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

$\mathcal{T}_1$: 

$\mathcal{T}_2$: 

$\hat{=} = \emptyset$

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

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

$\mathcal{T}_1$:  

$\mathcal{T}_2$:  

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

- $\emptyset \equiv \emptyset$
- $\{a\} \equiv \{a\}$
- $\{b\} \equiv \{b\}$
Trace equivalence

\[ \mathcal{T}_1: \]

\[ \mathcal{T}_2: \]

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

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

\[ \mathcal{T}_1: \]

\[ \mathcal{T}_2: \]

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

CTL-formula \( \Phi = \exists \bigcirc (\exists \bigcirc a \land \exists \bigcirc b) \)

\( \mathcal{T}_1 \not\models \Phi \) and \( \mathcal{T}_2 \models \Phi \)
Trace equivalence is not compatible with CTL

\[ \mathcal{T}_1 : \]

\[ \mathcal{T}_2 : \]

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

CTL-formula \( \Phi \) = \( \exists \bigcirc (\exists \bigcirc a \land \exists \bigcirc b) \)

\[ \mathcal{T}_1 \not\models \Phi \text{ and } \mathcal{T}_2 \models \Phi \]
Implementation relations

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

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

- for the **analysis** of complex systems
Implementation relations

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

- for the **analysis** of complex systems
  ⟷ 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
  → comparison of 2 transition systems

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

**use equivalence relation** ~ for the states of a single transition system $T$ and analyze the quotient $T/\sim$
Implementation relations

• for the design of complex systems
  \[\sim\] comparison of 2 transition systems

• for the analysis of complex systems
  \[\sim\] homogeneous model checking approach
  \[\sim\] graph minimization

use equivalence relation \(\sim\) for the states of a single transition system \(T\) and analyze the quotient \(T/\sim\)

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

\[T \models \Phi \iff T/\sim \models \Phi\]

for all “relevant” properties \(\Phi\)
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., $\text{Tracesfin}(\mathcal{T}_1) \subseteq \text{Tracesfin}(\mathcal{T}_2)$

preserves all linear-time safety properties

trace inclusion and trace equivalence:

e.g., $\text{Traces}(\mathcal{T}_1) \subseteq \text{Traces}(\mathcal{T}_2)$
Linear-time implementation relations

finite trace inclusion and equivalence:
    e.g., $\text{Traces}_{\text{fin}}(\mathcal{T}_1) \subseteq \text{Traces}_{\text{fin}}(\mathcal{T}_2)$

    preserves all linear-time safety properties

trace inclusion and trace equivalence:
    e.g., $\text{Traces}(\mathcal{T}_1) \subseteq \text{Traces}(\mathcal{T}_2)$

    preserves all LTL properties
Linear-time implementation relations

finite trace inclusion and equivalence:
  e.g., $\text{Traces}_{\text{fin}}(\mathcal{T}_1) \subseteq \text{Traces}_{\text{fin}}(\mathcal{T}_2)$
  preserves all linear-time safety properties

trace inclusion and trace equivalence:
  e.g., $\text{Traces}(\mathcal{T}_1) \subseteq \text{Traces}(\mathcal{T}_2)$
  preserves all LTL properties

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

finite trace inclusion and equivalence:

\[ \text{Traces}_{\text{fin}}(T_1) \subseteq \text{Traces}_{\text{fin}}(T_2) \]

preserves all linear-time safety properties

trace inclusion and trace equivalence:

\[ \text{Traces}(T_1) \subseteq \text{Traces}(T_2) \]

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., \( \text{Traces}^{\text{fin}}(I_1) \subseteq \text{Traces}^{\text{fin}}(I_2) \)
  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

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

$\mathcal{T}_1$:

$\mathcal{T}_2$:
Minimization w.r.t. trace equivalence?

$\mathcal{T}_1$:  

$\mathcal{T}_2$:  

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

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

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

\( \mathcal{T}_1: \)

\begin{itemize}
  \item \( \text{Traces}(\mathcal{T}_1) = \text{Traces}(\mathcal{T}_2) \)
  \item but \( \mathcal{T}_1 \) and \( \mathcal{T}_2 \) are not isomorphic
  \item \( \mathcal{T}_1, \mathcal{T}_2 \) have 5 states and 7 transitions each
  \item there is no smaller TS that is trace-equivalent to \( \mathcal{T}_i \)
\end{itemize}
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

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

- with the same set $AP$
- possibly containing terminal states
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

- with the same set \( AP \)
- possibly containing terminal states

Bisimulation equivalence of \( \mathcal{T}_1 \) and \( \mathcal{T}_2 \) requires that \( \mathcal{T}_1 \) and \( \mathcal{T}_2 \) 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 \((T_1, T_2)\)
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 \((T_1, T_2)\)

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 \((T_1, T_2)\)

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. \((s'_1, s'_2) \in R\)
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. \((s'_1, s'_2) \in \mathcal{R}\)

\[
\begin{array}{c}
\text{s}_1 \xrightarrow{\mathcal{R}} \text{s}_2 \\
\downarrow \\
\text{s}'_1
\end{array}
\quad \text{can be completed to}
\quad \begin{array}{c}
\text{s}_1 \xrightarrow{\mathcal{R}} \text{s}_2 \\
\downarrow \\
\text{s}'_1 \xrightarrow{\mathcal{R}} \text{s}'_2
\end{array}
\]
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 \((\mathcal{T}_1, \mathcal{T}_2)\)

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. \((s_1', s_2') \in \mathcal{R}\)

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

can be completed to

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

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

and such that the following initial condition holds:

1. \(\forall s_{0,1} \in S_{0,1} \exists s_{0,2} \in S_{0,2}\) 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. \((s'_1, s'_2) \in \mathcal{R}\)

\[
\begin{array}{|c|c|}
\hline
s_1 & s_2 \\
\hline
s'_1 & \mathcal{R} \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|}
\hline
s_1 & s_2 \\
\hline
s'_1 & \mathcal{R} \\
\hline
\end{array}
\]

can be completed to

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

\[
\begin{array}{|c|c|}
\hline
s_1 & s_2 \\
\hline
s'_1 & \mathcal{R} \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|}
\hline
s_1 & s_2 \\
\hline
s'_1 & \mathcal{R} \\
\hline
\end{array}
\]

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

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

\begin{align*}
(1) & \text{ labeling condition} \\
(2) & \text{ mutual stepwise simulation} \\
(3) & \text{ initial condition (I)}
\end{align*}

\textbf{bisimulation equivalence \( \sim \) for TS:}
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

and initial condition (I)

bisimulation equivalence $\sim$ for TS:

$\mathcal{T}_1 \sim \mathcal{T}_2$ iff there is a bisimulation $\mathcal{R}$ for $(\mathcal{T}_1, \mathcal{T}_2)$
Bisimulation equivalence $\sim$

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
3. initial condition

and initial condition (I)

bisimulation equivalence $\sim$ for TS:

$T_1 \sim T_2$ iff there is a bisimulation $R$ for $(T_1, T_2)$

for state $s_1$ of $T_1$ and state $s_2$ of $T_2$:

$s_1 \sim s_2$ iff there exists a bisimulation $R$ for $(T_1, T_2)$ such that $(s_1, s_2) \in R$
Two beverage machines

