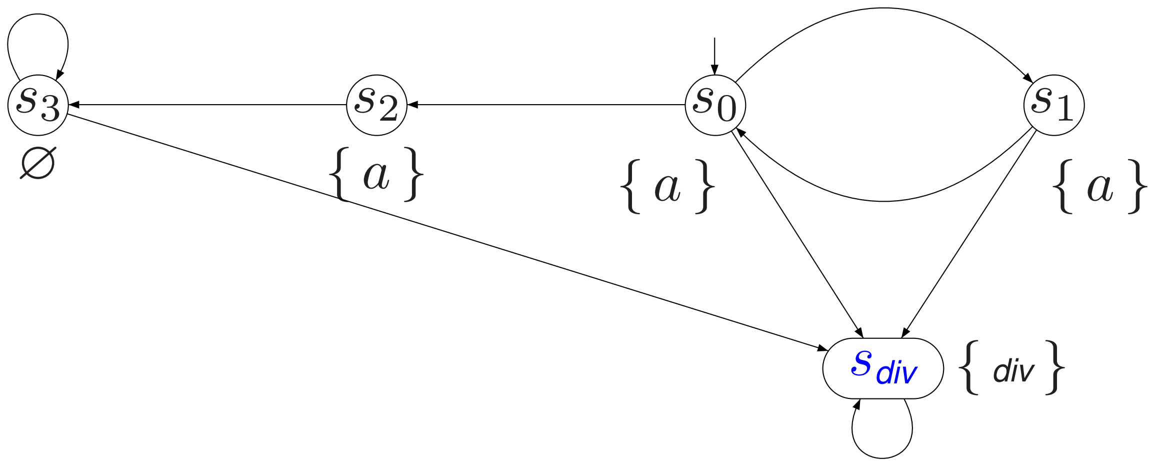
$$\overline{TS} = \left(S \cup \{ s_{div} \}, Act \cup \{ \tau \}, \rightarrow, I, AP \cup \{ div \}, \overline{L}\right) \quad \text{where}$$

- $s_{div} \notin S$
- $\rightarrow$ extends the transition relation of TS 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





Correctness

For finite transition system *TS*:

1. $\overline{TS}$ is divergence-sensitive, and
2. for all $s_1, s_2 \in S$: $s_1 \;\approx_{TS}^{div}\; s_2$ if and only if $s_1 \;\approx_{\overline{TS}}\; s_2$

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

1. Construct the divergence-sensitive expansion $\overline{TS}$
   - determine the SCCs in $G_{stutter}(TS)$, and insert transitions $s_{div} \to s_{div}$ 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 $S/\approx_{TS}^{div} = S/\approx_{\overline{TS}}$

3. Generate $\overline{TS}/\approx$
   - 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 $s \in C$
   - any transition $s \to s'$ with $s \not\approx_{TS}^{div} s'$ yields a transition between C_s and $C_{s'}$

4. "Distill" $TS/\approx^{div}$ from $\overline{TS}/\approx$:
   - 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 $\mathcal{O}\Big(\,|S| \cdot (|AP| + M)\Big)$

Summary

formal relation	trace equivalence	bisimulation	simulation
complexity	PSPACE-complete	$\mathcal{O}(M \cdot \log\lvert S\rvert)$	$\mathcal{O}(M \cdot \lvert S\rvert)$
logical fragment	LTL	CTL*	$\forall$CTL*
preservation	strong	strong match	weak match

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