Parallel Composition and Bisimulations

jana.wagemaker@ru.nl

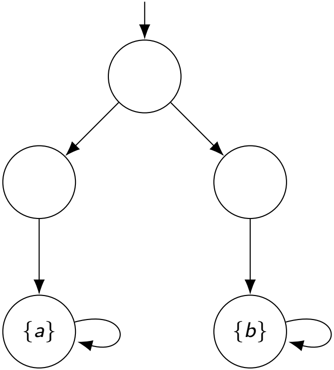
Slides credits: Marnix Suilen

Model Checking 2025

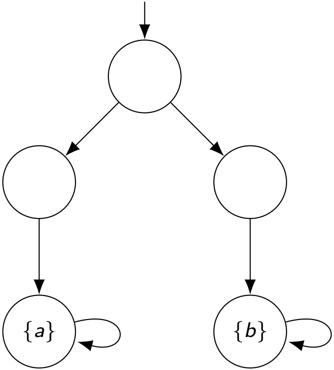
Overview

1. Bisimulations for transition systems,
2. Bisimulation minimization ('quotient system'),
3. CTL(*)-equivalence and bisimilarity (main point),
4. Parallel composition (and bisimulation),
5. Some final remarks.

A canonical example



A canonical example



$\varphi = \exists(\bigcirc(\exists \bigcirc a \wedge \exists \bigcirc b))$

What we want:

Define a relation $\sim$ such that $T_1 \sim T_2$ iff ($T_1 \models \varphi$ iff $T_2 \models \varphi$) for "relevant" properties φ.

What we want:

Define a relation $\sim$ such that $T_1 \sim T_2$ iff ($T_1 \models \varphi$ iff $T_2 \models \varphi$) for "relevant" properties φ.

Why?

What we want:

Define a relation $\sim$ such that $T_1 \sim T_2$ iff ($T_1 \models \varphi$ iff $T_2 \models \varphi$) for "relevant" properties φ.

Why?

- For designing complex systems, you need to be able to compare 2 transition systems.
- To analyse complex systems:
  - Minimisation based on bisimulation
  - Correspondence between CTL/CTL* equivalence and bisimilarity

Transition systems

Definition (Transition system)

A transition system is a tuple $TS = (S, Act, \longrightarrow, I, AP, L)$ where

- S is a set of states,
- Act is a set of actions,
- $\longrightarrow \subseteq S \times Act \times S$ is a transition relation,
- $I \subseteq S$ is a set of initial states,
- AP is a set of atomic propositions,
- $L \colon S \to 2^{AP}$ is a labeling function.

TS is finite when S, Act and AP are finite.

For a transitions $(s, a, s') \in \longrightarrow$ we write $s \xrightarrow{a} s'$.

Bisimilarity: the idea

Bisimulations are about **behavioral equivalence**.

Two states are bisimilar if they behave the same.

Bisimilarity: the idea

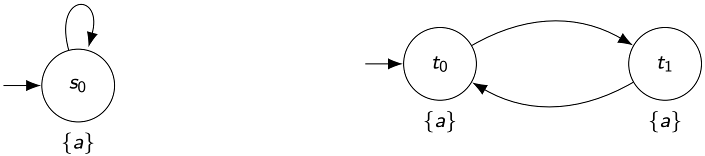
Bisimulations are about **behavioral equivalence**.

Two states are bisimilar if they behave the same.

Very general: both states should have the same label, and their successors should again behave the same.

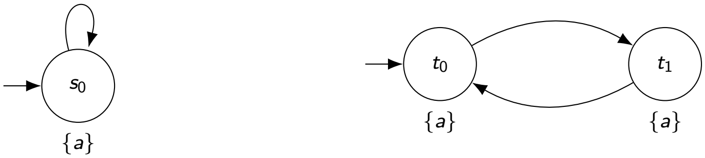
A trivial example

The two TS below are bisimilar. Why?

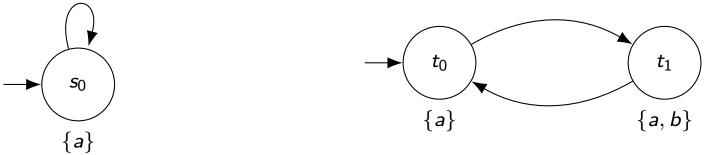


A trivial example

The two TS below are bisimilar. Why?



And what about the following TS?



Label-based Bisimulation on TS

Definition (Bisimulation between TS)

Given $TS_i = (S_i, Act, \longrightarrow_i, I_i, AP_i, L_i)$ for $i = 1, 2$, a **bisimulation** for (TS_1, TS_2) is a relation $R \subseteq S_1 \times S_2$ such that:

1. $\forall s_1 \in I_1 . \exists s_2 \in I_2 . (s_1, s_2) \in R$ and $\forall s_2 \in I_2 . \exists s_1 \in I_1 . (s_1, s_2) \in R$
2. for all $(s_1, s_2) \in R$ we have:
    - $L_1(s_1) = L_2(s_2)$
    - If $s_1 \longrightarrow_1 s_1'$ then there exists $s_2' \in S_2$ with $s_2 \longrightarrow_2 s_2'$ and $(s_1', s_2') \in R$,
    - If $s_2 \longrightarrow_2 s_2'$ then there exists $s_1' \in S_1$ with $s_1 \longrightarrow_1 s_1'$ and $(s_1', s_2') \in R$.

Bisimulation equivalence

T_1 and T_2 are **bisimilar**, denoted as $T_1 \sim T_2$, if there exists a bisimulation R for (T_1, T_2).

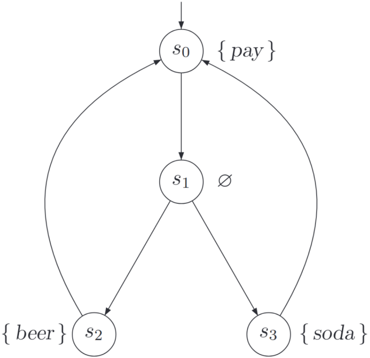
Bisimulation equivalence

T_1 and T_2 are **bisimilar**, denoted as $T_1 \sim T_2$, if there exists a bisimulation R for (T_1, T_2).

Let s_1, s_2 in state space of T_1, T_2 respectively. s_1 and s_2 are **bisimilar**, denoted as $s_1 \sim s_2$, if there exists a bisimulation R for (T_1, T_2) with $(s_1, s_2) \in R$.

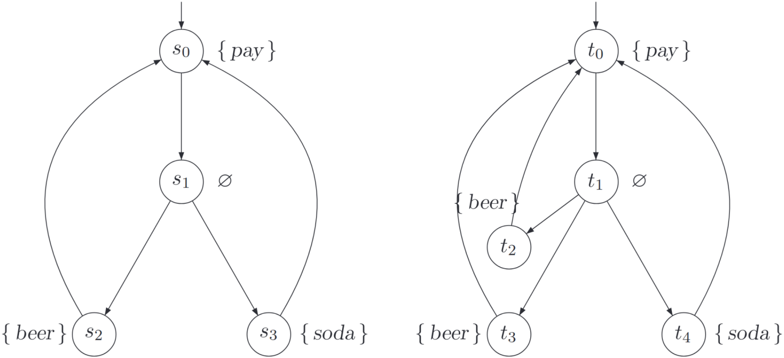
Example 1

The following two TS are bisimilar. Why?