$T_1$

$T_2$

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

$\mathcal{T}_1$

- pay
- select
- coke
- soda

$\mathcal{T}_2$

- pay
- select
- coke\textsubscript{1}
- coke\textsubscript{2}
- soda

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

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

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

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

$T_1$

$\text{pay}$

$\text{paid}_1$

$\text{paid}_2$

$\text{coke}$

$\text{soda}$

$T_2$

$\text{pay}$

$\text{select}$

$\text{coke}$

$\text{soda}$

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

\[ \mathcal{T}_1 \] and \[ \mathcal{T}_2 \]

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

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

- a successor labeled with \textit{coke} and
- a successor labeled with \textit{soda}

\[ AP = \{ \text{pay, coke, soda} \} \]
Simulation condition of bisimulations

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

can be completed to

\[
\begin{array}{c|c|c}
  s_1 & \mathcal{R} & s_2 \\
  \downarrow & \downarrow & \downarrow \\
  s'_1 & \mathcal{R} & s'_2 \\
\end{array}
\]
Path lifting for bisimulation $\mathcal{R}$
Path lifting for bisimulation $\mathcal{R}$

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

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

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

can be completed to

\[
\begin{array}{c}
\vdots \\
S_{1,4} \\
\vdots \\
S_{1,3} \\
\vdots \\
S_{1,2} \\
\vdots \\
S_{1,1} \\
\vdots \\
S_1
\end{array}
\quad \mathcal{R} \quad
\begin{array}{c}
\vdots \\
S_{1,4} \\
\vdots \\
S_{1,3} \\
\vdots \\
S_{1,2} \\
\vdots \\
S_{1,1} \\
\vdots \\
S_1
\end{array}
\quad \quad
\begin{array}{c}
\vdots \\
S_{2,4} \\
\vdots \\
S_{2,3} \\
\vdots \\
S_{2,2} \\
\vdots \\
S_{2,1} \\
\vdots \\
S_2
\end{array}
\quad \mathcal{R} \quad
\begin{array}{c}
\vdots \\
S_{2,4} \\
\vdots \\
S_{2,3} \\
\vdots \\
S_{2,2} \\
\vdots \\
S_{2,1} \\
\vdots \\
S_2
\end{array}
\]
Path lifting for bisimulation $\mathcal{R}$

\[
\begin{array}{c}
S_1 \\
\downarrow \\
S_{1,1} \\
\downarrow \\
S_{1,2} \\
\downarrow \\
S_{1,3} \\
\downarrow \\
S_{1,4} \\
\downarrow \\
\vdots
\end{array}
\quad
\begin{array}{c}
S_2 \\
\downarrow \\
S_{2,1} \\
\downarrow \\
S_{2,2} \\
\downarrow \\
S_{2,3} \\
\downarrow \\
S_{2,4} \\
\downarrow \\
\vdots
\end{array}
\]

$\mathcal{R}$

\[
\begin{array}{c}
\begin{array}{c}
S_1 \\
\downarrow \\
S_{1,1} \\
\downarrow \\
S_{1,2} \\
\downarrow \\
S_{1,3} \\
\downarrow \\
S_{1,4} \\
\downarrow \\
\vdots
\end{array}
\quad
\begin{array}{c}
\begin{array}{c}
S_2 \\
\downarrow \\
S_{2,1} \\
\downarrow \\
S_{2,2} \\
\downarrow \\
S_{2,3} \\
\downarrow \\
S_{2,4} \\
\downarrow \\
\vdots
\end{array}
\end{array}
\end{array}
\]

can be completed to
Properties of bisimulation equivalence
Properties of bisimulation equivalence

\[ \sim \text{ is an equivalence} \]
Properties of bisimulation equivalence

\[ \sim \] is an equivalence, i.e.,

- reflexivity: \( \mathcal{T} \sim \mathcal{T} \) for all transition systems \( \mathcal{T} \).
Properties of bisimulation equivalence

\[ \sim \] is an equivalence, i.e.,

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

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

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

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

∼ is an equivalence, i.e.,

- reflexivity: \( T \sim T \) for all transition systems \( T \)
- symmetry: \( T_1 \sim T_2 \) implies \( T_2 \sim T_1 \)

If \( R \) is a bisimulation for \((T_1, T_2)\) then

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

is a bisimulation for \((T_2, T_1)\)
Properties of bisimulation equivalence

\[ \sim \] 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 \( \mathcal{T}_2 \sim \mathcal{T}_1 \)
- transitivity: if \( \mathcal{T}_1 \sim \mathcal{T}_2 \) and \( \mathcal{T}_2 \sim \mathcal{T}_3 \) then \( \mathcal{T}_1 \sim \mathcal{T}_3 \)
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?
Correct or wrong?

\[ \sim \]

\textit{Wrong}
Correct or wrong?

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

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

\[ s_1 \rightarrow u, \text{ but } s_2 \not\rightarrow \text{blue} \]

(Thus \( s_1 \not\sim s_2 \))

---

\[ s_1 \sim s_2 \]

---

Correct
Correct or wrong?

\[ s_1 \not\sim s_2 \]

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

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

Correct
Correct or wrong?

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

\[ \{(s_1, s_2), (s_1', s_2'), (s_1', s_2''), (t_1, t_2), (t_1', t_2), (t_1'', t_2)\} \]
Correct or wrong?

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

\[
\text{correct}
\]
Correct or wrong?

Correct bisimulation: \[ \{(s_1, s_2), (t_1, t_2), (t'_1, t_2), (u_1, u_2), (v_1, v_2)\} \]
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 ...

\[ \text{Traces}(T_1) = \text{Traces}(T_2) \iff T_1 \sim T_2 \]
Bisimulation vs. trace equivalence

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

proof: ... path fragment lifting ...

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

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 ...

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

Trace equivalence is \textit{strictly coarser} than bisimulation equivalence.
Bisimulation vs. trace equivalence

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

Proof: ... path fragment lifting ...

$$\text{Traces}(T_1) = \text{Traces}(T_2) \not\implies T_1 \sim T_2$$

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

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

- 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

\[ \mathcal{T}_1 \]

\[ \mathcal{T}_2 \]

• as a relation on the states of 1 transition system

\[ \mathcal{T} \]

\[ s_1 \sim s_2 \text{ iff } \mathcal{T}_{s_1} \sim \mathcal{T}_{s_2} \]
Bisimulation equivalence ...

- as a relation that compares 2 transition systems

- 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

\[ \mathcal{T}_1 \]

\[ \mathcal{T}_2 \]

• as a relation on the states of 1 transition system

\[ \mathcal{T} \]

\[ \mathcal{T}_{s_1} \]

\[ \mathcal{T}_{s_2} \]

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

Let $\mathcal{T}$ be a TS with proposition set $AP$.

A bisimulation for $\mathcal{T}$ is a binary relation $R$ on the state space of $\mathcal{T}$ 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. $(s_1', s_2') \in R$
Bisimulation equivalence $\sim_{\mathcal{T}}$ on a single TS

Let $\mathcal{T}$ be a TS with proposition set $AP$.

