Overview

Introduction
Modelling parallel systems
Linear Time Properties
Regular Properties
Linear Temporal Logic (LTL)
Computation-Tree Logic
Equivalences and Abstraction
Trace equivalence

$\mathcal{I}_1$: $\mathcal{I}_2$: $\hat{=} = \emptyset$

$\hat{=} = \{a\}$

$\hat{=} = \{b\}$
Trace equivalence

\[ T_1: \quad T_2: \]

\[ \begin{align*}
\text{Traces}(T_1) &= \{ \emptyset \emptyset a^\omega, \emptyset \emptyset b^\omega \} = \text{Traces}(T_1) \\
\end{align*} \]
Trace equivalence

\( \mathcal{T}_1 : \)

\( \mathcal{T}_2 : \)

\[ \text{Traces}(\mathcal{T}_1) = \{ \emptyset \emptyset a^\omega, \emptyset \emptyset b^\omega \} = \text{Traces}(\mathcal{T}_1) \]

\( \text{CTL-formula } \Phi = \exists \Box (\exists \Box a \land \exists \Box b) \)
Trace equivalence

\( T_1: \)

\( T_2: \)

\[
|\ (\emptyset \emptyset a^\omega, \emptyset \emptyset b^\omega ) \rangle = \text{Traces}(T_1)
\]

\[
\text{CTL-formula } \Phi = \exists \bigcirc (\exists \bigcirc a \land \exists \bigcirc b)
\]

\( T_1 \not\models \Phi \quad \text{and} \quad T_2 \models \Phi \)
Trace equivalence is not compatible with CTL

\[ \mathcal{T}_1: \]

\[ \mathcal{T}_2: \]

\[ \forall \mathcal{P}, \mathcal{Q} \subseteq \{a, b\} : \mathcal{P} \equiv \mathcal{Q} \quad \Rightarrow \quad \mathcal{P} \equiv \mathcal{Q} \]

\[ \bigcap_{a \in \mathcal{P}} \mathcal{Q} \]
Implementation relations

- for the design of complex systems
  \[\sim\] comparison of 2 transition systems
Implementation relations

- for the **design** of complex systems
  ~≈ comparison of 2 transition systems
- for the **analysis** of complex systems
Implementation relations

- for the **design** of complex systems
  \(\leadsto\) comparison of 2 transition systems
- for the **analysis** of complex systems
  \(\leadsto\) homogeneous model checking approach
Implementation relations

- for the **design** of complex systems
  ~⇒ comparison of 2 transition systems

- for the **analysis** of complex systems
  ~⇒ homogeneous model checking approach
  ~⇒ graph minimization
Implementation relations

- for the **design** of complex systems
  \[ \sim \rightarrow \text{comparison of 2 transition systems} \]

- for the **analysis** of complex systems
  \[ \sim \rightarrow \text{homogeneous model checking approach} \]
  \[ \sim \rightarrow \text{graph minimization} \]

\[ \text{use equivalence relation } \sim \text{ for the states of a single transition system } \mathcal{T} \text{ and analyze the quotient } \mathcal{T}/\sim \]
Implementation relations

- for the **design** of complex systems
  ⇝ comparison of 2 transition systems

- for the **analysis** of complex systems
  ⇝ homogeneous model checking approach
  ⇝ graph minimization

**goal:** define the equivalence relation \( \sim \) in such a way that

\[
\mathcal{T} \models \Phi \iff \mathcal{T}/\sim \models \Phi
\]

for all “relevant” properties \( \Phi \)

use equivalence relation \( \sim \) for the states of a single transition system \( \mathcal{T} \) and analyze the quotient \( \mathcal{T}/\sim \)
Linear-time implementation relations
finite trace inclusion and equivalence:
  e.g., $\text{Traces}_{\text{fin}}(T_1) \subseteq \text{Traces}_{\text{fin}}(T_2)$

trace inclusion and trace equivalence:
  e.g., $\text{Traces}(T_1) \subseteq \text{Traces}(T_2)$
Linear-time implementation relations

finite trace inclusion and equivalence:
  e.g., $\text{Tracesfin}(T_1) \subseteq \text{Tracesfin}(T_2)$
  preserves all linear-time safety properties

trace inclusion and trace equivalence:
  e.g., $\text{Traces}(T_1) \subseteq \text{Traces}(T_2)$
Linear-time implementation relations

finite trace inclusion and equivalence:
  e.g., \( \text{Traces}_{\text{fin}}(I_1) \subseteq \text{Traces}_{\text{fin}}(I_2) \)
  preserves all linear-time safety properties

trace inclusion and trace equivalence:
  e.g., \( \text{Traces}(I_1) \subseteq \text{Traces}(I_2) \)
  preserves all LTL properties
Linear-time implementation relations

finite trace inclusion and equivalence:
e.g., $\text{Traces}_{\text{fin}}(T_1) \subseteq \text{Traces}_{\text{fin}}(T_2)$
preserves all linear-time safety properties

trace inclusion and trace equivalence:
e.g., $\text{Traces}(T_1) \subseteq \text{Traces}(T_2)$
preserves all LTL properties

* none of the LT relations is compatible with CTL
Linear-time implementation relations

finite trace inclusion and equivalence:
   e.g., \( \text{Traces}_{\text{fin}}(T_1) \subseteq \text{Traces}_{\text{fin}}(T_2) \)
   preserves all linear-time safety properties

trace inclusion and trace equivalence:
   e.g., \( \text{Traces}(T_1) \subseteq \text{Traces}(T_2) \)
   preserves all LTL properties

* none of the LT relations is compatible with CTL
* checking LT relations is computationally hard
Linear-time implementation relations

finite trace inclusion and equivalence:
  e.g., \( \text{Tracesfin}(I_1) \subseteq \text{Tracesfin}(I_2) \)
  preserves all linear-time safety properties

trace inclusion and trace equivalence:
  e.g., \( \text{Traces}(I_1) \subseteq \text{Traces}(I_2) \)
  preserves all LTL properties

* none of the LT relations is compatible with CTL
* checking LT relations is computationally hard
* minimization ???
Minimization w.r.t. trace equivalence?

$\mathcal{T}_1$:  

$\mathcal{T}_2$:  

BSEQOR5.1-min-LT
Minimization w.r.t. trace equivalence?

\[ T_1: \quad \text{Traces}(T_1) = \text{Traces}(T_2) \]

\[ T_2: \]
Minimization w.r.t. trace equivalence?

\( \mathcal{T}_1: \)

\( \mathcal{T}_2: \)

- \( \text{Traces}(\mathcal{T}_1) = \text{Traces}(\mathcal{T}_2) \)

but \( \mathcal{T}_1 \) and \( \mathcal{T}_2 \) are not isomorphic
Minimization w.r.t. trace equivalence?

\[ \mathcal{T}_1: \quad \mathcal{T}_2: \]

- \( \text{Traces}(\mathcal{T}_1) = \text{Traces}(\mathcal{T}_2) \)
  but \( \mathcal{T}_1 \) and \( \mathcal{T}_2 \) are not isomorphic

- \( \mathcal{T}_1, \mathcal{T}_2 \) have 5 states and 7 transitions each
Minimization w.r.t. trace equivalence?

\( T_1: \)

\( T_2: \)

- \( \text{Traces}(T_1) = \text{Traces}(T_2) \)
  but \( T_1 \) and \( T_2 \) are not isomorphic

- \( T_1, T_2 \) have 5 states and 7 transitions each

- there is no smaller TS that is trace-equivalent to \( T_i \)
Classification of implementation relations
Classification of implementation relations

- Linear vs. branching time
  - Linear time: trace relations
  - Branching time: (bi)simulation relations
Classification of implementation relations

- **linear vs. branching time**
  - linear time: trace relations
  - branching time: (bi)simulation relations

- **(nonsymmetric) preorders vs. equivalences**:
  - preorders: trace inclusion, simulation
  - equivalences: trace equivalence, bisimulation
Classification of implementation relations

- **linear vs. branching time**
  - linear time: trace relations
  - branching time: (bi)simulation relations

- **(nonsymmetric) preorders vs. equivalences:**
  - preorders: trace inclusion, simulation
  - equivalences: trace equivalence, bisimulation

- **strong vs. weak relations**
  - strong: reasoning about all transitions
  - weak: abstraction from stutter steps
Overview

Introduction
Modelling parallel systems
Linear Time Properties
Regular Properties
Linear Temporal Logic (LTL)
Computation-Tree Logic