Example 1

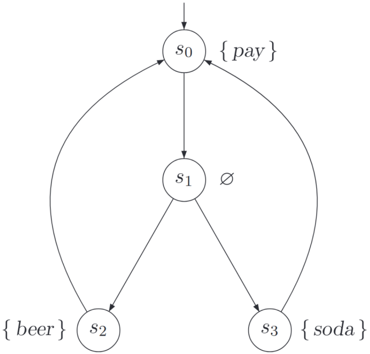
The following two TS are bisimilar. Why?



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

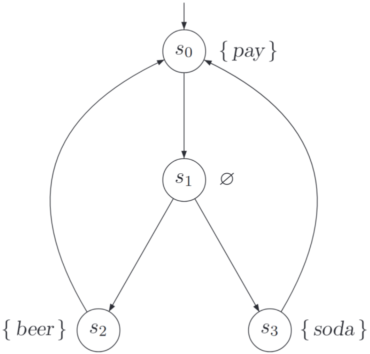
Example 2

Are the following two TS bisimilar? Why?



Example 2

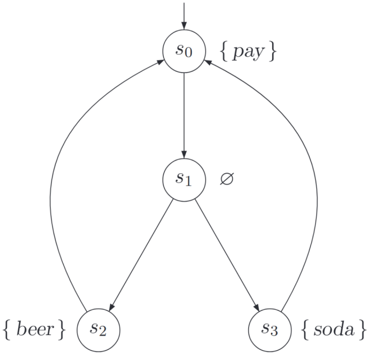
Are the following two TS bisimilar? Why?



No. By the labels we need $s_1 \sim u_1$ or $s_1 \sim u_2$, but neither u_1 nor u_2 have successors for both $\{beer\}$ and $\{soda\}$.

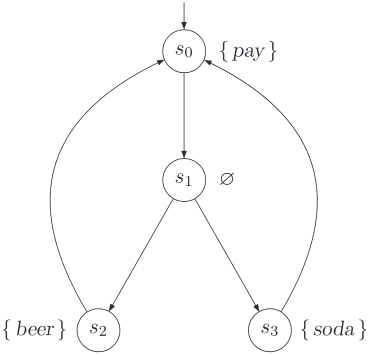
Bisimulation and trace equivalence

Are the following two TS trace equivalent?



Bisimulation and trace equivalence

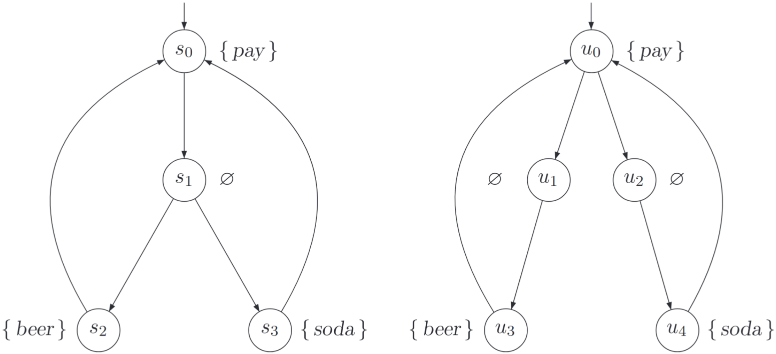
Are the following two TS trace equivalent?



Yes.

Bisimulation and trace equivalence

Are the following two TS trace equivalent?



Yes.

In general, we have $TS_1 \sim TS_2 \implies Traces(TS_1) = Traces(TS_2)$,
but **not** $Traces(TS_1) = Traces(TS_2) \implies TS_1 \sim TS_2$.

Bisimulation on paths

Notation: a path is a sequence of successive states $\pi = s_0 s_1 s_2 \ldots$
$Paths(s)$ denotes the set of all paths starting in s.

Bisimulation on paths

Notation: a path is a sequence of successive states $\pi = s_0 s_1 s_2 \ldots$
$Paths(s)$ denotes the set of all paths starting in s.

For a path π we write $\pi[i]$ for the i-th state of π, hence $\pi[i] \in S$.

And $\pi[i \ldots]$ denotes the path fragment starting at $\pi[i]$, hence $\pi[i \ldots] \in Paths(\pi[i])$.

Bisimulation on paths

Notation: a path is a sequence of successive states $\pi = s_0 s_1 s_2 \ldots$
$Paths(s)$ denotes the set of all paths starting in s.

For a path π we write $\pi[i]$ for the i-th state of π, hence $\pi[i] \in S$.

And $\pi[i \ldots]$ denotes the path fragment starting at $\pi[i]$, hence $\pi[i \ldots] \in Paths(\pi[i])$.

Lemma (Path-lifting)

Let R be a bisimulation and $(s_1, s_2) \in R$. Then for every path $\pi_1 \in Paths(s_1)$ there exists a path $\pi_2 \in Paths(s_2)$ of the same length, such that $(\pi_1[i], \pi_2[i]) \in R$ for all i.

Bisimulation vs trace equivalence

$T_1 \sim T_2 \Rightarrow \mathit{Traces}(T_1) = \mathit{Traces}(T_2)$ (via path lifting)

$\mathit{Traces}(T_1) = \mathit{Traces}(T_2) \not\Rightarrow T_1 \sim T_2$

Properties of $\sim$

Theorem

$\sim$ *is an equivalence relation:*

Properties of $\sim$

Theorem

$\sim$ *is an equivalence relation:*

- *Reflexivity:* $T \sim T$ *for all* T

Properties of $\sim$

Theorem

$\sim$ is an equivalence relation:

- *Reflexivity: $T \sim T$ for all T*

  If S is state space of T, take $R = \{(s, s) \mid s \in S\}$

Properties of $\sim$

Theorem

$\sim$ is an equivalence relation:

- *Reflexivity: $T \sim T$ for all T*
- *Symmetry: $T_1 \sim T_2$ implies $T_2 \sim T_1$*

Properties of $\sim$

Theorem

$\sim$ is an equivalence relation:

- *Reflexivity: $T \sim T$ for all T*
- *Symmetry: $T_1 \sim T_2$ implies $T_2 \sim T_1$*

  If R is bisimulation for T_1, T_2, then take $R^{-1} = \{(s_2, s_1) \mid (s_1, s_2) \in R\}$

Properties of $\sim$

Theorem

$\sim$ *is an equivalence relation:*

- *Reflexivity:* $T \sim T$ *for all* 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 $\sim$