A bisimulation for $\mathcal{T}$ is a binary relation $R$ on the state space of $\mathcal{T}$ 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. $(s_1', s_2') \in R$

bisimulation equivalence $\sim_{\mathcal{T}}$:

$s_1 \sim_{\mathcal{T}} s_2$ iff there exists a bisimulation $R$ for $\mathcal{T}$ s.t. $(s_1, s_2) \in R$
Let $\mathcal{T}$ be a transition system with state space $S$.

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

- the coarest bisimulation on $\mathcal{T}$
- and an equivalence on $S$
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 $s_2$:

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

can be completed to

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

\(\sim\) 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 \(\sim\)

\[
\text{for all states } s_1 \text{ and } s_2 \text{ of } T:
\]

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

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 \( \sim \)

\[
\text{for all states } s_1 \text{ and } s_2 \text{ of } T:
\]

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

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 $\mathcal{T} = \mathcal{T}_1 \cup \mathcal{T}_2$
(state space $S_1 \cup S_2$)
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 $S_1 \uplus S_2$)

$\mathcal{T}_1 \sim \mathcal{T}_2$ 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 $T_1$ and $T_2$

$T_1$ with state space $S_1$

$T_2$ with state space $S_2$

consider $T = T_1 \uplus T_2$
(state space $S_1 \uplus S_2$)

$T_1 \sim T_2$ iff $\forall$ initial states $s_1$ of $T_1$

$\exists$ initial state $s_2$ of $T_2$ s.t. $s_1 \sim_T s_2$
and vice versa
Bisimulation quotient

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

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

bisimulation quotient:

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

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:
  \[ S' = S / \sim_\mathcal{T} \]

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: $S'_0 = \{ [s]_{\sim_T} : s \in S_0 \}$
Bisimulation quotient

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

**Bisimulation quotient:**

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

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

Well-defined by the labeling condition of bisimulations.
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: $S'_0 = \{ [s]_{\sim_T} : s \in S_0 \}$
- labeling function: $L'([s]_{\sim_T}) = L(s)$
- transition relation:

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

bisimulation quotient:

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

- 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:

$$s \xrightarrow{\alpha} s' \quad \quad \quad [s]_{\sim_T} \xrightarrow{\tau} [s']_{\sim_T}$$

action labels irrelevant
Let $\mathcal{T} = (S, Act, \rightarrow, S_0, AP, L)$ 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 $\mathcal{T} = \underbrace{Printer || Printer || \ldots || Printer}_{n \text{ printer}}$
Example: interleaving of $n$ printers

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

$n$ printer

transition system for each printer
Example: interleaving of $n$ printers

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

$\mathcal{AP} = \{0, 1, \ldots, n\}$  
“number of available printers”

transition system for each printer
Example: $n=3$ printers

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

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

$p$: is printing

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

Parallel system $\mathcal{T} = \underbrace{\text{Printer} \ || \ \text{Printer} \ || \ \ldots \ || \ \text{Printer}}_{n \ \text{printer}}$

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

$p$: is printing

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

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

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

$p$: is printing
$r$: ready to print

bisimulation quotient
Example: $n=3$ printers

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

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

$2^n$ states

$n+1$ states
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: $x_1, x_2 \in \mathbb{N}$
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 $P_1$:

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

symmetric protocol for $P_2$
Bakery algorithm

protocol for $P_1$:

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

initially:

$x_1 = x_2 = 0$

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

\( x_1 := x_2 + 1 \)
\( (x_1 < x_2) \lor (x_2 = 0) \)

\( x_1 := 0 \)

\( x_2 := x_1 + 1 \)
\( (x_2 < x_1) \lor (x_1 = 0) \)

\( x_2 := 0 \)
Transition system for the Bakery algorithm

\[ x_1 := x_2 + 1 \quad \text{noncrit}_1 \]

\[ x_1 := 0 \quad \text{crit}_1 \]

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

\[ x_2 := x_1 + 1 \quad \text{noncrit}_2 \]

\[ x_2 := 0 \quad \text{crit}_2 \]

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

\[ x_1 := x_2 + 1 \quad \text{noncrit}_1 \]

\[ x_1 := 0 \quad \text{crit}_1 \]

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

\[ x_2 := x_1 + 1 \quad \text{noncrit}_2 \]

\[ x_2 := 0 \quad \text{crit}_2 \]

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

\[ x_1 := x_2 + 1 \quad \text{noncrit}_1 \]

\[ x_1 := 0 \quad \text{crit}_1 \]

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

\[ x_2 := x_1 + 1 \quad \text{noncrit}_2 \]

\[ x_2 := 0 \quad \text{crit}_2 \]

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

\[ x_1 := x_2 + 1 \quad \text{noncrit}_1 \]

\[ x_1 := 0 \quad \text{wait}_1 \]

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

\[ x_2 := x_1 + 1 \quad \text{noncrit}_2 \]

\[ x_2 := 0 \quad \text{wait}_2 \]

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

\[ x_1 := x_2 + 1 \quad \text{noncrit}_1 \]

\[ x_1 := 0 \quad \text{crit}_1 \]

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

\[ x_2 := x_1 + 1 \quad \text{noncrit}_2 \]

\[ x_2 := 0 \quad \text{crit}_2 \]

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

**noncrit₁**

\[ x₁ := x₂ + 1 \]

**wait₁**

\[ (x₁ < x₂) \lor (x₂ = 0) \]

**crit₁**

\[ x₁ := 0 \]

**noncrit₂**

\[ x₂ := x₁ + 1 \]

**wait₂**

\[ (x₂ < x₁) \lor (x₁ = 0) \]

**crit₂**

\[ x₂ := 0 \]
Bakery algorithm: bisimulation quotient

\[ x_1 := x_2 + 1 \quad \text{wait}_1 \]

\[ x_1 := 0 \quad \text{crit}_1 \]

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

\[ x_2 := x_1 + 1 \quad \text{noncrit}_1 \]

\[ x_2 := 0 \quad \text{crit}_2 \]

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

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

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

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

\[ x_1 := x_2 + 1 \]

\[ \text{crit}_1 \]

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

\[ x_1 := 0 \]

\[ \text{wait}_1 \]

\[ x_2 := x_1 + 1 \]

\[ \text{crit}_2 \]

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

\[ x_2 := 0 \]

\[ \text{wait}_2 \]
Bakery algorithm: bisimulation quotient

\[ x_1 := x_2 + 1 \]

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

\[ x_1 := 0 \]

\[ \text{crit}_1 \]

\[ x_1 := x_2 + 1 \]

\[ \text{wait}_2 \quad \forall (x_2 < x_1) \lor (x_1 = 0) \]

\[ x_2 := 0 \]

\[ \text{crit}_2 \]
Bakery algorithm: bisimulation quotient

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

- \(\Diamond, \Box, \ldots\) as in LTL
- universal quantification: \(\forall \varphi \equiv \neg \exists \neg \varphi\)
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

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

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 \( \Box \) and \( \mathbf{U} \) are state formulas
CTL equivalence
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 $\Phi$:

$$s_1 \models \Phi \text{ iff } s_2 \models \Phi$$
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

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

$$s_2 \not\models \exists \diamond (\exists \diamond a \land \exists \diamond b)$$
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 $\Phi$: 