**Equivalences and Abstraction**

bisimulation
CTL, CTL*-equivalence
computing the bisimulation quotient
abstraction stutter steps
simulation relations
Bisimulation for two transition systems
Bisimulation for two transition systems

let $\mathcal{T}_1 = (S_1, Act_1, \rightarrow_1, S_{0,1}, AP, L_1)$,
$\mathcal{T}_2 = (S_2, Act_2, \rightarrow_2, S_{0,2}, AP, L_2)$

be two transition systems
Bisimulation for two transition systems

let $\mathcal{T}_1 = (S_1, \text{Act}_1, \rightarrow_1, S_{0,1}, AP, L_1)$,
$\mathcal{T}_2 = (S_2, \text{Act}_2, \rightarrow_2, S_{0,2}, AP, L_2)$

be two transition systems

• with the same set $AP$
Bisimulation for two transition systems

let \( T_1 = (S_1, \text{Act}_1, \rightarrow_1, S_{0,1}, \text{AP}, L_1) \),
\( T_2 = (S_2, \text{Act}_2, \rightarrow_2, S_{0,2}, \text{AP}, L_2) \)

be two transition systems

- with the same set \( \text{AP} \)
- possibly containing terminal states
Bisimulation for two transition systems

let $T_1 = (S_1, Act_1, \rightarrow_1, S_{0,1}, AP, L_1)$,
$T_2 = (S_2, Act_2, \rightarrow_2, S_{0,2}, AP, L_2)$

be two transition systems

• with the same set $AP$

• possibly containing terminal states

Bisimulation equivalence of $T_1$ and $T_2$ requires that $T_1$ and $T_2$ can simulate each other in a stepwise manner.
Bisimulation for two transition systems

let $T_1 = (S_1, \text{Act}_1, \rightarrow_1, S_0,1, AP, L_1)$,
$T_2 = (S_2, \text{Act}_2, \rightarrow_2, S_0,2, AP, L_2)$

be two transition systems

• with the same set $AP$ observables

• possibly containing terminal states

Bisimulation equivalence of $T_1$ and $T_2$ requires that $T_1$ and $T_2$ can simulate each other in a stepwise manner.
Bisimulation for \((\mathcal{T}_1, \mathcal{T}_2)\)
Bisimulation for \((\mathcal{T}_1, \mathcal{T}_2)\)

binary relation \(\mathcal{R} \subseteq S_1 \times S_2\) s.t. for all \((s_1, s_2) \in \mathcal{R}\):
Bisimulation for \((\mathcal{T}_1, \mathcal{T}_2)\)

binary relation \(\mathcal{R} \subseteq S_1 \times S_2\) s.t. for all \((s_1, s_2) \in \mathcal{R}\):

\[(1) \quad L_1(s_1) = L_2(s_2)\]
Bisimulation for \((\mathcal{T}_1, \mathcal{T}_2)\)

binary relation \(\mathcal{R} \subseteq S_1 \times S_2\) s.t. for all \((s_1, s_2) \in \mathcal{R}\):

1. \(L_1(s_1) = L_2(s_2)\)

2. \(\forall s'_1 \in \text{Post}(s_1) \exists s'_2 \in \text{Post}(s_2)\) s.t. \((s'_1, s'_2) \in \mathcal{R}\)
Bisimulation for \((\mathcal{T}_1, \mathcal{T}_2)\)

binary relation \(R \subseteq S_1 \times S_2\) s.t. for all \((s_1, s_2) \in R:\)

1. \(L_1(s_1) = L_2(s_2)\)
2. \(\forall s'_1 \in \text{Post}(s_1) \exists s'_2 \in \text{Post}(s_2)\) s.t. \((s'_1, s'_2) \in R\)

\(s_1\) \hspace{1cm} \(s_2\) \hspace{1cm} can be completed to

\(s'_1\) \hspace{1cm} \(s'_2\)
Bisimulation for \((\mathcal{T}_1, \mathcal{T}_2)\)

binary relation \(\mathcal{R} \subseteq S_1 \times S_2\) s.t. for all \((s_1, s_2) \in \mathcal{R}\):

1. \(L_1(s_1) = L_2(s_2)\)

2. \(\forall s'_1 \in \text{Post}(s_1) \exists s'_2 \in \text{Post}(s_2)\) s.t. \((s'_1, s'_2) \in \mathcal{R}\)

3. \(\forall s'_2 \in \text{Post}(s_2) \exists s'_1 \in \text{Post}(s_1)\) s.t. \((s'_1, s'_2) \in \mathcal{R}\)
Bisimulation for \((\mathcal{I}_1, \mathcal{I}_2)\)

binary relation \(\mathcal{R} \subseteq S_1 \times S_2\) s.t. for all \((s_1, s_2) \in \mathcal{R}\):

1. \(L_1(s_1) = L_2(s_2)\)

2. \(\forall s'_1 \in \text{Post}(s_1) \exists s'_2 \in \text{Post}(s_2)\) s.t. \((s'_1, s'_2) \in \mathcal{R}\)

\[
\begin{array}{ccc}
  s_1 & \sim_{\mathcal{R}} & s_2 \\
  \downarrow & & \downarrow \\
  s'_1 & & s'_2
\end{array}
\]

can be completed to

\[
\begin{array}{ccc}
  s_1 & \sim_{\mathcal{R}} & s_2 \\
  \downarrow & & \downarrow \\
  s'_1 & \sim_{\mathcal{R}} & s'_2
\end{array}
\]

3. \(\forall s'_2 \in \text{Post}(s_2) \exists s'_1 \in \text{Post}(s_1)\) s.t. \((s'_1, s'_2) \in \mathcal{R}\)

and such that the following initial condition holds:

(I) \(\forall s_{0,1} \in S_{0,1} \exists s_{0,2} \in S_{0,2}\) s.t. \((s_{0,1}, s_{0,2}) \in \mathcal{R}\)
Bisimulation for \((\mathcal{T}_1, \mathcal{T}_2)\)

binary relation \(\mathcal{R} \subseteq S_1 \times S_2\) s.t. for all \((s_1, s_2) \in \mathcal{R}\):

1. \(L_1(s_1) = L_2(s_2)\)
2. \(\forall s'_1 \in Post(s_1) \exists s'_2 \in Post(s_2)\) s.t. \((s'_1, s'_2) \in \mathcal{R}\)

\[
\begin{array}{ccc}
  s_1 & \mathcal{R}^- & s_2 \\
  \downarrow & & \downarrow \\
  s'_1 & & s'_2
\end{array}
\]

can be completed to

\[
\begin{array}{ccc}
  s_1 & \mathcal{R}^- & s_2 \\
  \downarrow & & \downarrow \\
  s'_1 & \mathcal{R}^- & s'_2
\end{array}
\]

3. \(\forall s'_2 \in Post(s_2) \exists s'_1 \in Post(s_1)\) s.t. \((s'_1, s'_2) \in \mathcal{R}\)

and such that the following initial condition holds:

(I) \(\forall s_{0,1} \in S_{0,1} \exists s_{0,2} \in S_{0,2}\) s.t. \((s_{0,1}, s_{0,2}) \in \mathcal{R}\)

\(\forall s_{0,2} \in S_{0,2} \exists s_{0,1} \in S_{0,1}\) s.t. \((s_{0,1}, s_{0,2}) \in \mathcal{R}\)
Bisimulation equivalence ~
Bisimulation equivalence \( \sim \)

Bisimulation for \((\mathcal{T}_1, \mathcal{T}_2)\): relation \( \mathcal{R} \subseteq S_1 \times S_2 \) s.t.

for all \((s_1, s_2) \in \mathcal{R}\):

1. labeling condition
2. mutual stepwise simulation
3. initial condition

and initial condition (I)
Bisimulation equivalence $\sim$

bisimulation for $\left( T_1, T_2 \right)$: relation $R \subseteq S_1 \times S_2$ s.t.

for all $(s_1, s_2) \in R$: (1) labeling condition (2) mutual stepwise simulation (3) initial condition

and initial condition (I)

bisimulation equivalence $\sim$ for TS:
Bisimulation equivalence ~

bisimulation for \((T_1, T_2)\): relation \(R \subseteq S_1 \times S_2\) s.t.

for all \((s_1, s_2) \in R\):

1. labeling condition
2. mutual stepwise simulation

and initial condition \((I)\)

bisimulation equivalence \(\sim\) for TS:

\[ T_1 \sim T_2 \] iff there is a bisimulation \(R\) for \((T_1, T_2)\)
Bisimulation equivalence \( \sim \)

bisimulation for \((\mathcal{T}_1, \mathcal{T}_2)\): relation \( \mathcal{R} \subseteq S_1 \times S_2 \) s.t.

for all \((s_1, s_2) \in \mathcal{R}\):

1. labeling condition
2. \{ mutual stepwise simulation \}
3. initial condition \((I)\)

and initial condition \((I)\)

bisimulation equivalence \(\sim\) for TS:

\[ \mathcal{T}_1 \sim \mathcal{T}_2 \quad \text{iff} \quad \text{there is a bisimulation} \ \mathcal{R} \ \text{for} \ (\mathcal{T}_1, \mathcal{T}_2) \]

for state \(s_1\) of \(\mathcal{T}_1\) and state \(s_2\) of \(\mathcal{T}_2\):

\[ s_1 \sim s_2 \quad \text{iff} \quad \text{there exists a bisimulation} \ \mathcal{R} \ \text{for} \ (\mathcal{T}_1, \mathcal{T}_2) \]

such that \((s_1, s_2) \in \mathcal{R}\)
Two beverage machines

\[ \mathcal{T}_1 \]