Theorem

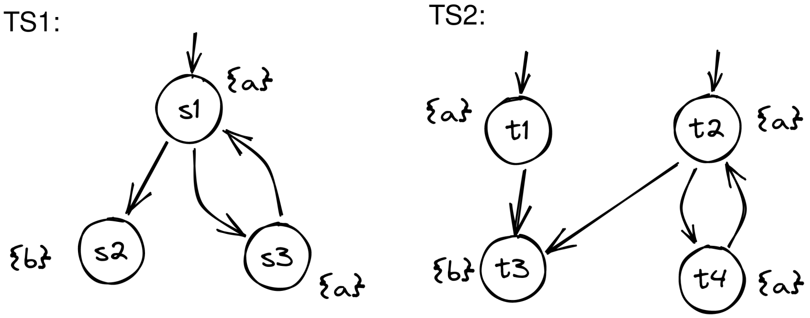
$\sim$ *is an equivalence relation:*

- *Reflexivity:* $T \sim T$ *for all* 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* $T_1 \sim T_3$

  If R_1 *is bisimulation for* (T_1, T_2) *and* R_2 *is bisimulation for* T_2, T_3*,*
  take $R = \{(s_1, s_3) \mid (s_1, s_2) \in R_1, (s_2, s_3) \in R_2\}$

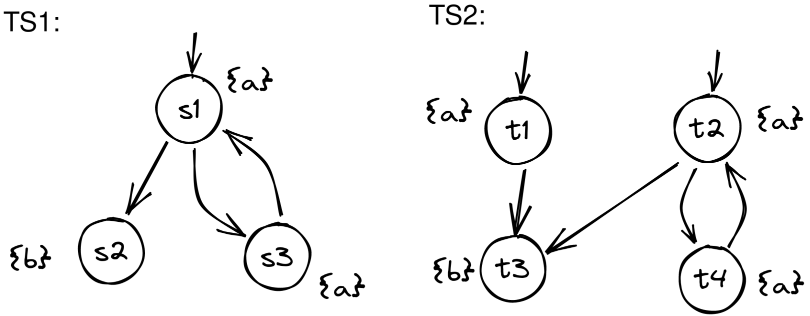
Example 1

Are the following two TS bisimilar?



Example 1

Are the following two TS bisimilar?

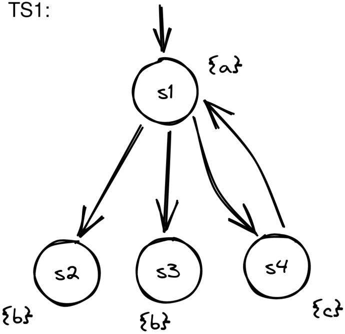


No. $s_1 \to s_3$ with label $\{a\}$ but for t_1 there is not successor with label $\{a\}$.

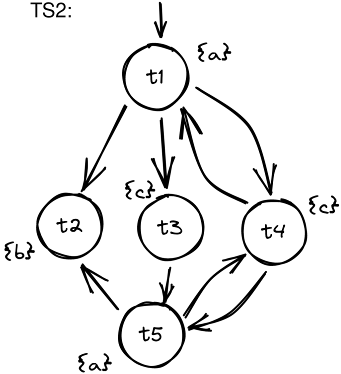
Example 2

Are the following two TS bisimilar?

TS1:

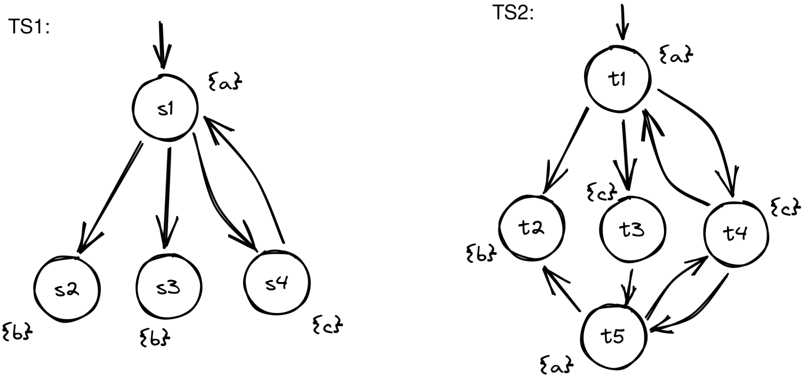


TS2:



Example 2

Are the following two TS bisimilar?



Yes. $R = \{(s_2, t_2), (s_3, t_2), (s_1, t_1), (s_1, t_5), (s_4, t_3), (s_4, t_4)\}$.

Bisimulation on a single TS

To check $T \models \varphi$, we want to check $T/_\sim \models \varphi$.

Bisimulation on a single TS

Definition (Bisimulation between states)

Given $T = (S, Act, \longrightarrow, I, AP, L)$, a **bisimulation** between states is a relation $R \subseteq S \times S$ such that:

1. for all $(s_1, s_2) \in R$ we have:
    - $L(s_1) = L(s_2)$,
    - If $s_1 \longrightarrow s_1'$ then there exists $s_2' \in S$ with $s_2 \longrightarrow s_2'$ and $(s_1', s_2') \in R$,
    - If $s_2 \longrightarrow s_2'$ then there exists $s_1' \in S$ with $s_1 \longrightarrow s_1'$ and $(s_1', s_2') \in R$.

$s_1 \sim_T s_2$ iff $T_{s_1} \sim T_{s_2}$ iff there exists a bisimulation R for T such that $(s_1, s_2) \in R$.

Properties of $\sim_T$

Theorem

$\sim_T$ satisfies the following properties:

- *$\sim_T$ is an equivalence relation on S,*
- *$\sim_T$ is a bisimulation on T,*
- *$\sim_T$ is the **coarsest** bisimulation for T:*
  if R' is also a bisimulation for T, then $R' \subseteq \sim_T$.

Bisimulation Minimization

Why bisimulation minimization?

Bisimulations can be used to **minimize** a TS.

Key for model checking: complexity typically depends on the size of (LTL / CTL) formula and **size of the model**.

Bisimulation quotient

Let $T = (S, Act, \longrightarrow, I, AP, L)$ be a TS.

Definition

Bisimulation quotient:

Bisimulation quotient

Let $T = (S, Act, \longrightarrow, I, AP, L)$ be a TS.

Definition

Bisimulation quotient: $T/_{\sim} = (S', Act', \longrightarrow', I', AP, L')$

- $S' = S/_{\sim_T}$ (set of bisimulation equivalence classes)

Bisimulation quotient

Let $T = (S, Act, \longrightarrow, I, AP, L)$ be a TS.

Definition

Bisimulation quotient: $T/_\sim = (S', Act', \longrightarrow', I', AP, L')$

- $S' = S/_{\sim_T}$ (set of bisimulation equivalence classes)
- $I' = \{[s]_{\sim_T} \mid s \in I\}$

Bisimulation quotient

Let $T = (S, Act, \longrightarrow, I, AP, L)$ be a TS.