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

analogous definition for **CTL* and **LTL**
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 $\Phi$:

$$s_1 \models \Phi \text{ iff } s_2 \models \Phi$$

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

$$s_1 \models \Phi \text{ iff } s_2 \models \Phi$$

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

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

\[
\text{bisimulation equivalence} \quad = \quad \text{CTL equivalence} \quad = \quad \text{CTL}^* \text{ equivalence}
\]
CTL/CTL* and bisimulation

bisimulation equivalence

=  **CTL** equivalence

=  **CTL**\(^*\) equivalence

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

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

$$s_1 \sim_\mathcal{T} s_2$$

iff $s_1$ and $s_2$ are CTL equivalent

iff $s_1$ and $s_2$ are CTL* equivalent
CTL/CTL* and bisimulation

- CTL equivalence
- CTL* equivalence
- Bisimulation equivalence

CTL/CTL* and bisimulation

CTL equivalence

CTL* equivalence

Bisimulation equivalence
CTL/CTL* and bisimulation

CTL is a sublogic of CTL*

CTL equivalence

CTL* equivalence

bisimulation equivalence ~(approximately equal to)
CTL/CTL* and bisimulation

for TS that are finitely branching

CTL equivalence

CTL is a sublogic of CTL*

CTL* equivalence

bisimulation equivalence ~
CTL/CTL* and bisimulation

CTL equivalence \sim \CTL* equivalence for arbitrary TS

CTL equivalence for TS that are finitely branching

CTL is a sublogic of CTL*
Bisimulation equivalence $\Rightarrow$ CTL* equivalence

For arbitrary (possibly infinite) transition systems without terminal states:

If $s_1, s_2$ are states with $s_1 \sim_T s_2$ then for all CTL* formulas $\Phi$:

$$s_1 \models \Phi \iff s_2 \models \Phi$$
Bisimulation equivalence ⇒ CTL* equivalence

show by structural induction on CTL* formulas:

(a) if \( s_1, s_2 \) are states with \( s_1 \sim_T s_2 \) then
    for all CTL* state formulas \( \Phi \):
    \[
    s_1 \models \Phi \iff s_2 \models \Phi
    \]

(b) if \( \pi_1, \pi_2 \) are paths with \( \pi_1 \sim_T \pi_2 \) then
    for all CTL* path formulas \( \varphi \):
    \[
    \pi_1 \models \varphi \iff \pi_2 \models \varphi
    \]
show by structural induction on $\text{CTL}^*$ formulas:

(a) if $s_1, s_2$ are states with $s_1 \sim_T s_2$ then for all $\text{CTL}^*$ state formulas $\Phi$:

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

(b) if $\pi_1, \pi_2$ are paths with $\pi_1 \sim_T \pi_2$ then for all $\text{CTL}^*$ path formulas $\varphi$:

$$ \pi_1 \models \varphi \iff \pi_2 \models \varphi $$

$$ \pi_1 \sim_T \pi_2 \iff \text{def} \quad \pi_1 \text{ and } \pi_2 \text{ are statewise bisimulation equivalent} $$
Bisimulation equivalence $\Rightarrow$ CTL* equivalence

statewise bisimulation equivalent paths:

$s_1$ $\sim_T$ $s_2$

$\downarrow$ $\downarrow$

$s_{11}$ $\sim_T$ $s_{12}$

$\downarrow$ $\downarrow$

$s_{21}$ $\sim_T$ $s_{22}$

$\downarrow$ $\downarrow$

$s_{31}$ $\sim_T$ $s_{32}$

$\downarrow$ $\downarrow$

$\ldots$ $\ldots$

path $\pi_1$ path $\pi_2$
Bisimulation equivalence $\Rightarrow$ CTL* equivalence

For all CTL* state formulas $\Phi$ and path formulas $\varphi$:

(a) if $s_1 \sim_T s_2$ then: $s_1 \models \Phi$ iff $s_2 \models \Phi$

(b) if $\pi_1 \sim_T \pi_2$ then: $\pi_1 \models \varphi$ iff $\pi_2 \models \varphi$
Bisimulation equivalence $\Rightarrow$ CTL* equivalence

For all CTL* state formulas $\Phi$ and path formulas $\varphi$:

(a) if $s_1 \sim_T s_2$ then: $s_1 \models \Phi$ iff $s_2 \models \Phi$

(b) if $\pi_1 \sim_T \pi_2$ then: $\pi_1 \models \varphi$ iff $\pi_2 \models \varphi$

Proof by structural induction
Bisimulation equivalence \(\Rightarrow\) CTL* equivalence

For all CTL* state formulas \(\Phi\) and path formulas \(\varphi\):

(a) if \(s_1 \sim_T s_2\) then: \(s_1 \models \Phi\) iff \(s_2 \models \Phi\)

(b) if \(\pi_1 \sim_T \pi_2\) then: \(\pi_1 \models \varphi\) iff \(\pi_2 \models \varphi\)

Proof by structural induction

base of induction:

(a) \(\Phi = \text{true}\) or \(\Phi = a \in AP\)

(b) \(\varphi = \Phi\) for some state formula \(\Phi\) s.t. statement (a) holds for \(\Phi\)
Bisimulation equivalence $\Rightarrow$ CTL* equivalence

For all CTL* state formulas $\Phi$ and path formulas $\varphi$:

(a) if $s_1 \simT s_2$ then: $s_1 \models \Phi$ iff $s_2 \models \Phi$

(b) if $\pi_1 \simT \pi_2$ then: $\pi_1 \models \varphi$ iff $\pi_2 \models \varphi$

Proof by structural induction

step of induction:

(a) consider $\Phi = \Phi_1 \land \Phi_2, \neg \Psi$ or $\exists \varphi$ s.t.

   (a) holds for $\Phi_1, \Phi_2, \Psi$

   (b) holds for $\varphi$

(b) consider $\varphi = \varphi_1 \land \varphi_2, \neg \varphi', \bigcirc \varphi', \varphi_1 \cup \varphi_2$ s.t.

   (a) holds for $\varphi_1, \varphi_2, \varphi'$
Path lifting for $\sim_T$

can be completed to
Path lifting for $\sim_T$

If $s_1 \sim_T s_2$ then for all $\pi_1 \in \text{Paths}(s_1)$ there exists $\pi_2 \in \text{Paths}(s_2)$ with $\pi_1 \sim_T \pi_2$
Path lifting for $\sim_T$

If $s_1 \sim_T s_2$ then for all $\pi_1 \in Paths(s_1)$ there exists $\pi_2 \in Paths(s_2)$ with $\pi_1 \sim_T \pi_2$
Path lifting for $\sim_T$

If $s_1 \sim_T s_2$ then for all $\pi_1 \in Paths(s_1)$ there exists $\pi_2 \in Paths(s_2)$ with $\pi_1 \sim_T \pi_2$
Correct or wrong?

If $s_1, s_2$ are not $\text{CTL}$ equivalent then there exists a $\text{CTL}$ formula $\Phi$ with $s_1 \models \Phi$ and $s_2 \not\models \Phi$
Correct or wrong?