- pay \rightarrow select
- select \rightarrow coke
- select \rightarrow soda

\[ \mathcal{T}_2 \]

- pay \rightarrow select
- select \rightarrow coke_1
- select \rightarrow coke_2
- coke_1 \rightarrow soda
- coke_2 \rightarrow soda

\[ \mathbb{AP} = \{ \text{pay, coke, soda} \} \]
Two beverage machines

$\mathcal{T}_1$

$\text{pay}$

$\text{select}$

$\text{coke}$

$\text{soda}$

$\mathcal{T}_2$

$\text{pay}$

$\text{select}$

$\text{coke}_1$

$\text{coke}_2$

$\text{soda}$

$AP = \{ \text{pay}, \text{coke}, \text{soda} \}$
Two beverage machines

$\mathcal{T}_1$  
\[ \text{pay} \rightarrow \text{select} \]  
\[ \text{coke} \rightarrow \text{select} \]  
\[ \text{soda} \rightarrow \text{select} \]  

$\mathcal{T}_2$  
\[ \text{pay} \rightarrow \text{select} \]  
\[ \text{coke}_1 \rightarrow \text{select} \]  
\[ \text{coke}_2 \rightarrow \text{select} \]  
\[ \text{soda} \rightarrow \text{select} \]  

$\mathcal{AP} = \{ \text{pay}, \text{coke}, \text{soda} \}$

$\mathcal{T}_1 \sim \mathcal{T}_2$
Two beverage machines

\[ \mathcal{T}_1 \sim \mathcal{T}_2 \] as there is a bisimulation for \((\mathcal{T}_1, \mathcal{T}_2)\):

\[ AP = \{ \text{pay, coke, soda} \} \]
Two beverage machines

$$\mathcal{T}_1 \sim \mathcal{T}_2$$ as there is a bisimulation for $$(\mathcal{T}_1, \mathcal{T}_2)$$:

$$\{ (\text{pay,pay}), (\text{select,select}), (\text{soda,soda}), (\text{coke,coke}_1), (\text{coke,coke}_2) \}$$

$$\mathcal{T}_1$$
- pay
- select
- coke
- soda

$$\mathcal{T}_2$$
- pay
- select
- coke
- coke
- soda

$$AP = \{ \text{pay}, \text{coke}, \text{soda} \}$$
Two beverage machines

\[ T_1 \]

\[ \mathcal{T}_1 \]

\[ \text{pay} \]

\[ \text{paid}_1 \]

\[ \text{coke} \]

\[ \text{paid}_2 \]

\[ \text{soda} \]

\[ T_2 \]

\[ \mathcal{T}_2 \]

\[ \text{pay} \]

\[ \text{select} \]

\[ \text{coke} \]

\[ \text{soda} \]

\[ AP = \{ pay, coke, soda \} \]
Two beverage machines

$T_1$  

$T_2$

$AP = \{\text{pay, coke, soda}\}$

$T_1 \not \sim T_2$
Two beverage machines

\[ \mathcal{T}_1 \]
\[ \mathcal{T}_1 \xrightarrow{} \text{pay} \]
\[ \text{paid}_1 \]
\[ \text{paid}_2 \]
\[ \text{coke} \]
\[ \text{soda} \]

\[ \mathcal{T}_2 \]
\[ \mathcal{T}_2 \xrightarrow{} \text{pay} \]
\[ \text{select} \]
\[ \text{coke} \]
\[ \text{soda} \]

\[ \text{AP} = \{ \text{pay}, \text{coke}, \text{soda} \} \]

\[ \mathcal{T}_1 \not\sim \mathcal{T}_2 \]

because there is no state in \( \mathcal{T}_1 \) that has both

- a successor labeled with \( \text{coke} \) and
- a successor labeled with \( \text{soda} \)
Simulation condition of bisimulations

\[
\begin{array}{c}
\mathcal{s}_1 \xrightarrow{-R-} \mathcal{s}_2 \\
\downarrow \\
\mathcal{s}_1'
\end{array}
\hspace{1cm}
\begin{array}{c}
\mathcal{s}_1 \xrightarrow{-R-} \mathcal{s}_2 \\
\downarrow \\
\mathcal{s}_1' \xrightarrow{-R-} \mathcal{s}_2'
\end{array}
\]

can be completed to
Path lifting for bisimulation $\mathcal{R}$
Path lifting for bisimulation $\mathcal{R}$

can be completed to

$\mathcal{R}$

$\downarrow$

$S_{1,1}$

$\downarrow$

$S_{1,2}$

$\downarrow$

$S_{1,3}$

$\downarrow$

$S_{1,4}$

$\vdots$
Path lifting for bisimulation $\mathcal{R}$

\[ s_1 \xrightarrow{\neg \mathcal{R}} s_2 \]

\[ \downarrow \]

\[ s_{1,1} \]

\[ \downarrow \]

\[ s_{1,2} \]

\[ \downarrow \]

\[ s_{1,3} \]

\[ \downarrow \]

\[ s_{1,4} \]

\[ \vdots \]

\[ \downarrow \]

\[ s_1,4 \]

\[ \vdots \]

\[ s_1,3 \]

\[ \vdots \]

\[ s_1,2 \]

\[ \vdots \]

\[ s_1,1 \]

\[ \vdots \]

can be completed to

\[ s_1 \xrightarrow{\neg \mathcal{R}} s_2 \]

\[ \downarrow \]

\[ s_{1,1} \xrightarrow{\neg \mathcal{R}} s_{2,1} \]

\[ \downarrow \]

\[ s_{1,2} \]

\[ \downarrow \]

\[ s_{1,3} \]

\[ \downarrow \]

\[ s_{1,4} \]

\[ \vdots \]
Path lifting for bisimulation $\mathcal{R}$

can be completed to
Path lifting for bisimulation $\mathcal{R}$

can be completed to
Path lifting for bisimulation $\mathcal{R}$

<table>
<thead>
<tr>
<th>$s_1$</th>
<th>$\mathcal{R}$</th>
<th>$s_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\downarrow$</td>
<td>$s_{1,1}$</td>
<td>$\downarrow$</td>
</tr>
<tr>
<td>$\downarrow$</td>
<td>$s_{1,2}$</td>
<td>$\downarrow$</td>
</tr>
<tr>
<td>$\downarrow$</td>
<td>$s_{1,3}$</td>
<td>$\downarrow$</td>
</tr>
<tr>
<td>$\downarrow$</td>
<td>$s_{1,4}$</td>
<td>$\downarrow$</td>
</tr>
<tr>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
</tr>
</tbody>
</table>

can be completed to

<table>
<thead>
<tr>
<th>$s_1$</th>
<th>$\mathcal{R}$</th>
<th>$s_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\downarrow$</td>
<td>$s_{1,1}$</td>
<td>$\downarrow$</td>
</tr>
<tr>
<td>$\downarrow$</td>
<td>$s_{1,2}$</td>
<td>$\downarrow$</td>
</tr>
<tr>
<td>$\downarrow$</td>
<td>$s_{1,3}$</td>
<td>$\downarrow$</td>
</tr>
<tr>
<td>$\downarrow$</td>
<td>$s_{1,4}$</td>
<td>$\downarrow$</td>
</tr>
<tr>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
</tr>
</tbody>
</table>
Properties of bisimulation equivalence
Properties of bisimulation equivalence

~ is an equivalence
Properties of bisimulation equivalence

is an equivalence, i.e.,

- reflexivity: for all transition systems
Properties of bisimulation equivalence

\( \sim \) is an equivalence, i.e.,

- reflexivity: \( \mathcal{T} \sim \mathcal{T} \) for all transition systems \( \mathcal{T} \)

If \( S \) is the state space of \( \mathcal{T} \) then

\[ \mathcal{R} = \{(s, s) : s \in S\} \]

is a bisimulation for \( (\mathcal{T}, \mathcal{T}) \)
~ is an equivalence, i.e.,

- reflexivity: $\mathcal{T} \sim \mathcal{T}$ for all transition systems $\mathcal{T}$
- symmetry: $\mathcal{T}_1 \sim \mathcal{T}_2$ implies $\mathcal{T}_2 \sim \mathcal{T}_1$
Properties of bisimulation equivalence

\(~\) is an equivalence, i.e.,

- reflexivity: \( \mathcal{T} \sim \mathcal{T} \) for all transition systems \( \mathcal{T} \)
- symmetry: \( \mathcal{T}_1 \sim \mathcal{T}_2 \) implies \( \mathcal{T}_2 \sim \mathcal{T}_1 \)

If \( \mathcal{R} \) is a bisimulation for \( (\mathcal{T}_1, \mathcal{T}_2) \) then

\[
\mathcal{R}^{-1} = \{(s_2, s_1) : (s_1, s_2) \in \mathcal{R}\}
\]

is a bisimulation for \( (\mathcal{T}_2, \mathcal{T}_1) \)
Properties of bisimulation equivalence