Definition

Bisimulation quotient: $T/_{\sim} = (S', Act', \longrightarrow', I', AP, L')$

- $S' = S/_{\sim_T}$ (set of bisimulation equivalence classes)
- $I' = \{[s]_{\sim_T} \mid s \in I\}$
- $L'([s]_{\sim_T}) = L(s)$

Bisimulation quotient

Let $T = (S, Act, \longrightarrow, I, AP, L)$ be a TS.

Definition

Bisimulation quotient: $T/_{\sim} = (S', Act', \longrightarrow', I', AP, L')$

- $S' = S/_{\sim_T}$ (set of bisimulation equivalence classes)
- $I' = \{[s]_{\sim_T} \mid s \in I\}$
- $L'([s]_{\sim_T}) = L(s)$
- If $s \longrightarrow s'$, then $[s]_{\sim_T} \longrightarrow [s']_{\sim_T}$

Bisimulation quotient

Let $T = (S, Act, \longrightarrow, I, AP, L)$ be a TS.

Definition

Bisimulation quotient: $T/_{\sim} = (S', Act', \longrightarrow', I', AP, L')$

- $S' = S/_{\sim_T}$ (set of bisimulation equivalence classes)
- $I' = \{[s]_{\sim_T} \mid s \in I\}$
- $L'([s]_{\sim_T}) = L(s)$
- If $s \longrightarrow s'$, then $[s]_{\sim_T} \longrightarrow [s']_{\sim_T}$

You can prove $T \sim T/_{\sim}$.

Bisimulation quotient

Let $T = (S, Act, \longrightarrow, I, AP, L)$ be a TS.

Definition

Bisimulation quotient: $T/_{\sim} = (S', Act', \longrightarrow', I', AP, L')$

- $S' = S/_{\sim_T}$ (set of bisimulation equivalence classes)
- $I' = \{[s]_{\sim_T} \mid s \in I\}$
- $L'([s]_{\sim_T}) = L(s)$
- If $s \longrightarrow s'$, then $[s]_{\sim_T} \longrightarrow [s']_{\sim_T}$

You can prove $T \sim T/_{\sim}$.

How? $R = \{(s, [s]_{\sim}) \mid s \in S\}$ is a bisimulation relation.

How to find $T/_{\sim}$

A **partition** splits a set into a number of disjoint subsets.

Definition (Partition)

a partition Π of S is a set $\Pi = \{B_1, \ldots, B_k\}$ with

- For all $1 \leq i \leq k$. $\quad B_i \neq \emptyset$,
- for all $1 \leq i < j \leq k$. $\quad B_i \cap B_j = \emptyset$,
- $S = \cup_{i=1}^{k} B_i$.

How to find $T/_{\sim}$

A **partition** splits a set into a number of disjoint subsets.

Definition (Partition)

a partition Π of S is a set $\Pi = \{B_1, \ldots, B_k\}$ with

- For all $1 \leq i \leq k. \quad B_i \neq \emptyset$,
- for all $1 \leq i < j \leq k. \quad B_i \cap B_j = \emptyset$,
- $S = \cup_{i=1}^{k} B_i$.

A bisimulation R induces a partition $\Pi_R = \{[s]_R \mid s \in S\}$ where $[s]_R = \{s' \in S \mid (s, s') \in R\}$ is the **equivalence class** of s.

A partition Π can be turned into a relation: $R_\Pi = \{(s, s') \mid \exists B \in \Pi . s, s' \in B\}$.

Partition refinement (minimization algorithm)

Start with the initial relation $R_{AP} = \{(s_1, s_2) \in S \times S \mid L(s_1) = L(s_2)\}$.

Partition refinement (minimization algorithm)

Start with the initial relation $R_{AP} = \{(s_1, s_2) \in S \times S \mid L(s_1) = L(s_2)\}$.

Define **initial partition** $\Pi_0 = \Pi_{R_{AP}}$.

Refine Π_i into Π_{i+1} until $\Pi_{i+1} = \Pi_i$:

Partition refinement (minimization algorithm)

Start with the initial relation $R_{AP} = \{(s_1, s_2) \in S \times S \mid L(s_1) = L(s_2)\}$.

Define **initial partition** $\Pi_0 = \Pi_{R_{AP}}$.

Refine Π_i into Π_{i+1} until $\Pi_{i+1} = \Pi_i$:

- For each block $B \in \Pi_i$, check if that block can be split.

Partition refinement (minimization algorithm)

Start with the initial relation $R_{AP} = \{(s_1, s_2) \in S \times S \mid L(s_1) = L(s_2)\}$.

Define **initial partition** $\Pi_0 = \Pi_{R_{AP}}$.

Refine Π_i into Π_{i+1} until $\Pi_{i+1} = \Pi_i$:

- For each block $B \in \Pi_i$, check if that block can be split.
- Pick block $B \in \Pi_i$ with states $s_1, s_2 \in B$:

Partition refinement (minimization algorithm)

Start with the initial relation $R_{AP} = \{(s_1, s_2) \in S \times S \mid L(s_1) = L(s_2)\}$.

Define **initial partition** $\Pi_0 = \Pi_{R_{AP}}$.

Refine Π_i into Π_{i+1} until $\Pi_{i+1} = \Pi_i$:

- For each block $B \in \Pi_i$, check if that block can be split.
- Pick block $B \in \Pi_i$ with states $s_1, s_2 \in B$:
  check if for all s_1' such that $s_1 \longrightarrow s_1'$ and $s_1' \in B_1'$, there exists s_2' such that $s_2 \longrightarrow s_2'$ with $s_2' \in B_1'$,

Partition refinement (minimization algorithm)

Start with the initial relation $R_{AP} = \{(s_1, s_2) \in S \times S \mid L(s_1) = L(s_2)\}$.

Define **initial partition** $\Pi_0 = \Pi_{R_{AP}}$.

Refine Π_i into Π_{i+1} until $\Pi_{i+1} = \Pi_i$:

- For each block $B \in \Pi_i$, check if that block can be split.
- Pick block $B \in \Pi_i$ with states $s_1, s_2 \in B$:
  check if for all s_1' such that $s_1 \longrightarrow s_1'$ and $s_1' \in B_1'$, there exists s_2' such that $s_2 \longrightarrow s_2'$ with $s_2' \in B_1'$,
- Split B into B_1, B_2 with
  $B_1 = \{s \in B \mid s \sim_T s_1\}, B_2 = \{s \in B \mid s \not\sim_T s_1\}$,

Partition refinement (minimization algorithm)

Start with the initial relation $R_{AP} = \{(s_1, s_2) \in S \times S \mid L(s_1) = L(s_2)\}$.

Define **initial partition** $\Pi_0 = \Pi_{R_{AP}}$.