If \( s_1, s_2 \) are not \textbf{CTL} equivalent then there exists a \textbf{CTL} formula \( \Phi \) with \( s_1 \models \Phi \) and \( s_2 \not\models \Phi \)

correct.
Correct or wrong?

If $s_1, s_2$ are not **CTL** equivalent then there exists a **CTL** formula $\Phi$ with $s_1 \models \Phi$ and $s_2 \not\models \Phi$

correct.

If $s_1, s_2$ are not **LTL** equivalent then there exists a **LTL** formula $\varphi$ with $s_1 \models \varphi$ and $s_2 \not\models \varphi$
Correct or wrong?

If $s_1, s_2$ are not **CTL** equivalent then there exists a **CTL** formula $\Phi$ with $s_1 \models \Phi$ and $s_2 \not\models \Phi$

correct.

If $s_1, s_2$ are not **LTL** equivalent then there exists a **LTL** formula $\varphi$ with $s_1 \models \varphi$ and $s_2 \not\models \varphi$

wrong.
Correct or wrong?

If $s_1, s_2$ are not **CTL** equivalent then there exists a **CTL** formula $\Phi$ with $s_1 \models \Phi$ and $s_2 \not\models \Phi$

correct.

If $s_1, s_2$ are not **LTL** equivalent then there exists a **LTL** formula $\varphi$ with $s_1 \models \varphi$ and $s_2 \not\models \varphi$

wrong.
Correct or wrong?

If $s_1, s_2$ are not **CTL** equivalent then there exists a **CTL** formula $\Phi$ with $s_1 \models \Phi$ and $s_2 \not\models \Phi$

correct.

If $s_1, s_2$ are not **LTL** equivalent then there exists a **LTL** formula $\varphi$ with $s_1 \models \varphi$ and $s_2 \not\models \varphi$

wrong.

$Traces(s_2) \subset Traces(s_1)$
Correct or wrong?

If $s_1$, $s_2$ are not **CTL** equivalent then there exists a **CTL** formula $\Phi$ with $s_1 \models \Phi$ and $s_2 \not\models \Phi$

correct.

If $s_1$, $s_2$ are not **LTL** equivalent then there exists a **LTL** formula $\varphi$ with $s_1 \models \varphi$ and $s_2 \not\models \varphi$

wrong.

$\text{Traces}(s_2) \subset \text{Traces}(s_1)$

hence: $s_1 \models \varphi$ implies $s_2 \models \varphi$
If $\mathcal{T}$ is a finite TS then, for all states $s_1, s_2$ in $\mathcal{T}$:

if $s_1, s_2$ are **CTL** equivalent then $s_1 \sim_\mathcal{T} s_2$
If $\mathcal{T}$ is a finite TS then, for all states $s_1$, $s_2$ in $\mathcal{T}$:

if $s_1$, $s_2$ are CTL equivalent then $s_1 \sim_{\mathcal{T}} s_2$

**Proof:** show that

$\mathcal{R} \overset{\text{def}}{=} \{ (s_1, s_2) : s_1, s_2 \text{ satisfy the same CTL formulas} \}$

is a bisimulation, i.e., for all $(s_1, s_2) \in \mathcal{R}$:

1. $L(s_1) = L(s_2)$
2. if $s_1 \rightarrow t_1$ then there exists a transition $s_2 \rightarrow t_2$ s.t. $(t_1, t_2) \in \mathcal{R}$
Example: CTL master formulas

\[ ^\wedge = \{ a \} \]
\[ \equiv \{ b \} \]
\[ \equiv \emptyset \]
Example: CTL master formulas

\[ \hat{=} = \{a\} \]
\[ \hat{=} = \{b\} \]
\[ \hat{=} = \emptyset \]

bisimulation equivalence \( \sim_T \)
\[ = \{(v_1, v_2), (w_1, w_2), \ldots\} \]
Example: CTL master formulas

\[ \hat{=} = \{a\} \]
\[ \hat{=} = \{b\} \]
\[ \hat{=} = \emptyset \]

bisimulation equivalence \( \sim_T \)
\[ = \{(v_1, v_2), (w_1, w_2), \ldots\} \]

but \( u_1 \not\sim_T u_2 \)
Example: CTL master formulas

\[
\begin{align*}
\hat{=} &= \{a\} \\
\hat{=} &= \{b\} \\
\hat{=} &= \emptyset
\end{align*}
\]

bisimulation equivalence \(\sim_T\)

\[
= \{(v_1, v_2), (w_1, w_2), \ldots\}
\]

but \(u_1 \not\sim_T u_2\)

as \(u_1 \rightarrow \{w_1, w_2\}\)

\(u_2 \not\rightarrow \{w_1, w_2\}\)
Example: CTL master formulas

CTL master formulas:

$w_1, w_2 \models ?$

$v_1, v_2 \models ?$

$u_1 \models ?$

$u_2 \models ?$

bisimulation equivalence $\sim_T$

$= \{(v_1, v_2), (w_1, w_2), \ldots\}$
Example: CTL master formulas

bisimulation equivalence $\sim_T = \{(v_1, v_2), (w_1, w_2), \ldots\}$

CTL master formulas:

$w_1, w_2 \models b$

$v_1, v_2 \models ?$

$u_1 \models ?$

$u_2 \models ?$
Example: CTL master formulas

 CTL master formulas:

\[
\begin{align*}
w_1, w_2 & \models b \\
v_1, v_2 & \models \neg a \land \neg b \\
u_1 & \models ? \\
u_2 & \models ? 
\end{align*}
\]

bisimulation equivalence \( \sim_T \)

\[
\{ (v_1, v_2), (w_1, w_2), \ldots \}
\]
Example: CTL master formulas

\[ \hat{=} = \{ a \} \]
\[ \hat{=} = \{ b \} \]
\[ \hat{=} = \emptyset \]

\[ v_1, v_2 \models \neg a \land \neg b \]
\[ u_1 \models (\exists \bigcirc b) \land a \]
\[ u_2 \models ? \]

Bisimulation equivalence \( \sim_T \)

\[ = \{(v_1, v_2), (w_1, w_2), \ldots\} \]
Example: CTL master formulas

\[ u_1 \bowtie u_1 \bowtie u_1 \bowtie u_2 \bowtie u_2 \bowtie u_2 \bowtie \]  

\[ v_1 \bowtie v_1 \bowtie v_2 \bowtie v_2 \bowtie v_2 \bowtie \]  

\[ w_1 \bowtie w_1 \bowtie w_2 \bowtie w_2 \bowtie w_2 \bowtie \]  

\[ ^\mathcal{E}=\{a\} \]  

\[ ^\mathcal{E}=\{b\} \]  

\[ ^\mathcal{E}=\emptyset \]  

bisimulation equivalence \( \sim_T \)  
\[ = \{ (v_1, v_2), (w_1, w_2), \ldots \} \]  

**CTL master formulas:**

\[ w_1, w_2 \models b \]  

\[ v_1, v_2 \models \neg a \land \neg b \]  

\[ u_1 \models (\exists \mathcal{E} b) \land a \]  

\[ u_2 \models (\neg \exists \mathcal{E} b) \land a \]
...master formulas for $\sim_T$-classes?