\( \sim \) is an equivalence, i.e.,

- **reflexivity**: \( T \sim T \) for all transition systems \( T \)
- **symmetry**: \( T_1 \sim T_2 \) implies \( T_2 \sim T_1 \)
- **transitivity**: if \( T_1 \sim T_2 \) and \( T_2 \sim T_3 \) then \( T_1 \sim T_3 \)
Properties of bisimulation equivalence

\sim is an equivalence, i.e.,

- reflexivity: \( T \sim T \) for all transition systems \( T \)
- symmetry: \( T_1 \sim T_2 \) implies \( T_2 \sim T_1 \)
- transitivity: if \( T_1 \sim T_2 \) and \( T_2 \sim T_3 \) then \( T_1 \sim T_3 \)

Let \( R_{1,2} \) be a bisimulation for \((T_1, T_2)\),
\( R_{2,3} \) be a bisimulation for \((T_2, T_3)\).
Properties of bisimulation equivalence

\( \sim \) is an equivalence, i.e.,

- reflexivity: \( T \sim T \) for all transition systems \( T \)
- symmetry: \( T_1 \sim T_2 \) implies \( T_2 \sim T_1 \)
- transitivity: if \( T_1 \sim T_2 \) and \( T_2 \sim T_3 \) then \( T_1 \sim T_3 \)

Let \( R_{1,2} \) be a bisimulation for \( (T_1, T_2) \),
\( R_{2,3} \) be a bisimulation for \( (T_2, T_3) \).

\[
R \overset{\text{def}}{=} \{ (s_1, s_3) : \exists s_2 \text{ s.t. } (s_1, s_2) \in R_{1,2} \text{ and } (s_2, s_3) \in R_{2,3} \}
\]

is a bisimulation for \( (T_1, T_3) \)
Correct or wrong?

![Diagram](image)
Correct or wrong?

\begin{itemize}
  \item Right
  \item Wrong
\end{itemize}
Correct or wrong?

$s_1 \rightarrow u$, but $s_2 \not\rightarrow \text{blue}$ (thus $s_1 \not\sim s_2$)
Correct or wrong?

\[ s_1 \rightarrow u, \text{ but } s_2 \not\rightarrow \text{ blue} \quad \text{(thus } s_1 \not \sim s_2 \text{)} \]

\[ s_1 \not\sim s_2 \]
Correct or wrong?

\( s_1 \rightarrow u, \) but \( s_2 \not\rightarrow blue \) (thus \( s_1 \not\sim s_2 \))

---

\( s_1 \sim s_2 \)

---

\( s_1 \not\sim s_2 \)
Correct or wrong?

\[ s_1 \xrightarrow{u} \text{blue} \text{ (thus } s_1 \not\sim s_2 \text{)} \]

\[ \{ (w_1, w_2), (w_1', w_2), (s_1, s_2), (s_1, s_2'), (u, x), (u, y) \} \]
Correct or wrong?

\[ \sim \]
Correct or wrong?

BSEQOR5.1-20

~

correct
Correct or wrong?

\begin{align*}
\text{bisimulation} & \quad \{ (s_1, s_2), (s_1', s_2'), (s_1', s_2''), (t_1, t_2), (t_1', t_2), (t_1'', t_2) \} \\
\end{align*}
Correct or wrong?

\[ \text{bisimulation} \]
\[ \{(s_1, s_2), (s'_1, s'_2), (s'_1, s''_2), (t_1, t_2), (t'_1, t_2), (t''_1, t_2)\} \]
Correct or wrong?

\[
\text{bisimulation } \{ (s_1, s_2), (s'_1, s'_2), (s'_1, s''_2), (t_1, t_2), (t'_1, t_2), (t''_1, t_2) \}
\]
Correct or wrong?

bisimulation: \{ (s_1, s_2), (t_1, t_2), (t_1', t_2), (u_1, u_2), (v_1, v_2) \}

correct
Bisimulation vs. trace equivalence
Bisimulation vs. trace equivalence

\[ \mathcal{I}_1 \sim \mathcal{I}_2 \implies \text{Traces}(\mathcal{I}_1) = \text{Traces}(\mathcal{I}_2) \]
Bisimulation vs. trace equivalence

\[ \mathcal{I}_1 \sim \mathcal{I}_2 \implies \text{Traces}(\mathcal{I}_1) = \text{Traces}(\mathcal{I}_2) \]

proof: ... path fragment lifting ...
Bisimulation vs. trace equivalence

\[ T_1 \sim T_2 \implies \text{Traces}(T_1) = \text{Traces}(T_2) \]

proof: ... path fragment lifting ... 

\[ \text{Traces}(T_1) = \text{Traces}(T_2) \not\iff T_1 \sim T_2 \]
Bisimulation vs. trace equivalence

\[ \mathcal{T}_1 \sim \mathcal{T}_2 \implies \text{Traces}(\mathcal{T}_1) = \text{Traces}(\mathcal{T}_2) \]

proof: ... path fragment lifting ...

\[ \text{Traces}(\mathcal{T}_1) = \text{Traces}(\mathcal{T}_2) \not\iff \mathcal{T}_1 \sim \mathcal{T}_2 \]

trace equivalent, but not bisimulation equivalent
Bisimulation vs. trace equivalence

\[ \mathcal{I}_1 \sim \mathcal{I}_2 \implies \text{Traces}(\mathcal{I}_1) = \text{Traces}(\mathcal{I}_2) \]

proof: ... path fragment lifting ...

\[ \text{Traces}(\mathcal{I}_1) = \text{Traces}(\mathcal{I}_2) \not\iff \mathcal{I}_1 \sim \mathcal{I}_2 \]

Trace equivalence is strictly coarser than bisimulation equivalence.
Bisimulation vs. trace equivalence

\[ T_1 \sim T_2 \implies Traces(T_1) = Traces(T_2) \]

**proof:** ... path fragment lifting ...

\[ Traces(T_1) = Traces(T_2) \not\implies T_1 \sim T_2 \]

---

Trace equivalence is strictly coarser than bisimulation equivalence.

---

Bisimulation equivalent transition systems satisfy the same LT properties (e.g., LTL formulas).
Bisimulation equivalence ...

- as a relation that compares 2 transition systems
Bisimulation equivalence ... 

- as a relation that compares 2 transition systems
Bisimulation equivalence ...

- as a relation that compares 2 transition systems

- as a relation on the states of 1 transition system
Bisimulation equivalence ...

- as a relation that compares 2 transition systems

- as a relation on the states of 1 transition system
Bisimulation equivalence ...

- as a relation that compares 2 transition systems

\[ T_1 \sim T_2 \text{ iff } s_1 \sim s_2 \]

- as a relation on the states of 1 transition system

\[ s_1 \sim s_2 \text{ iff } T_{s_1} \sim T_{s_2} \]
Bisimulation equivalence...

- as a relation that compares 2 transition systems

\[ T_1 \sim T_2 \]

- as a relation on the states of 1 transition system

\[ s_1 \sim s_2 \iff T_{s_1} \sim T_{s_2} \]
Bisimulation equivalence ...

- as a relation that compares 2 transition systems

\[ T_1 \sim T_2 \]

- as a relation on the states of 1 transition system

\[ s_1 \sim s_2 \iff T_{s_1} \sim T_{s_2} \]

iff there exists a bisimulation \( R \) for \( T \) s.t. \( (s_1, s_2) \in R \)
Bisimulations on a single TS
Let $\mathcal{T}$ be a TS with proposition set $\mathcal{AP}$. 
Bisimulations on a single TS

Let $T$ be a TS with proposition set $AP$.

A bisimulation for $T$ is a binary relation $R$ on the state space of $T$ s.t. for all $(s_1, s_2) \in R$:

1. $L(s_1) = L(s_2)$
2. $\forall s'_1 \in \text{Post}(s_1) \exists s'_2 \in \text{Post}(s_2)$ s.t. $(s'_1, s'_2) \in R$
3. $\forall s'_2 \in \text{Post}(s_2) \exists s'_1 \in \text{Post}(s_1)$ s.t. $(s'_1, s'_2) \in R$
Bisimulation equivalence \( \sim_T \) on a single TS

Let \( T \) be a TS with proposition set \( AP \).

A bisimulation for \( T \) is a binary relation \( R \) on the state space of \( T \) s.t. for all \((s_1, s_2) \in R\):

1. \( L(s_1) = L(s_2) \)
2. \( \forall s'_1 \in Post(s_1) \exists s'_2 \in Post(s_2) \text{ s.t. } (s'_1, s'_2) \in R \)
3. \( \forall s'_2 \in Post(s_2) \exists s'_1 \in Post(s_1) \text{ s.t. } (s'_1, s'_2) \in R \)

bisimulation equivalence \( \sim_T \):

\[ s_1 \sim_T s_2 \text{ iff there exists a bisimulation } R \text{ for } T \]

s.t. \((s_1, s_2) \in R \)
Bisimulation equivalence $\sim_T$ on a single TS

Let $T$ be a TS with proposition set $AP$.