Refine Π_i into Π_{i+1} until $\Pi_{i+1} = \Pi_i$:

- For each block $B \in \Pi_i$, check if that block can be split.
- Pick block $B \in \Pi_i$ with states $s_1, s_2 \in B$:
  check if for all s_1' such that $s_1 \longrightarrow s_1'$ and $s_1' \in B_1'$, there exists s_2' such that $s_2 \longrightarrow s_2'$ with $s_2' \in B_1'$,
- Split B into B_1, B_2 with
  $B_1 = \{s \in B \mid s \sim_T s_1\}, B_2 = \{s \in B \mid s \not\sim_T s_1\}$,
- Π_{i+1} is the partition Π_i where B is replaced by B_1 and B_2,

Partition refinement (minimization algorithm)

Start with the initial relation $R_{AP} = \{(s_1, s_2) \in S \times S \mid L(s_1) = L(s_2)\}$.

Define **initial partition** $\Pi_0 = \Pi_{R_{AP}}$.

Refine Π_i into Π_{i+1} until $\Pi_{i+1} = \Pi_i$:

- For each block $B \in \Pi_i$, check if that block can be split.
- Pick block $B \in \Pi_i$ with states $s_1, s_2 \in B$:
  check if for all s_1' such that $s_1 \longrightarrow s_1'$ and $s_1' \in B_1'$, there exists s_2' such that $s_2 \longrightarrow s_2'$ with $s_2' \in B_1'$,
- Split B into B_1, B_2 with
  $B_1 = \{s \in B \mid s \sim_T s_1\}, B_2 = \{s \in B \mid s \not\sim_T s_1\}$,
- Π_{i+1} is the partition Π_i where B is replaced by B_1 and B_2,
- Repeat until all blocks B cannot be split any further
  (then $\Pi_{i+1} = \Pi_i$).

After this, R_{Π_i} is a bisimulation, and $\sim_T = R_{\Pi_i}$.

Partition refinement (minimization algorithm)

Start with the initial relation $R_{AP} = \{(s_1, s_2) \in S \times S \mid L(s_1) = L(s_2)\}$.

Define **initial partition** $\Pi_0 = \Pi_{R_{AP}}$.

Refine Π_i into Π_{i+1} until $\Pi_{i+1} = \Pi_i$:

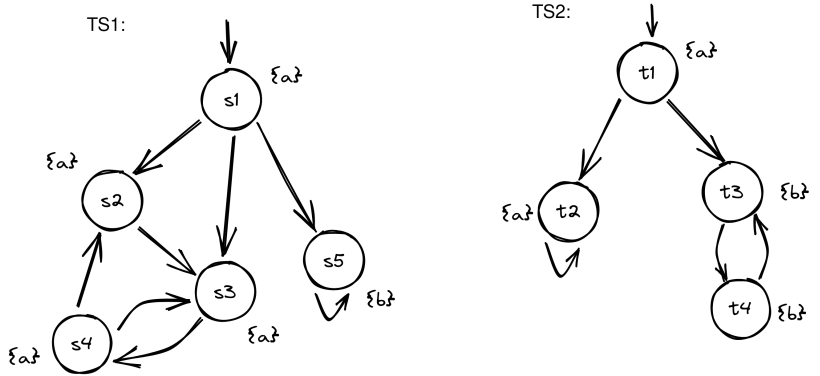
- For each block $B \in \Pi_i$, check if that block can be split.
- Pick block $B \in \Pi_i$ with states $s_1, s_2 \in B$:
  check if for all s_1' such that $s_1 \longrightarrow s_1'$ and $s_1' \in B_1'$, there exists s_2' such that $s_2 \longrightarrow s_2'$ with $s_2' \in B_1'$,
- Split B into B_1, B_2 with
  $B_1 = \{s \in B \mid s \sim_T s_1\}, B_2 = \{s \in B \mid s \not\sim_T s_1\}$,
- Π_{i+1} is the partition Π_i where B is replaced by B_1 and B_2,
- Repeat until all blocks B cannot be split any further
  (then $\Pi_{i+1} = \Pi_i$).

After this, R_{Π_i} is a bisimulation, and $\sim_T = R_{\Pi_i}$.

Note: this procedure looks at successor states of s_1, s_2, also possible to look at predecessors.

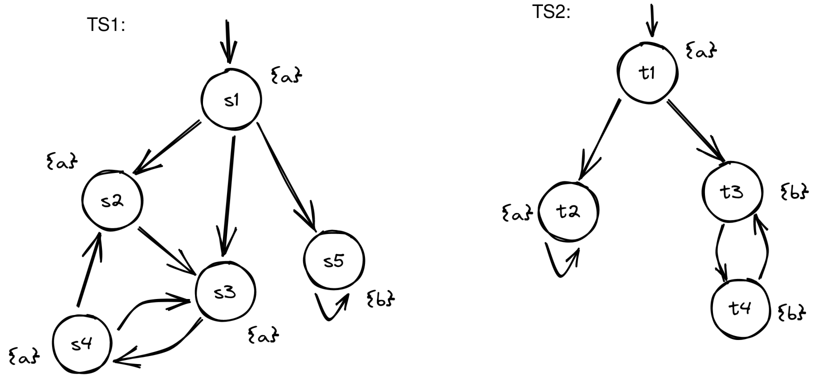
Example 3

Are the following two TS bisimilar?



Example 3

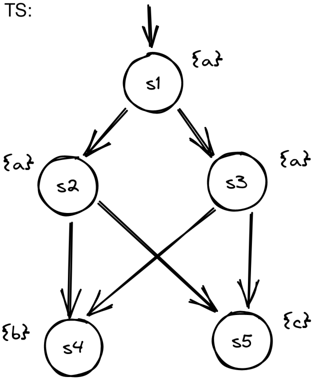
Are the following two TS bisimilar?



Yes. $R = \{(s_1, t_1), (s_2, t_2), (s_3, t_2), (s_4, t_2), (s_5, t_3), (s_5, t_4)\}$.

Example 4

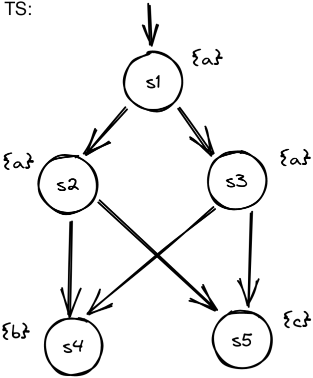
Compute the bisimulation quotient $TS/_{\sim}$ for the following TS:



Example 4

Compute the bisimulation quotient $TS/_{\sim}$ for the following TS:

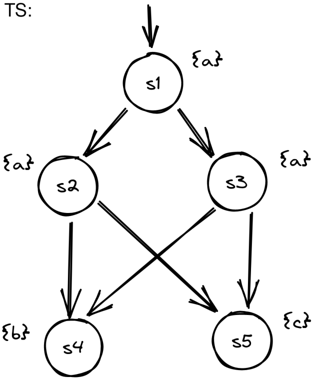
TS:



$$\Pi_0 = \Pi_{AP} = \{\{s_1, s_2, s_3\}, \{s_4\}, \{s_5\}\}.$$

Example 4

Compute the bisimulation quotient $TS/_\sim$ for the following TS:

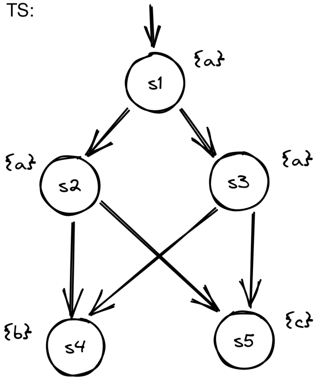


$$\Pi_0 = \Pi_{AP} = \{\{s_1, s_2, s_3\}, \{s_4\}, \{s_5\}\}.$$

Π_1: Pick $B = \{s_1, s_2, s_3\}$,

Example 4

Compute the bisimulation quotient $TS/_\sim$ for the following TS:



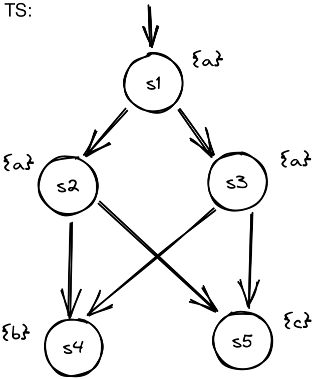
$$\Pi_0 = \Pi_{AP} = \{\{s_1, s_2, s_3\}, \{s_4\}, \{s_5\}\}.$$

Π_1: Pick $B = \{s_1, s_2, s_3\}$,
we have $s_1 \not\sim s_2$, so we split:

$$\Pi_1 = \{\{s_1\}, \{s_2, s_3\}, \{s_4\}, \{s_5\}\}.$$

Example 4

Compute the bisimulation quotient $TS/_\sim$ for the following TS:



$$\Pi_0 = \Pi_{AP} = \{\{s_1, s_2, s_3\}, \{s_4\}, \{s_5\}\}.$$

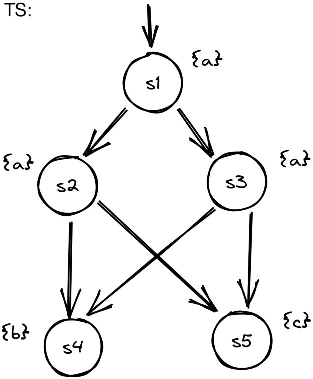
Π_1: Pick $B = \{s_1, s_2, s_3\}$,
we have $s_1 \not\sim s_2$, so we split:

$$\Pi_1 = \{\{s_1\}, \{s_2, s_3\}, \{s_4\}, \{s_5\}\}.$$

Π_2: Pick $B = \{s_2, s_3\}$,

Example 4

Compute the bisimulation quotient $TS/_{\sim}$ for the following TS:



$$\Pi_0 = \Pi_{AP} = \{\{s_1, s_2, s_3\}, \{s_4\}, \{s_5\}\}.$$

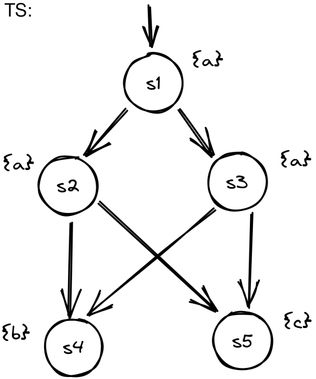
Π_1: Pick $B = \{s_1, s_2, s_3\}$,
we have $s_1 \not\sim s_2$, so we split:

$$\Pi_1 = \{\{s_1\}, \{s_2, s_3\}, \{s_4\}, \{s_5\}\}.$$

Π_2: Pick $B = \{s_2, s_3\}$,
we have $s_2 \sim s_3$, so $\Pi_2 = \Pi_1$.

Example 4

Compute the bisimulation quotient $TS/_\sim$ for the following TS:



$$\Pi_0 = \Pi_{AP} = \{\{s_1, s_2, s_3\}, \{s_4\}, \{s_5\}\}.$$

Π_1: Pick $B = \{s_1, s_2, s_3\}$,
we have $s_1 \not\sim s_2$, so we split:

$$\Pi_1 = \{\{s_1\}, \{s_2, s_3\}, \{s_4\}, \{s_5\}\}.$$

Π_2: Pick $B = \{s_2, s_3\}$,
we have $s_2 \sim s_3$, so $\Pi_2 = \Pi_1$.
No block can be refined any further, done.

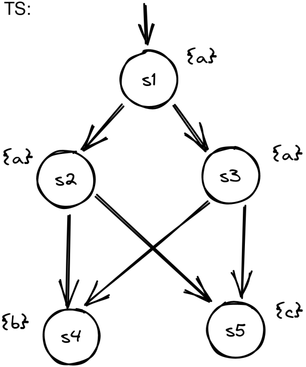
$$\sim = R_{\Pi_2} = \{(s_i, s_i), (s_2, s_3)\}.$$

Example 4, continued

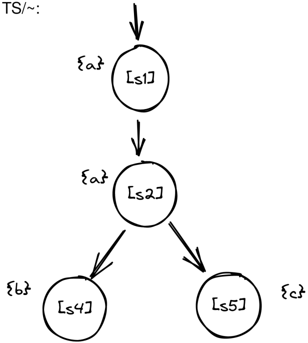
Compute the bisimulation quotient $TS/_{\sim}$ for the following TS:

$$\sim\; = R_{\Pi_2} = \{(s_i, s_i), (s_2, s_3)\}.$$

TS:

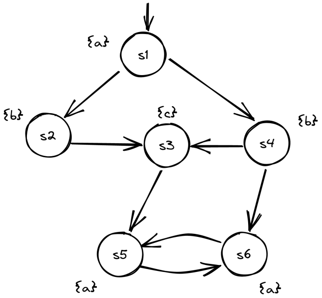


TS/~:



Example 5

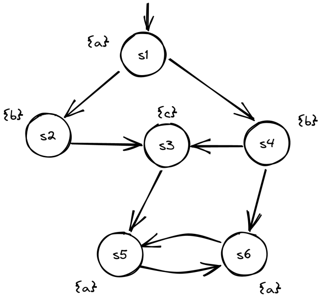
Compute the bisimulation quotient $TS/_\sim$ for the following TS:



Example 5

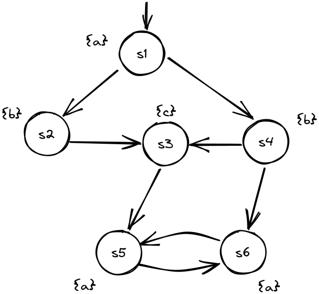
Compute the bisimulation quotient $TS/_{\sim}$ for the following TS:

$$\Pi_0 = \Pi_{AP} = \{\{s_1, s_5, s_6\}, \{s_2, s_4\}, \{s_3\}\}.$$



Example 5

Compute the bisimulation quotient $TS/_{\sim}$ for the following TS:

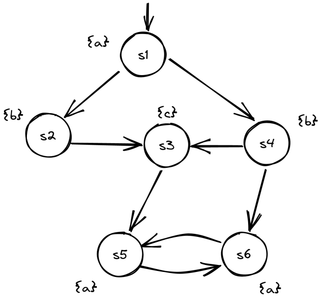


$$\Pi_0 = \Pi_{AP} = \{\{s_1, s_5, s_6\}, \{s_2, s_4\}, \{s_3\}\}.$$

Π_1: Pick $B = \{s_1, s_5, s_6\}$,

Example 5

Compute the bisimulation quotient $TS/_\sim$ for the following TS:



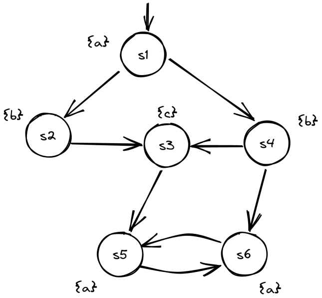
$$\Pi_0 = \Pi_{AP} = \{\{s_1, s_5, s_6\}, \{s_2, s_4\}, \{s_3\}\}.$$

Π_1: Pick $B = \{s_1, s_5, s_6\}$,
we have $s_1 \not\sim s_5$, so we split:

$$\Pi_1 = \{\{s_1\}, \{s_5, s_6\}, \{s_2, s_4\}, \{s_3\}\}.$$

Example 5

Compute the bisimulation quotient $TS/_{\sim}$ for the following TS:



$$\Pi_0 = \Pi_{AP} = \{\{s_1, s_5, s_6\}, \{s_2, s_4\}, \{s_3\}\}.$$

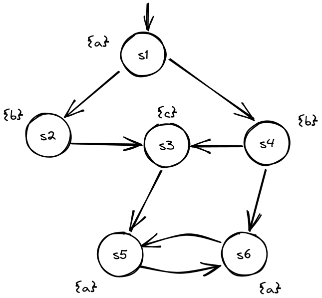
Π_1: Pick $B = \{s_1, s_5, s_6\}$,
we have $s_1 \not\sim s_5$, so we split:

$$\Pi_1 = \{\{s_1\}, \{s_5, s_6\}, \{s_2, s_4\}, \{s_3\}\}.$$

Π_2: Pick $B = \{s_5, s_6\}$, $s_5 \sim s_6$, next block.

Example 5

Compute the bisimulation quotient $TS/_{\sim}$ for the following TS:



$$\Pi_0 = \Pi_{AP} = \{\{s_1, s_5, s_6\}, \{s_2, s_4\}, \{s_3\}\}.$$

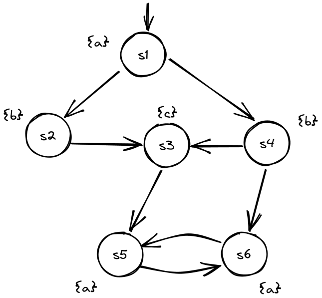
Π_1: Pick $B = \{s_1, s_5, s_6\}$,
we have $s_1 \not\sim s_5$, so we split:

$$\Pi_1 = \{\{s_1\}, \{s_5, s_6\}, \{s_2, s_4\}, \{s_3\}\}.$$

Π_2: Pick $B = \{s_5, s_6\}$, $s_5 \sim s_6$, next block.
Π_2: Pick $B = \{s_2, s_4\}$,

Example 5

Compute the bisimulation quotient $TS/_{\sim}$ for the following TS:



$$\Pi_0 = \Pi_{AP} = \{\{s_1, s_5, s_6\}, \{s_2, s_4\}, \{s_3\}\}.$$

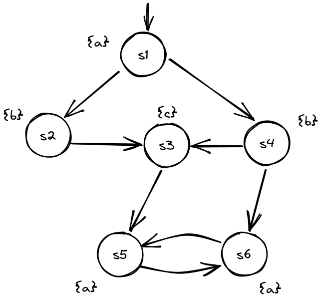
Π_1: Pick $B = \{s_1, s_5, s_6\}$,
we have $s_1 \not\sim s_5$, so we split:

$$\Pi_1 = \{\{s_1\}, \{s_5, s_6\}, \{s_2, s_4\}, \{s_3\}\}.$$

Π_2: Pick $B = \{s_5, s_6\}$, $s_5 \sim s_6$, next block.
Π_2: Pick $B = \{s_2, s_4\}$,
we have $s_2 \not\sim s_4$, so we split:

Example 5

Compute the bisimulation quotient $TS/_\sim$ for the following TS:



$$\Pi_0 = \Pi_{AP} = \{\{s_1, s_5, s_6\}, \{s_2, s_4\}, \{s_3\}\}.$$

Π_1: Pick $B = \{s_1, s_5, s_6\}$,
we have $s_1 \not\sim s_5$, so we split:

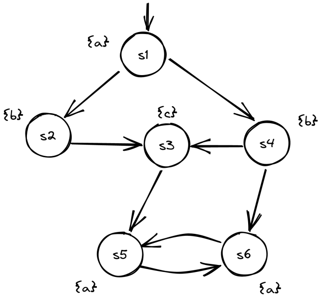
$$\Pi_1 = \{\{s_1\}, \{s_5, s_6\}, \{s_2, s_4\}, \{s_3\}\}.$$

Π_2: Pick $B = \{s_5, s_6\}$, $s_5 \sim s_6$, next block.
Π_2: Pick $B = \{s_2, s_4\}$,
we have $s_2 \not\sim s_4$, so we split:

$$\Pi_2 = \{\{s_1\}, \{s_5, s_6\}, \{s_2\}, \{s_4\}, \{s_3\}\}.$$

Example 5

Compute the bisimulation quotient $TS/_\sim$ for the following TS:



$$\Pi_0 = \Pi_{AP} = \{\{s_1, s_5, s_6\}, \{s_2, s_4\}, \{s_3\}\}.$$

Π_1: Pick $B = \{s_1, s_5, s_6\}$,
we have $s_1 \not\sim s_5$, so we split:

$$\Pi_1 = \{\{s_1\}, \{s_5, s_6\}, \{s_2, s_4\}, \{s_3\}\}.$$

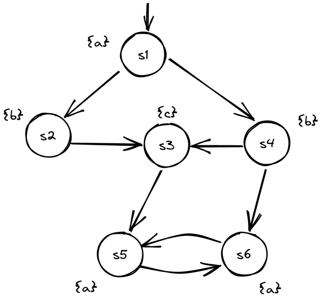
Π_2: Pick $B = \{s_5, s_6\}$, $s_5 \sim s_6$, next block.
Π_2: Pick $B = \{s_2, s_4\}$,
we have $s_2 \not\sim s_4$, so we split:

$$\Pi_2 = \{\{s_1\}, \{s_5, s_6\}, \{s_2\}, \{s_4\}, \{s_3\}\}.$$

Π_3: Pick $B = \{s_5, s_6\}$, we have $s_5 \sim s_6$.