$AP = \{\text{blue, red}\}$
...master formulas for $\sim_T$-classes?

$$s_1 \sim_T s_2 \not\sim_T u$$

$$AP = \{ \text{blue, red} \}$$
...master formulas for $\sim_T$-classes?

$AP = \{ \text{blue, red} \}$

$s_1 \sim_T s_2 \not\sim_T u$

$\Phi_w = ?$

$\Phi_C = ?$

where $C = \{s_1, s_2\}$

$\Phi_u = ?$
...master formulas for $\sim_T$-classes?

$AP = \{\text{blue, red}\}$

$s_1 \sim_T s_2 \not\sim_T u$

$\Phi_w = \text{red}$

$\Phi_C = \text{?}$

$\Phi_u = \text{?}$

where $C = \{s_1, s_2\}$
...master formulas for $\sim_T$-classes?

$\Phi_w = \text{red}$

$\Phi_C = \text{blue} \land \forall \bigcirc \text{blue}$ where $C = \{s_1, s_2\}$

$\Phi_u = ?$

$AP = \{\text{blue}, \text{red}\}$

$s_1 \sim_T s_2 \not
\sim_T u$
...master formulas for $\sim_T$-classes?

$AP = \{ \text{blue, red} \}$

$s_1 \sim_T s_2 \not\sim_T u$

\[
\Phi_w = \text{red} \\
\Phi_C = \text{blue} \land \forall \bigcirc \text{blue} \quad \text{where} \ C = \{s_1, s_2\} \\
\Phi_u = \exists \bigcirc \text{red}
\]
If $T$ is a finite TS then, for all states $s_1$, $s_2$ in $T$: if $s_1$, $s_2$ are CTL equivalent then $s_1 \sim_T s_2$
If $\mathcal{T}$ is a finite TS then, for all states $s_1, s_2$ in $\mathcal{T}$:

if $s_1, s_2$ are **CTL** equivalent then $s_1 \sim_T s_2$

- wrong for infinite TS
If $\mathcal{T}$ is a finite TS then, for all states $s_1, s_2$ in $\mathcal{T}$:
if $s_1, s_2$ are CTL equivalent then $s_1 \sim_\mathcal{T} s_2$

- wrong for infinite TS
- but also holds for finitely branching TS
If $\mathcal{T}$ is a finite TS then, for all states $s_1, s_2$ in $\mathcal{T}$:

- if $s_1, s_2$ are CTL equivalent then $s_1 \sim_T s_2$

- wrong for infinite TS

- but also holds for finitely branching TS

possibly infinite-state TS such that

- the number of initial states is finite
- for each state the number of successors is finite
CTL equivalence $\iff$ bisimulation equivalence

Let $\mathcal{T} = (S, Act, \rightarrow, S_0, AP, L)$ be finitely branching.
CTL equivalence $\iff$ bisimulation equivalence

Let $T = (S, \text{Act}, \rightarrow, S_0, AP, L)$ be finitely branching.

* $S_0$ is finite
* $Post(s)$ is finite for all $s \in S$
Let $\mathcal{T} = (S, \text{Act}, \rightarrow, S_0, AP, L)$ be finitely branching.

* $S_0$ is finite
* $\text{Post}(s)$ is finite for all $s \in S$

Then, for all states $s_1, s_2$ in $\mathcal{T}$:

if $s_1, s_2$ are CTL equivalent then $s_1 \sim_\mathcal{T} s_2$
CTL equivalence $\iff$ bisimulation equivalence

Let $\mathcal{T} = (S, \text{Act}, \rightarrow, S_0, AP, L)$ be finitely branching.

* $S_0$ is finite
* $\text{Post}(s)$ is finite for all $s \in S$

Then, for all states $s_1, s_2$ in $\mathcal{T}$:

if $s_1, s_2$ are CTL equivalent then $s_1 \sim_\mathcal{T} s_2$

Proof: as for finite TS.
Let $\mathcal{T} = (S, \text{Act}, \rightarrow, S_0, AP, L)$ be finitely branching.

* $S_0$ is finite
* $\text{Post}(s)$ is finite for all $s \in S$

Then, for all states $s_1, s_2$ in $\mathcal{T}$:

If $s_1, s_2$ are CTL equivalent then $s_1 \sim_\mathcal{T} s_2$

Proof: as for finite TS. Amounts showing that

$\mathcal{R} \overset{\text{def}}{=} \{ (s_1, s_2) : s_1, s_2 \text{ satisfy the same CTL formulas} \}$

is a bisimulation.
CTL equivalence $\implies$ bisimulation equivalence
c
If $\mathcal{T}$ is a finitely branching TS then for all states $s_1$, $s_2$:

if $s_1$, $s_2$ are CTL equivalent then $s_1 \sim_\mathcal{T} s_2$

Proof: show that

$\mathcal{R} \overset{\text{def}}{=} \{(s_1, s_2) : s_1$, $s_2$ satisfy the same CTL formulas $\}$

is a bisimulation, i.e., for $(s_1, s_2) \in \mathcal{R}$:

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

(2) if $s_1 \rightarrow t_1$ then there exists a transition $s_2 \rightarrow t_2$

s.t. $(t_1, t_2) \in \mathcal{R}$
Summary: CTL/CTL* and bisimulation
Summary: CTL/CTL* and bisimulation

Let $\mathcal{T}$ be a finitely branching TS without terminal states, and $s_1$, $s_2$ states in $\mathcal{T}$. Then:

\[
\begin{align*}
    s_1 & \sim_{\mathcal{T}} s_2 \\
     \text{iff} & \quad s_1 \text{ and } s_2 \text{ are CTL equivalent} \\
    \text{iff} & \quad s_1 \text{ and } s_2 \text{ are CTL* equivalent}
\end{align*}
\]
Summary: CTL/CTL* and bisimulation

CTL equivalence \[\sim\] for arbitrary TS

for finite TS: proof relies on master formulas

CTL is a sublogic of CTL*

CTL equivalence

CTL* equivalence
Summary: CTL/CTL* and bisimulation

- CTL/CTL* and bisimulation equivalence
- Proof for finitely branching transition systems: "local" master formulas
- CTL equivalence
- CTL* equivalence
- CTL is a sublogic of CTL*
- For arbitrary TS
so far: we considered

- **CTL/CTL**\(^*\) equivalence
- bisimulation equivalence \(\sim_T\)

for the *states* of a single transition system \(T\)
If $\mathcal{T}_1$, $\mathcal{T}_2$ are finitely branching TS over $AP$ without terminal states then:

\[
\mathcal{T}_1 \sim \mathcal{T}_2
\]

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

iff $\mathcal{T}_1$ and $\mathcal{T}_2$ satisfy the same $\text{CTL}^*$ formulas
Summary: equivalences

LTL equivalence

CTL equivalence

CTL* equivalence

for finitely branching TS
Summary: equivalences

- Trace equivalence
- Bisimulation equivalence
- LTL 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

-equivalence w.r.t. LTL safety properties

LTL equivalence

CTL equivalence

CTL* equivalence

for finitely branching TS
Correct or wrong?

Let $\mathcal{T}$ be a finite TS without terminal states and $s_1, s_2$ states of $\mathcal{T}$.