A bisimulation for $T$ is a binary relation $R$ on the state space of $T$ s.t. for all $(s_1, s_2) \in R$:

1. $L(s_1) = L(s_2)$
2. $\forall s'_1 \in Post(s_1) \exists s'_2 \in Post(s_2)$ s.t. $(s'_1, s'_2) \in R$
3. $\forall s'_2 \in Post(s_2) \exists s'_1 \in Post(s_1)$ s.t. $(s'_1, s'_2) \in R$

coinductive definition of $\sim_T$:

$s_1 \sim_T s_2$ iff there exists a bisimulation $R$ for $T$ s.t. $(s_1, s_2) \in R$
Bisimulation equivalence

Let $\mathcal{T}$ be a transition system with state space $S$.

Bisimulation equivalence $\sim_\mathcal{T}$ is
Let $\mathcal{T}$ be a transition system with state space $S$.

Bisimulation equivalence $\sim_{\mathcal{T}}$ is

- the coarsest bisimulation on $\mathcal{T}$
Bisimulation equivalence

Let $\mathcal{T}$ be a transition system with state space $S$.

Bisimulation equivalence $\sim_{\mathcal{T}}$ is

- the **coarsest bisimulation** on $\mathcal{T}$
- and an **equivalence** on $S$
Bisimulation equivalence

Let $\mathcal{T}$ be a transition system with state space $\mathcal{S}$.

Bisimulation equivalence $\sim_{\mathcal{T}}$ is the coarsest equivalence on $\mathcal{S}$ s.t. for all states $s_1, s_2 \in \mathcal{S}$ with $s_1 \sim_{\mathcal{T}} s_2$:
Bisimulation equivalence

Let $\mathcal{T}$ be a transition system with state space $S$.

Bisimulation equivalence $\sim_{\mathcal{T}}$ is the coarsest equivalence on $S$ s.t. for all states $s_1, s_2 \in S$ with $s_1 \sim_{\mathcal{T}} s_2$:

1. $L(s_1) = L(s_2)$
2. each transition of $s_1$ can be mimicked by a transition of $s_2$:
Bisimulation equivalence

Let $\mathcal{T}$ be a transition system with state space $S$.

Bisimulation equivalence $\sim_\mathcal{T}$ is the coarsest equivalence on $S$ s.t. for all states $s_1, s_2 \in S$ with $s_1 \sim_\mathcal{T} s_2$:

1. $L(s_1) = L(s_2)$
2. each transition of $s_1$ can be mimicked by a transition of $s_2$:

\[
\begin{array}{c}
 s_1 \\
 \downarrow \\
 s'_1
\end{array}
\sim_\mathcal{T}
\begin{array}{c}
 s_2 \\
 \downarrow \\
 s'_2
\end{array}
\]

can be completed to

\[
\begin{array}{c}
 s_1 \\
 \downarrow \\
 s'_1
\end{array}
\sim_\mathcal{T}
\begin{array}{c}
 s_2 \\
 \downarrow \\
 s'_2
\end{array}
\]
Two variants of bisimulation equivalence

\[ \sim \] relation that compares 2 transition systems

\[ \sim_T \] equivalence on the state space of a single TS \( T \)
Two variants of bisimulation equivalence

∼  relation that compares 2 transition systems

∼_T  equivalence on the state space of a single TS \( T \)

1.  ∼_T  can be derived from ∼
Two variants of bisimulation equivalence

∼ relation that compares 2 transition systems
∼_T equivalence on the state space of a single TS _T

1. ∼_T can be derived from ∼

for all states s_1 and s_2 of _T:

s_1 ∼_T s_2 iff _T_{s_1} ∼ _T_{s_1}
Two variants of bisimulation equivalence

\(\sim\) relation that compares 2 transition systems

\(\sim_T\) equivalence on the state space of a single TS \(T\)

1. \(\sim_T\) can be derived from \(\sim\)

for all states \(s_1\) and \(s_2\) of \(T\):

\[ s_1 \sim_T s_2 \text{ iff } T_{s_1} \sim T_{s_1} \]

where \(T_s\) agrees with \(T\), except that state \(s\) is declared to be the unique initial state
Two variants of bisimulation equivalence

\( \sim \) relation that compares 2 transition systems

\( \sim_T \) equivalence on the state space of a single TS \( T \)

1. \( \sim_T \) can be derived from \( \sim \)

\[
\text{for all states } s_1 \text{ and } s_2 \text{ of } T:\n
s_1 \sim_T s_2 \text{ iff } T_{s_1} \sim T_{s_1}
\]

where \( T_s \) agrees with \( T \), except that state \( s \) is declared to be the unique initial state

2. \( \sim \) can be derived from \( \sim_T \)
Derivation of $\sim$ from $\sim_T$

given two transition systems $\mathcal{T}_1$ and $\mathcal{T}_2$

$\mathcal{T}_1$ with state space $S_1$

$\mathcal{T}_2$ with state space $S_2$
Derivation of $\sim$ from $\sim_T$

given two transition systems $\mathcal{T}_1$ and $\mathcal{T}_2$

$\mathcal{T}_1$ with state space $S_1$

$\mathcal{T}_2$ with state space $S_2$

consider $\mathcal{T} = \mathcal{T}_1 \uplus \mathcal{T}_2$

(state space $S_1 \uplus S_2$)
Derivation of $\sim$ from $\sim_T$

given two transition systems $\mathcal{T}_1$ and $\mathcal{T}_2$

$\mathcal{T}_1$ with state space $S_1$

$\mathcal{T}_2$ with state space $S_2$

consider $\mathcal{T} = \mathcal{T}_1 \cup \mathcal{T}_2$

(state space $S_1 \cup S_2$)

$\mathcal{T}_1 \sim \mathcal{T}_2$ iff $\forall$ initial states $s_1$ of $\mathcal{T}_1$

$\exists$ initial state $s_2$ of $\mathcal{T}_2$ s.t. $s_1 \sim_T s_2$, 
Derivation of $\sim$ from $\sim_T$

given two transition systems $\mathcal{T}_1$ and $\mathcal{T}_2$

$\mathcal{T}_1$ with state space $S_1$

$\mathcal{T}_2$ with state space $S_2$

consider $\mathcal{T} = \mathcal{T}_1 \sqcup \mathcal{T}_2$
(state space $S_1 \sqcup S_2$)

$\mathcal{T}_1 \sim \mathcal{T}_2$ iff $\forall$ initial states $s_1$ of $\mathcal{T}_1$
exists initial state $s_2$ of $\mathcal{T}_2$ s.t. $s_1 \sim_T s_2$, and vice versa
Bisimulation quotient
Let $\mathcal{T} = (S, \text{Act}, \rightarrow, S_0, AP, L)$ be a TS.
Bisimulation quotient

Let $\mathcal{T} = (S, Act, \rightarrow, S_0, AP, L)$ be a TS.

bisimulation quotient $\mathcal{T}/\sim$ arises from $\mathcal{T}$
by collapsing bisimulation equivalent states
Let $\mathcal{T} = (S, Act, \rightarrow, S_0, AP, L)$ be a TS.

bisimulation quotient:

$\mathcal{T} / \sim = (S', Act', \rightarrow', S'_0, AP, L')$
Let \( T = (S, \text{Act}, \rightarrow, S_0, AP, L) \) be a TS.

bisimulation quotient:
\[
T / \sim = (S', \text{Act}', \rightarrow', S'_0, AP, L')
\]

- state space: \( S' = S / \sim_T \)

set of bisimulation equivalence classes
Bisimulation quotient

Let $\mathcal{T} = (S, Act, \rightarrow, S_0, AP, L)$ be a TS.

bisimulation quotient:

$\mathcal{T}/\sim = (S', Act', \rightarrow', S'_0, AP, L')$

- state space: $S' = S/\sim_T$
- set of initial states: $S'_0 = \{[s]_{\sim_T} : s \in S_0\}$
Let $\mathcal{T} = (S, \text{Act}, \rightarrow, S_0, AP, L)$ be a TS.

**Bisimulation quotient:**

$\mathcal{T} / \sim = (S', \text{Act}', \rightarrow', S_0', AP, L')$

- **state space:** $S' = S / \sim_{\mathcal{T}}$
- **set of initial states:** $S_0' = \{ [s]_{\sim_{\mathcal{T}}} : s \in S_0 \}$
- **labeling function:** $L'([s]_{\sim_{\mathcal{T}}}) = L(s)$
Let $\mathcal{T} = (S, \text{Act}, \rightarrow, S_0, \text{AP}, L)$ be a TS.

bisimulation quotient:

$\mathcal{T}/\sim = (S', \text{Act}', \rightarrow', S'_0, \text{AP}, L')$

- state space: $S' = S/\sim_T$
- set of initial states: $S'_0 = \{[s]_\sim_T : s \in S_0\}$
- labeling function: $L'([s]_\sim_T) = L(s)$

well-defined by the labeling condition of bisimulations
Let $\mathcal{T} = (S, \text{Act}, \rightarrow, S_0, AP, L)$ be a TS.