If $s_1, s_2$ satisfy the same $\text{CTL} \cup \text{CTL}$ formulas then $s_1 \sim_{\mathcal{T}} s_2$. 

101 / 167
Let $T$ be a finite TS without terminal states and $s_1$, $s_2$ states of $T$.

If $s_1$, $s_2$ satisfy the same $\text{CTL}_U$ formulas then $s_1 \sim_T s_2$.

where $\text{CTL}_U \equiv \text{CTL}$ without until operator $U$
Correct or wrong?

Let $\mathcal{T}$ be a finite TS without terminal states and $s_1, s_2$ states of $\mathcal{T}$.

If $s_1, s_2$ satisfy the same $CTL\setminus U$ formulas then $s_1 \sim_T s_2$.

where $CTL\setminus U \equiv CTL$ without until operator $U$

correct.
Let $\mathcal{T}$ be a finite TS without terminal states and $s_1, s_2$ states of $\mathcal{T}$.

If $s_1, s_2$ satisfy the same $\text{CTL}_{\mathcal{U}}$ formulas then $s_1 \sim_\mathcal{T} s_2$.

where $\text{CTL}_{\mathcal{U}} \equiv \text{CTL}$ without until operator $\mathcal{U}$

**correct.** see the proof

"$\text{CTL}$ equivalence $\implies$ bisimulation equivalence"
\textbf{CTL}_\text{U}-\text{equivalence} \Rightarrow \text{bisimulation equivalence} \hspace{1cm} \text{ctleq5.2-11}

Let $\mathcal{T}$ be a finite TS without terminal states and $s_1, s_2$ states of $\mathcal{T}$.

If $s_1, s_2$ satisfy the same \text{CTL}_\text{U} formulas then $s_1 \sim_T s_2$.

\textit{Proof.} Show that \text{CTL}_\text{U} equivalence is a \text{bisimulation}
\[ \text{CTL}_U\text{-equivalence} \Rightarrow \text{bisimulation equivalence} \]

Let \( T \) be a finite TS without terminal states and \( s_1, s_2 \) states of \( T \).

If \( s_1, s_2 \) satisfy the same \( \text{CTL}_U \) formulas then \( s_1 \sim_T s_2 \).

**Proof.** Show that \( \text{CTL}_U \) equivalence is a bisimulation

- labeling condition only uses atomic propositions
$\textbf{CTL}_U$-equivalence $\Rightarrow$ bisimulation equivalence

Let $\mathcal{T}$ be a finite TS without terminal states and $s_1$, $s_2$ states of $\mathcal{T}$.

If $s_1$, $s_2$ satisfy the same $\textbf{CTL}_U$ formulas then $s_1 \sim_T s_2$.

$\textbf{Proof}$. Show that $\textbf{CTL}_U$ equivalence is a bisimulation

- labeling condition only uses atomic propositions
- simulation condition can be established by $\textbf{CTL}_U$ master formulas of the form:
Let $\mathcal{T}$ be a finite TS without terminal states and $s_1$, $s_2$ states of $\mathcal{T}$.

If $s_1$, $s_2$ satisfy the same $\text{CTL}_U$ formulas then $s_1 \sim_T s_2$.

**Proof.** Show that $\text{CTL}_U$ equivalence is a bisimulation

- labeling condition only uses atomic propositions
- simulation condition can be established by $\text{CTL}_U$ master formulas of the form:

$$\exists \bigcirc \Phi_C$$

where $\Phi_C = \bigwedge_D \Phi_{C,D}$
**CTL\_U-equivalence ⇒ bisimulation equivalence**

Let $\mathcal{T}$ be a finite TS without terminal states and $s_1, s_2$ states of $\mathcal{T}$.

If $s_1, s_2$ satisfy the same CTL\_U formulas then $s_1 \sim_T s_2$.

Proof. Show that CTL\_U equivalence is a bisimulation

- labeling condition only uses atomic propositions
- simulation condition can be established by CTL\_U master formulas of the form:

$$\exists \bigcirc \Phi_C \quad \text{where} \quad \Phi_C = \bigwedge_D \Phi_{C,D}$$

and $\text{Sat}(\Phi_{C,D}) \subseteq C \setminus D$
Let $T$ be a finite TS without terminal states.

$T$ and its bisimulation quotient $T/\sim$ satisfy the same $\text{CTL}^*$ formulas.
Correct or wrong?

Let $\mathcal{T}$ be a finite TS without terminal states.

$\mathcal{T}$ and its bisimulation quotient $\mathcal{T}/\sim$ satisfy the same $\text{CTL}^*$ formulas.

correct.
Let $T$ be a finite TS without terminal states.

$T$ and its bisimulation quotient $T/\sim$ satisfy the same $\text{CTL}^*$ formulas.

correct. Recall that $T \sim T/\sim$
Correct or wrong?

Let $T$ be a finite TS without terminal states.

$T$ and its bisimulation quotient $T/\sim$ satisfy the same $CTL^*$ formulas.

**correct.** Recall that $T \sim T/\sim$ as

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

is a bisimulation for $(T, T/\sim)$

here: $[s] = \sim_T$-equivalence class of state $s$
Let $\mathcal{T}$ be a finite TS without terminal states and let $\textit{fair}$ be a $\text{CTL}$ fairness assumption.

If $s_1 \sim_{\mathcal{T}} s_2$ then for all $\text{CTL}$ formulas $\Phi$:

$s_1 \models_{\textit{fair}} \Phi$ iff $s_2 \models_{\textit{fair}} \Phi$
Let \( \mathcal{T} \) be a finite TS without terminal states and let \( \text{fair} \) be a \textbf{CTL} fairness assumption.

If \( s_1 \sim_\mathcal{T} s_2 \) then for all \textbf{CTL} formulas \( \Phi \):

\[
\models_{\text{fair}} \Phi \quad \text{iff} \quad \models_{\text{fair}} \Phi
\]

\text{correct}
Let $\mathcal{T}$ be a finite TS without terminal states and let $\textit{fair}$ be a CTL fairness assumption.

If $s_1 \sim_{\mathcal{T}} s_2$ then for all CTL formulas $\Phi$:

$$s_1 \models_{\textit{fair}} \Phi \iff s_2 \models_{\textit{fair}} \Phi$$

correct, as $\models_{\textit{fair}}$ is “CTL*-definable”
Let $\mathcal{T}$ be a finite TS without terminal states and let $\textit{fair}$ be a $\text{CTL}$ fairness assumption.

If $s_1 \sim_{\mathcal{T}} s_2$ then for all $\text{CTL}$ formulas $\Phi$:

$$s_1 \models_{\text{fair}} \Phi \text{ iff } s_2 \models_{\text{fair}} \Phi$$

correct, as $\models_{\text{fair}}$ is "$\text{CTL}^*$-definable"

For each $\text{CTL}^*$ state formula $\Phi$ there exists a $\text{CTL}^*$ formula $\psi$ s.t. $s \models \psi$ iff $s \models_{\text{fair}} \Phi$
Let \( \mathcal{T} \) be a finite TS without terminal states and let \( \textit{fair} \) be a CTL fairness assumption.