**Bisimulation quotient:**

$$\mathcal{T} / \sim = (S', \text{Act}', \rightarrow', S'_0, AP, L')$$

- **state space:** $S' = S / \sim_T$
- **set of initial states:** $S'_0 = \{ [s]_\sim_T : s \in S_0 \}$
- **labeling function:** $L'( [s]_\sim_T ) = L(s)$
- **transition relation:**

  $s \xrightarrow{\text{Transition}} s'$

  $[s]_\sim_T \xrightarrow{\text{Transition}} [s']_\sim_T$
Let $\mathcal{T} = (S, \text{Act}, \rightarrow, S_0, AP, L)$ be a TS.

bisimulation quotient:

$\mathcal{T}/\sim = (S', \text{Act}', \rightarrow', S'_0, AP, L')$

- state space: $S' = S/\sim_T$
- set of initial states: $S'_0 = \{[s]_\sim_T : s \in S_0\}$
- labeling function: $L'([s]_\sim_T) = L(s)$
- transition relation:

\[
\begin{align*}
[s]_\sim_T \rightarrow [s']_\sim_T & \quad \text{action labels irrelevant}
\end{align*}
\]
Let \( T = (S, \text{Act}, \rightarrow, S_0, \text{AP}, L) \) be a TS.

**Bisimulation quotient:**

\[
T / \sim = (S', \{\tau\}, \rightarrow', S'_0, \text{AP}, L')
\]

- **state space:** \( S' = S / \sim_T \)
- **set of initial states:** \( S'_0 = \{[s]_\sim_T : s \in S_0\} \)
- **labeling function:** \( L'([s]_\sim_T) = L(s) \)
- **transition relation:**
  \[
  s \xrightarrow{\alpha} s' \\
  [s]_\sim_T \xrightarrow{\tau} [s']_\sim_T \\
  \text{action labels irrelevant}
  \]
Let $\mathcal{T} = (S, \text{Act}, \rightarrow, S_0, AP, L)$ be a TS.

bisimulation quotient:

$\mathcal{T} / \sim = (S', \{\tau\}, \rightarrow', S'_0, AP, L')$

• state space: $S' = S / \sim_T$
• set of initial states: $S'_0 = \{[s]_\sim_T : s \in S_0\}$
• labeling function: $L'([s]_\sim_T) = L(s)$
• transition relation:

$\frac{s \xrightarrow{\alpha} s'}{[s]_\sim_T \xrightarrow{\tau} [s']_\sim_T}$
Example: interleaving of $n$ printers

parallel system $\mathcal{T} = \underbrace{\text{Printer} \parallel \text{Printer} \parallel \ldots \parallel \text{Printer}}_{n \text{ printer}}$
Example: interleaving of $n$ printers

parallel system $\mathcal{T} = \text{Printer} \ ||| \ \text{Printer} \ ||| \ldots \ ||| \ \text{Printer}$

$n$ printer

transition system for each printer

ready_to_print

is_printing
Example: interleaving of $n$ printers

Parallel system $\mathcal{T} = \text{Printer} \ || \ |\ | \ldots \ ||\ |\ | \text{Printer}$

$AP = \{0, 1, \ldots, n\}$

"number of available printers"

Transition system for each printer
Example: \( n=3 \) printers

parallel system \( \mathcal{T} = \underbrace{\text{Printer} \ ||| \ldots \ ||| \text{Printer}}_{n \ \text{printer}} \)

\( AP = \{0, 1, 2, 3\} \)

\[ p \quad \text{is printing} \]

\[ r \quad \text{ready to print} \]
Example: $n=3$ printers

parallel system $\mathcal{T} = \text{Printer} \parallel \text{Printer} \parallel \ldots \parallel \text{Printer}$

$AP = \{0, 1, 2, 3\}$

$p$: is printing

$r$: ready to print
Example: $n=3$ printers

Parallel system $\mathcal{T} = \text{Printer} ||\text{Printer} || \ldots || \text{Printer}$

$AP = \{0, 1, 2, 3\}$

$\begin{align*}
p &:\text{ is printing} \\
r &:\text{ ready to print}
\end{align*}$

Bisimulation quotient
Example: \( n=3 \) printers

Parallel system \( \mathcal{T} = \text{Printer} \parallel \text{Printer} \parallel \ldots \parallel \text{Printer} \)

\[ \mathcal{A}P = \{0, 1, 2, 3\} \]

\( 2^n \) states

\( n+1 \) states
Mutual exclusion
Mutual exclusion

solutions for mutual exclusion problems:

- semaphore
- Peterson’s algorithm
Mutual exclusion: Bakery algorithm

solutions for mutual exclusion problems:

- semaphore
- Peterson’s algorithm
- Bakery algorithm
Mutual exclusion: Bakery algorithm

solutions for mutual exclusion problems:

- semaphore
- Peterson’s algorithm
- Bakery algorithm

given two concurrent processes $P_1$ and $P_2$
Mutual exclusion: Bakery algorithm

solutions for mutual exclusion problems:

- semaphore
- Peterson’s algorithm
- Bakery algorithm

given two concurrent processes $P_1$ and $P_2$

- two additional shared variables: $x_1, x_2 \in \mathbb{N}$
Mutual exclusion: Bakery algorithm

solutions for mutual exclusion problems:

- semaphore
- Peterson’s algorithm
- Bakery algorithm

Given two concurrent processes $P_1$ and $P_2$

- two additional shared variables: $x_1, x_2 \in \mathbb{N}$
- if $P_1$ and $P_2$ are waiting then:
solutions for mutual exclusion problems:

- semaphore
- Peterson’s algorithm
- Bakery algorithm

given two concurrent processes $P_1$ and $P_2$

- two additional shared variables: $x_1, x_2 \in \mathbb{N}$
- if $P_1$ and $P_2$ are waiting then:
  
  - if $x_1 < x_2$ then $P_1$ enters its critical section
  - if $x_2 < x_1$ then $P_2$ enters its critical section
solutions for mutual exclusion problems:

- semaphore
- Peterson’s algorithm
- Bakery algorithm

given two concurrent processes $P_1$ and $P_2$

- two additional shared variables: $x_1, x_2 \in \mathbb{N}$
- if $P_1$ and $P_2$ are waiting then:
  - if $x_1 < x_2$ then $P_1$ enters its critical section
  - if $x_2 < x_1$ then $P_2$ enters its critical section
- $x_1 = x_2$: cannot happen
Bakery algorithm

protocol for $P_1$:

```
LOOP FOREVER
    noncritical actions
    $x_1 := x_2 + 1$
    AWAIT ($x_1 < x_2$) V ($x_2 = 0$);
    critical section;
    $x_1 := 0$
END LOOP
```

symmetric protocol for $P_2$
protocol for $P_1$:

\[
\text{LOOP FOREVER}
\]
\[
\text{noncritical actions}
\]
\[
\begin{align*}
x_1 & := x_2 + 1 \\
\text{AWAIT} \ (x_1 < x_2) \lor (x_2 = 0) \\
\text{critical section;}
\end{align*}
\]
\[
\begin{align*}
x_1 & := 0 \\
\end{align*}
\]
\[
\text{END LOOP}
\]

initially:
\[
\begin{align*}
x_1 &= x_2 &= 0
\end{align*}
\]

symmetric protocol for $P_2$
Program graphs for the Bakery algorithm

\[
\begin{align*}
\text{wait}_1 & \quad \xrightarrow{\text{noncrit}_1} \quad \text{crit}_1 \\
\text{wait}_2 & \quad \xrightarrow{\text{noncrit}_2} \quad \text{crit}_2
\end{align*}
\]

\[
\begin{align*}
x_1 & := x_2 + 1 \\
(x_1 < x_2) \lor (x_2 = 0)
\end{align*}
\]

\[
\begin{align*}
x_1 & := 0
\end{align*}
\]

\[
\begin{align*}
x_2 & := x_1 + 1 \\
(x_2 < x_1) \lor (x_1 = 0)
\end{align*}
\]

\[
\begin{align*}
x_2 & := 0
\end{align*}
\]
Transition system for the Bakery algorithm

\[ \text{wait}_1 :\ x_1 := x_2 + 1 \quad \text{crit}_1 \]
\[ (x_1 < x_2) \lor (x_2 = 0) \]

\[ \text{wait}_2 :\ x_2 := x_1 + 1 \quad \text{crit}_2 \]
\[ (x_2 < x_1) \lor (x_1 = 0) \]
Transition system for the Bakery algorithm

\begin{align*}
\text{wait}_1 & \rightarrow \text{crit}_1 \quad (x_1 < x_2) \lor (x_2 = 0) \\
\text{wait}_2 & \rightarrow \text{crit}_2 \quad (x_2 < x_1) \lor (x_1 = 0) \\
\end{align*}

\text{noncrit}_1

\begin{align*}
x_1 & := x_2 + 1 \\
x_1 & := 0
\end{align*}

\text{noncrit}_2

\begin{align*}
x_2 & := x_1 + 1 \\
x_2 & := 0
\end{align*}
Transition system for the Bakery algorithm

wait$_1$  \[ \text{noncrit}_1 \]
\[ x_1 := x_2 + 1 \]
\( x_1 := 0 \)
\( (x_1 < x_2) \lor (x_2 = 0) \)

crit$_1$

wait$_2$
\[ \text{noncrit}_2 \]
\[ x_2 := x_1 + 1 \]
\( x_2 := 0 \)
\( (x_2 < x_1) \lor (x_1 = 0) \)

crit$_2$
Transition system for the Bakery algorithm

\[ x_1 := x_2 + 1 \]

\[ x_1 := 0 \]

\[ (x_1 < x_2) \lor (x_2 = 0) \]

\[ x_2 := x_1 + 1 \]

\[ x_2 := 0 \]

\[ (x_2 < x_1) \lor (x_1 = 0) \]
Transition system for the Bakery algorithm

\[ x_1 := x_2 + 1 \]

\[ x_1 := 0 \]

\[ n \]

\[ w \]

\[ c \]

\[ x_1 = 0 \]

\[ x_2 = 0 \]

\[ x_1 = 1 \]

\[ x_2 = 1 \]

\[ x_1 = 2 \]

\[ x_2 = 2 \]

\[ x_1 = 3 \]

\[ x_2 = 2 \]

\[ x_1 = 1 \]

\[ x_2 = 1 \]

\[ x_1 = 2 \]

\[ x_2 = 2 \]

\[ x_1 = 2 \]

\[ x_2 = 0 \]

\[ x_1 = 3 \]

\[ x_2 = 3 \]
Transition system for the Bakery algorithm

\[
\begin{align*}
\text{noncrit}_1 & \quad x_1 := x_2 + 1 \\
\text{wait}_1 & \quad (x_1 < x_2) \vee (x_2 = 0) \\
& \quad x_1 := 0 \\
\text{crit}_1 & \\
\text{noncrit}_2 & \quad x_2 := x_1 + 1 \\
\text{wait}_2 & \quad (x_2 < x_1) \vee (x_1 = 0) \\
& \quad x_2 := 0 \\
\text{crit}_2 &
\end{align*}
\]
Bakery algorithm: bisimulation quotient

\[
x_1 := x_2 + 1
\]

\[
x_1 := 0
\]

\(\text{wait}_1\) \rightarrow \text{noncrit}_1 \rightarrow \text{crit}_1

\[(x_1 < x_2) \lor (x_2 = 0)\]

\[
x_2 := x_1 + 1
\]

\[
x_2 := 0
\]

\(\text{wait}_2\) \rightarrow \text{noncrit}_2 \rightarrow \text{crit}_2

\[(x_2 < x_1) \lor (x_1 = 0)\]

infinite transition system with a finite bisimulation quotient
Bakery algorithm: bisimulation quotient

\[
\begin{align*}
x_1 &:= x_2 + 1 \\
\text{wait}_1 &\rightarrow \text{noncrit}_1 \\
(x_1 < x_2) \lor (x_2 = 0) &\rightarrow \text{crit}_1 \\
\end{align*}
\]

\[
\begin{align*}
x_1 &:= 0 \\
\text{wait}_2 &\rightarrow \text{noncrit}_2 \\
(x_2 < x_1) \lor (x_1 = 0) &\rightarrow \text{crit}_2 \\
\end{align*}
\]

\[
\begin{align*}
x_1 &= 0, x_2 &= 0 \\
\end{align*}
\]

\[
\begin{align*}
x_1 &= 0, x_2 > 0 \\
\end{align*}
\]

\[
\begin{align*}
x_1 &= 0, x_2 = 0 \\
\end{align*}
\]

\[
\begin{align*}
x_1 > x_2 > 0 \\
\end{align*}
\]

\[
\begin{align*}
x_2 > x_1 > 0 \\
\end{align*}
\]

\[
\begin{align*}
x_2 > x_1 > 0 \\
\end{align*}
\]

\[
\begin{align*}
x_2 > x_1 > 0 \\
\end{align*}
\]

\[
\begin{align*}
x_1 > x_2 > 0 \\
\end{align*}
\]

\[
\begin{align*}
x_1 > x_2 > 0 \\
\end{align*}
\]

\[
\begin{align*}
x_1 > x_2 > 0 \\
\end{align*}
\]
Bakery algorithm: bisimulation quotient

\[
\begin{align*}
    \text{wait}_1 & \quad n_1 \quad \text{c}_2 \\
    x_1 & := x_2 + 1 \\
    (x_1 < x_2) \lor (x_2 = 0)
\end{align*}
\]

\[
\begin{align*}
    \text{crit}_1 & \quad n_2 \\
    x_1 & := 0
\end{align*}
\]

\[
\begin{align*}
    \text{wait}_2 & \quad w_1 \quad n_2 \\
    x_2 & := x_1 + 1 \\
    (x_2 < x_1) \lor (x_1 = 0)
\end{align*}
\]

\[
\begin{align*}
    \text{crit}_2 & \quad w_2 \\
    x_2 & := 0
\end{align*}
\]

\[
\begin{align*}
    n_1 & \quad w_2 \\
    x_1 & := 0 \\
    x_2 & > 0
\end{align*}
\]

\[
\begin{align*}
    w_1 & \quad n_2 \\
    x_1 & > 0 \\
    x_2 & := 0
\end{align*}
\]

\[
\begin{align*}
    n_1 & \quad c_2 \\
    x_1 & := 0 \\
    x_2 & > 0
\end{align*}
\]

\[
\begin{align*}
    w_1 & \quad w_2 \\
    x_1 & > x_2 > 0
\end{align*}
\]

\[
\begin{align*}
    w_1 & \quad w_2 \\
    x_2 & > x_1 > 0
\end{align*}
\]

\[
\begin{align*}
    c_1 & \quad w_2 \\
    x_2 & > x_1 > 0
\end{align*}
\]

\[
\begin{align*}
    c_1 & \quad n_2 \\
    x_1 & > 0 \\
    x_2 & := 0
\end{align*}
\]

\[
\begin{align*}
    w_1 & \quad c_2 \\
    x_1 & > x_2 > 0
\end{align*}
\]
Bakery algorithm: bisimulation quotient

noncrit\(_1\)  
\(x_1 := x_2 + 1\)  
wait\(_1\)  
\((x_1 < x_2) \lor (x_2 = 0)\)  
crit\(_1\)

noncrit\(_2\)  
\(x_1 := 0\)  
wait\(_2\)  
\((x_2 < x_1) \lor (x_1 = 0)\)  
crit\(_2\)
Bakery algorithm: bisimulation quotient

\[ x_1 := x_2 + 1 \quad x_1 := 0 \]

\[ \text{wait}_1 \quad (x_1 < x_2) \lor (x_2 = 0) \quad \text{crit}_1 \]

\[ n_1 \quad n_2 \quad x_1 = 0 \quad x_2 = 0 \]

\[ w_1 \quad w_2 \quad x_1 > x_2 > 0 \]

\[ c_1 \quad n_2 \quad x_1 > 0 \quad x_2 = 0 \]

\[ w_1 \quad w_2 \quad x_2 > x_1 > 0 \]

\[ w_1 \quad c_2 \quad x_1 > x_2 > 0 \]

\[ c_1 \quad n_2 \quad x_1 > 0 \quad x_2 = 0 \]
Introduction
Modelling parallel systems
Linear Time Properties
Regular Properties
Linear Temporal Logic (LTL)
Computation-Tree Logic

**Equivalences and Abstraction**

- bisimulation
- CTL, CTL*-equivalence
- computing the bisimulation quotient
- abstraction stutter steps
- simulation relations
Recall: CTL*

### CTL* state formulas

\[ \Phi ::= \text{true} \mid a \mid \Phi_1 \land \Phi_2 \mid \neg \Phi \mid \exists \varphi \]

### CTL* path formulas

\[ \varphi ::= \Phi \mid \varphi_1 \land \varphi_2 \mid \neg \varphi \mid \Box \varphi \mid \varphi_1 \mathbin{U} \varphi_2 \]
Recall: CTL*

**CTL* state formulas**

\[ \Phi ::= \text{true} \mid a \mid \Phi_1 \land \Phi_2 \mid \neg \Phi \mid \exists \varphi \]

**CTL* path formulas**

\[ \varphi ::= \Phi \mid \varphi_1 \land \varphi_2 \mid \neg \varphi \mid \Box \varphi \mid \varphi_1 U \varphi_2 \]

derived operators:

- ◊, □, ... as in LTL
Recall: CTL*

\[ \Phi ::= \text{true} \mid a \mid \Phi_1 \land \Phi_2 \mid \neg \Phi \mid \exists \varphi \]

\[ \varphi ::= \Phi \mid \varphi_1 \land \varphi_2 \mid \neg \varphi \mid \Diamond \varphi \mid \varphi_1 \mathbin{U} \varphi_2 \]

derived operators:

- \( \Diamond, \Box, \ldots \) as in LTL
- universal quantification: \( \forall \varphi \overset{\text{def}}{=} \neg \exists \neg \varphi \)
Recall: CTL* and CTL

**CTL* state formulas**

\[ \Phi ::= \text{true} \mid a \mid \Phi_1 \land \Phi_2 \mid \neg \Phi \mid \exists \varphi \]

**CTL* path formulas**

\[ \varphi ::= \Phi \mid \varphi_1 \land \varphi_2 \mid \neg \varphi \mid \square \varphi \mid \varphi_1 \mathbin{U} \varphi_2 \]

**CTL**: sublogic of **CTL***
Recall: $\text{CTL}^*$ and $\text{CTL}$

### $\text{CTL}^*$ state formulas

$$\Phi ::= \text{true} \mid a \mid \Phi_1 \land \Phi_2 \mid \neg \Phi \mid \exists \varphi$$

### $\text{CTL}^*$ path formulas

$$\varphi ::= \Phi \mid \varphi_1 \land \varphi_2 \mid \neg \varphi \mid \bigcirc \varphi \mid \varphi_1 \mathbf{U} \varphi_2$$

**$\text{CTL}$**: sublogic of $\text{CTL}^*$

- with path quantifiers $\exists$ and $\forall$
Recall: CTL* and CTL

**CTL* state formulas**

\[ \Phi ::= \text{true} | a | \Phi_1 \land \Phi_2 | \neg \Phi | \exists \varphi \]

**CTL* path formulas**

\[ \varphi ::= \Phi | \varphi_1 \land \varphi_2 | \neg \varphi | \mathcal{O} \varphi | \varphi_1 \mathcal{U} \varphi_2 \]

**CTL**: sublogic of CTL*

- with path quantifiers \( \exists \) and \( \forall \)
- restricted syntax of path formulas:
Recall: CTL* and CTL

**CTL* state formulas**

\[ \Phi ::= \text{true} \mid a \mid \Phi_1 \land \Phi_2 \mid \neg \Phi \mid \exists \varphi \]

**CTL* path formulas**

\[ \varphi ::= \Phi \mid \varphi_1 \land \varphi_2 \mid \neg \varphi \mid \bigcirc \varphi \mid \varphi_1 \mathbin{U} \varphi_2 \]

**CTL**: sublogic of **CTL***

- with path quantifiers \( \exists \) and \( \forall \)
- restricted syntax of path formulas:
  - * no boolean combinations of path formulas
  - * arguments of temporal operators \( \bigcirc \) and \( \mathbin{U} \) are state formulas
CTL equivalence
Let $s_1, s_2$ be states of a TS $T$ without terminal states.
Let $s_1, s_2$ be states of a TS $T$ without terminal states. $s_1, s_2$ are CTL equivalent if for all CTL formulas $\Phi$:

$$s_1 \models \Phi \iff s_2 \models \Phi$$
Let $s_1, s_2$ be states of a TS $\mathcal{T}$ without terminal states.

$s_1, s_2$ are CTL equivalent if for all CTL formulas $\Phi$:

$s_1 \models \Phi$ iff $s_2 \models \Phi$
Let $s_1, s_2$ be states of a TS $T$ without terminal states.

$s_1, s_2$ are **CTL** equivalent if for all **CTL** formulas $\Phi$:

$$s_1 \models \Phi \iff s_2 \models \Phi$$

$s_1, s_2$ are not **CTL** equivalent

$$s_1 \models \exists O (\exists O a \land \exists O b)$$

$$s_2 \not\models \exists O (\exists O a \land \exists O b)$$
Let $s_1, s_2$ be states of a TS $T$ without terminal states

$s_1, s_2$ are $\text{CTL}$ equivalent if for all $\text{CTL}$ formulas $\Phi$:

$$s_1 \models \Phi \iff s_2 \models \Phi$$

analogous definition for $\text{CTL}^*$ and $\text{LTL}$
Let $s_1, s_2$ be states of a TS $\mathcal{T}$ without terminal states.

$s_1, s_2$ are **CTL** equivalent if for all **CTL** formulas $\Phi$:

$s_1 \models \Phi$ iff $s_2 \models \Phi$

$s_1, s_2$ are **CTL*** equivalent if for all **CTL*** formulas $\Phi$:

$s_1 \models \Phi$ iff $s_2 \models \Phi$

$s_1, s_2$ are **LTL** equivalent if for all **LTL** formulas $\varphi$:

$s_1 \models \varphi$ iff $s_2 \models \varphi$
CTL/CTL* and bisimulation
CTL/CTL* and bisimulation

bisimulation equivalence

= \textcolor{blue}{CTL} equivalence

= \textcolor{blue}{CTL^*} equivalence
CTL/CTL* and bisimulation

\[
\text{bisimulation equivalence} = \text{CTL equivalence} = \text{CTL* equivalence}
\]

←−←−←− for finite TS
CTL/CTL* and bisimulation

Let $T$ be a finite TS without terminal states, and $s_1, s_2$ states in $T$. Then:

$$s_1 \sim_T s_2$$

iff $s_1$ and $s_2$ are $\text{CTL}$ equivalent

iff $s_1$ and $s_2$ are $\text{CTL}^*$ equivalent

bisimulation equivalence

$= \text{CTL equivalence}$

$= \text{CTL}^*\text{ equivalence}$

← for finite TS
CTL/CTL* and bisimulation

\[ \sim \]

CTL equivalence

CTL* equivalence

CTL/CTL* and bisimulation

CTL equivalence

CTL* equivalence
CTL/CTL* and bisimulation

CTL is a sublogic of CTL*

CTL equivalence

CTL* equivalence

bisimulation equivalence ~
**CTL/CTL* and bisimulation**

For TS that are finitely branching:

-CTL equivalence
-CTL* equivalence
-Bisimulation equivalence $\sim$

CTL is a sublogic of CTL*
CTL/CTL* and bisimulation

 CTL equivalence

 bisimulation equivalence ~

 for TS that are finitely branching

 for arbitrary TS

 CTL equivalence

 CTL* equivalence

 CTL is a sublogic of CTL*
If $\mathcal{T}_1$, $\mathcal{T}_2$ are finitely branching TS over $\mathcal{AP}$ without terminal states then:

$$\mathcal{T}_1 \sim \mathcal{T}_2$$

iff $\mathcal{T}_1$ and $\mathcal{T}_2$ satisfy the same $\mathbf{CTL}$ formulas

iff $\mathcal{T}_1$ and $\mathcal{T}_2$ satisfy the same $\mathbf{CTL}^*$ formulas
**Correct or wrong?**

\[
\text{CTL equivalence is finer than LTL equivalence}
\]
Correct or wrong?

CTL equivalence is finer than LTL equivalence

correct.
Correct or wrong?

**CTL** equivalence is finer than **LTL** equivalence

Correct.

**CTL** equivalence $= \textbf{CTL}^*$ equivalence

**LTL** is sublogic of **CTL***
Correct or wrong?

**CTL** equivalence is finer than **LTL** equivalence

Correct.

**LTL** equivalence is finer than **CTL** equivalence
Correct or wrong?

**CTL** equivalence is finer than **LTL** equivalence

**correct.**

**LTL** equivalence is finer than **CTL** equivalence

**wrong.**
Correct or wrong?

**CTL** equivalence is finer than **LTL** equivalence

correct.

**LTL** equivalence is finer than **CTL** equivalence

wrong.
Correct or wrong?

**IFT** equivalence is finer than **LTL** equivalence

**Correct.**

**LTL** equivalence is finer than **IFT** equivalence

**Wrong.**

\[ s_1, s_2 \text{ are trace equivalent} \]
Correct or wrong?

**CTL** equivalence is finer than **LTL** equivalence

correct.

**LTL** equivalence is finer than **CTL** equivalence

wrong.

$s_1, s_2$ are trace equivalent and **LTL** equivalent
Correct or wrong?

**CTL** equivalence is finer than **LTL** equivalence

**Correct.**

**LTL** equivalence is finer than **CTL** equivalence

**Wrong.**

\[ s_1 \models \exists \diamond \exists \diamond (\exists \diamond a \land \exists \diamond b) \]

\[ s_2 \not\models \exists \diamond \exists \diamond (\exists \diamond a \land \exists \diamond b) \]

\[ s_1, s_2 \] are trace equivalent and **LTL** equivalent.
Summary: equivalences

- bisimulation equivalence
- LTL equivalence
- CTL equivalence
- CTL* equivalence

for finitely branching TS
Summary: equivalences

- Trace equivalence
- LTL equivalence
- Bisimulation equivalence
- CTL equivalence
- CTL* equivalence

for finitely branching TS
Summary: equivalences

finite trace equivalence

trace equivalence

bisimulation equivalence

LTL equivalence

CTL equivalence

CTL* equivalence

for finitely branching TS
Summary: equivalences

finite trace equivalence

trace equivalence

bisimulation equivalence

CTLEQ5.2-10

equivalence w.r.t.
LTL safety properties

LTL equivalence

CTL equivalence

CTL* equivalence

for finitely branching TS