If \( s_1 \sim_{\mathcal{T}} s_2 \) then for all CTL formulas \( \Phi \):

\[
s_1 \models_{\textit{fair}} \Phi \text{ iff } s_2 \models_{\textit{fair}} \Phi
\]

Correct, as \( \models_{\textit{fair}} \) is “CTL*-definable”

For each CTL* state formula \( \Phi \) there exists a CTL* formula \( \Psi \) s.t. \( s \models \Psi \) iff \( s \models_{\textit{fair}} \Phi \)

Example: for \( \Phi = \exists \boxdot (a \land \forall \diamond b) \)
Correct or wrong?

Let $T$ be a finite TS without terminal states and let $\textit{fair}$ be a CTL fairness assumption.

If $s_1 \sim_T s_2$ then for all CTL formulas $\phi$:

$$s_1 \models_{\textit{fair}} \phi \iff s_2 \models_{\textit{fair}} \phi$$

Correct, as $\models_{\textit{fair}}$ is “CTL*-definable”

For each CTL* state formula $\phi$ there exists a CTL* formula $\psi$ s.t.

$$s \models \psi \iff s \models_{\textit{fair}} \phi$$

Example: for $\phi = \exists \Box(a \land \forall \Diamond b)$

$$\psi = \exists (\textit{fair} \land \Box(a \land \forall (\textit{fair} \rightarrow \Diamond b)))$$
Let $T$ be a finite TS over $AP$ without terminal states.

If $s_1 \sim_T s_2$ then for all LT properties $E \subseteq (2^{AP})^\omega$:

$s_1 \models E$ iff $s_2 \models E$
Let $T$ be a finite TS over $AP$ without terminal states.

If $s_1 \sim_T s_2$ then for all LT properties $E \subseteq (2^{AP})^\omega$:

$$s_1 \models E \text{ iff } s_2 \models E$$

correct.
Correct or wrong?

Let $\mathcal{T}$ be a finite TS over $\text{AP}$ without terminal states.

If $s_1 \sim_T s_2$ then for all $\text{LT}$ properties $E \subseteq (2^\text{AP})^\omega$: $s_1 \models E$ iff $s_2 \models E$

is correct.

Note that:

(1) $s_1 \sim_T s_2 \implies \text{Traces}(s_1) = \text{Traces}(s_2)$
Correct or wrong?

Let $T$ be a finite TS over $AP$ without terminal states.

If $s_1 \sim_T s_2$ then for all LT properties $E \subseteq (2^{AP})^\omega$:

$$s_1 \models E \iff s_2 \models E$$

correct.

Note that:

1. $s_1 \sim_T s_2 \implies \text{Traces}(s_1) = \text{Traces}(s_2)$
2. $s \models E \iff \text{Traces}(s) \subseteq E$
Correct or wrong?

Let $\mathcal{F}$ be an action-based strong fairness assumption e.g., strong fairness for a single action $\alpha$

If $s_1 \sim T s_2$ then for all LT properties $E \subseteq (2^{AP})^\omega$:

$$s_1 \models_\mathcal{F} E \text{ iff } s_2 \models_\mathcal{F} E$$
Correct or wrong?

Let $\mathcal{F}$ be an action-based strong fairness assumption

e.g., strong fairness for a single action $\alpha$

If $s_1 \sim_T s_2$ then for all $\mathbf{LT}$ properties $E \subseteq (2^{AP})^\omega$:

\[
s_1 \models \mathcal{F} E \text{ iff } s_2 \models \mathcal{F} E
\]

wrong.
Correct or wrong?

Let $\mathcal{F}$ be an action-based strong fairness assumption e.g., strong fairness for a single action $\alpha$

If $s_1 \sim_T s_2$ then for all LT properties $E \subseteq (2^{AP})^\omega$:

$$s_1 \models_{\mathcal{F}} E \text{ iff } s_2 \models_{\mathcal{F}} E$$

wrong.

$\mathcal{F} \equiv$ strong fairness assumption for action $\alpha$
Correct or wrong?

Let $\mathcal{F}$ be an action-based strong fairness assumption e.g., strong fairness for a single action $\alpha$

If $s_1 \sim_T s_2$ then for all LT properties $E \subseteq (2^{AP})^\omega$:

$$s_1 \models_{\mathcal{F}} E \text{ iff } s_2 \models_{\mathcal{F}} E$$

wrong.

$$E \equiv \diamond \text{red}$$

$\mathcal{F} \equiv$ strong fairness assumption for action $\alpha$
Correct or wrong?

Let $\mathcal{F}$ be an action-based strong fairness assumption e.g., strong fairness for a single action $\alpha$

If $s_1 \sim_T s_2$ then for all $\text{LT}$ properties $E \subseteq (2^{AP})^\omega$: $s_1 \models_{\mathcal{F}} E$ iff $s_2 \models_{\mathcal{F}} E$

Wrong.

$E \equiv \Diamond \text{red}$

$s_1 \models_{\mathcal{F}} E$

$s_2 \not\models_{\mathcal{F}} E$

$\mathcal{F} \equiv \text{strong fairness assumption for action } \alpha$
Let $\mathcal{F}$ be an action-based strong fairness assumption

If $s_1 \sim_T s_2$ then for all LT properties $E \subseteq (2^{AP})^\omega$: $s_1 \models_\mathcal{F} E$ iff $s_2 \models_\mathcal{F} E$

wrong.

If $s_1 \sim_T s_2$ then for all safety properties $E \subseteq (2^{AP})^\omega$: $s_1 \models_\mathcal{F} E$ iff $s_2 \models_\mathcal{F} E$
Correct or wrong?

Let $\mathcal{F}$ be an action-based strong fairness assumption

If $s_1 \sim_T s_2$ then for all LT properties $E \subseteq (2^{AP})^\omega$:

$$s_1 \models_{\mathcal{F}} E \iff s_2 \models_{\mathcal{F}} E$$

Wrong.

If $s_1 \sim_T s_2$ then for all safety properties $E \subseteq (2^{AP})^\omega$:

$$s_1 \models_{\mathcal{F}} E \iff s_2 \models_{\mathcal{F}} E$$

Correct.
Let $\mathcal{F}$ be an action-based strong fairness assumption

If $s_1 \sim_T s_2$ then for all LT properties $E \subseteq (2^{AP})^\omega$:

$$s_1 \models \mathcal{F} E \iff s_2 \models \mathcal{F} E$$

wrong.

If $s_1 \sim_T s_2$ then for all safety properties $E \subseteq (2^{AP})^\omega$:

$$s_1 \models \mathcal{F} E \iff s_2 \models \mathcal{F} E$$

correct.

- realizable fairness irrelevant for safety properties
Let $\mathcal{F}$ be an action-based strong fairness assumption.

If $s_1 \sim_T s_2$ then for all LT properties $E \subseteq (2^{AP})^\omega$:  
$s_1 \models_{\mathcal{F}} E$ iff $s_2 \models_{\mathcal{F}} E$

Wrong.

If $s_1 \sim_T s_2$ then for all safety properties $E \subseteq (2^{AP})^\omega$:  
$s_1 \models_{\mathcal{F}} E$ iff $s_2 \models_{\mathcal{F}} E$

Correct.

- realizable fairness irrelevant for safety properties
- strong action-based fairness assumptions are realizable