Example 5

Compute the bisimulation quotient $TS/_\sim$ for the following TS:



$$\Pi_0 = \Pi_{AP} = \{\{s_1, s_5, s_6\}, \{s_2, s_4\}, \{s_3\}\}.$$

Π_1: Pick $B = \{s_1, s_5, s_6\}$,
we have $s_1 \not\sim s_5$, so we split:

$$\Pi_1 = \{\{s_1\}, \{s_5, s_6\}, \{s_2, s_4\}, \{s_3\}\}.$$

Π_2: Pick $B = \{s_5, s_6\}$, $s_5 \sim s_6$, next block.
Π_2: Pick $B = \{s_2, s_4\}$,
we have $s_2 \not\sim s_4$, so we split:

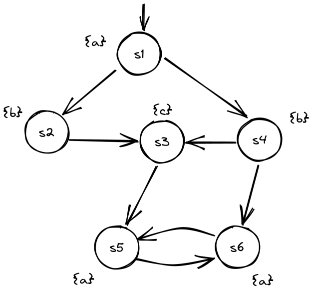
$$\Pi_2 = \{\{s_1\}, \{s_5, s_6\}, \{s_2\}, \{s_4\}, \{s_3\}\}.$$

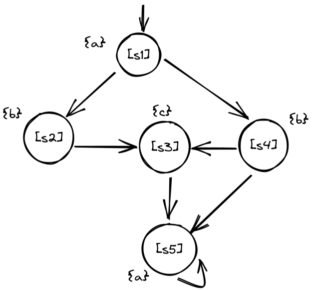
Π_3: Pick $B = \{s_5, s_6\}$, we have $s_5 \sim s_6$.
No block can be refined any further, done.

Example 5, continued

Compute the bisimulation quotient $TS/_\sim$ for the following TS:

$$R_{\Pi_3} = \{(s_i, s_i), (s_5, s_6)\}.$$





Bisimulations and CTL/CTL*-equivalence

CTL*

CTL* is a logic that extends CTL and contains both CTL and LTL.

CTL*

CTL* is a logic that extends CTL and contains both CTL and LTL.

CTL* allows arbitrary nesting of temporal logic operators: $\exists\Diamond\Box a$ is a CTL* formula, but not a CTL formula.

CTL*

CTL* is a logic that extends CTL and contains both CTL and LTL.

CTL* allows arbitrary nesting of temporal logic operators: $\exists\Diamond\Box a$ is a CTL* formula, but not a CTL formula.

CTL* contains LTL: $\Phi \in LTL \implies \forall\Phi \in CTL^*$.

CTL*

CTL* is a logic that extends CTL and contains both CTL and LTL.

CTL* allows arbitrary nesting of temporal logic operators: $\exists\Diamond\Box a$ is a CTL* formula, but not a CTL formula.

CTL* contains LTL: $\Phi \in LTL \implies \forall\Phi \in CTL^*$.

Grammar (in existential normal form):

$$\Phi ::= \text{true} \mid a \mid \Phi \wedge \Phi \mid \neg\Phi \mid \exists\varphi \qquad \text{(State formulae)}$$

$$\varphi ::= \Phi \mid \varphi \wedge \varphi \mid \neg\varphi \mid \bigcirc\varphi \mid \varphi \mathsf{U} \varphi \qquad \text{(Path formulae)}$$

CTL/CTL*-bisimulation equivalence

Theorem (CTL/CTL* and bisimulation equivalence)

For a finitely branching transition system TS, we have

$$\sim_{TS} \quad = \quad \equiv_{CTL} \quad = \quad \equiv_{CTL^*}$$

Proof.

Split into three lemmas. $\square$

Lemma 1

Lemma (CTL equivalence is finer than bisimulation)

For a finitely branching transition system TS, we have

$$s_1 \equiv_{CTL} s_2 \quad \Longrightarrow \quad s_1 \sim_{TS} s_2.$$

I.e. $\equiv_{CTL} \quad \subseteq \quad \sim_{TS}$.

Proof.

$R = \{(s_1, s_2) \in S \times S \mid s_1 \equiv_{CTL} s_2\}$ is a bisimulation for TS. □

Detailed proof

$R = \{(s_1, s_2) \in S \times S \mid s_1 \equiv_{CTL} s_2\}.$

Detailed proof

$R = \{(s_1, s_2) \in S \times S \mid s_1 \equiv_{CTL} s_2\}$.

$\forall a \in AP.\ s_1 \models a \iff s_2 \models a$, thus $a \in L(s_1)$ if and only if $a \in L(s_2)$.

Detailed proof

$R = \{(s_1, s_2) \in S \times S \mid s_1 \equiv_{CTL} s_2\}$.

$\forall a \in AP.\ s_1 \models a \iff s_2 \models a$, thus $a \in L(s_1)$ if and only if $a \in L(s_2)$.

Suppose $(s_1, s_2) \in R$, let s_1' be such that $s_1 \longrightarrow s_1'$,
and assume for all s_2' with $s_2 \longrightarrow s_2'$. $(s_1', s_2') \notin R$.

Detailed proof

$R = \{(s_1, s_2) \in S \times S \mid s_1 \equiv_{CTL} s_2\}$.

$\forall a \in AP.\ s_1 \models a \iff s_2 \models a$, thus $a \in L(s_1)$ if and only if $a \in L(s_2)$.

Suppose $(s_1, s_2) \in R$, let s_1' be such that $s_1 \longrightarrow s_1'$,
and assume for all s_2' with $s_2 \longrightarrow s_2'$. $(s_1', s_2') \notin R$.

TS is finitely branching:

Detailed proof

$R = \{(s_1, s_2) \in S \times S \mid s_1 \equiv_{CTL} s_2\}$.

$\forall a \in AP.\ s_1 \models a \iff s_2 \models a$, thus $a \in L(s_1)$ if and only if $a \in L(s_2)$.

Suppose $(s_1, s_2) \in R$, let s_1' be such that $s_1 \longrightarrow s_1'$,
and assume for all s_2' with $s_2 \longrightarrow s_2'$. $(s_1', s_2') \notin R$.

TS is finitely branching: finitely many s_2', and for each exists a formula $\Psi_{s_2'}$ such that

$$s_1' \models \Psi_{s_2'} \quad \text{and} \quad s_2' \not\models \Psi_{s_2'}.$$

Detailed proof

$R = \{(s_1, s_2) \in S \times S \mid s_1 \equiv_{CTL} s_2\}$.

$\forall a \in AP.\ s_1 \models a \iff s_2 \models a$, thus $a \in L(s_1)$ if and only if $a \in L(s_2)$.

Suppose $(s_1, s_2) \in R$, let s_1' be such that $s_1 \longrightarrow s_1'$,
and assume for all s_2' with $s_2 \longrightarrow s_2'$. $(s_1', s_2') \notin R$.

TS is finitely branching: finitely many s_2', and for each exists a formula $\Psi_{s_2'}$ such that

$$s_1' \models \Psi_{s_2'} \quad \text{and} \quad s_2' \not\models \Psi_{s_2'}.$$

Let $\Phi = \exists \bigcirc \left(\bigwedge_{s_2''} \Psi_{s_2''}\right)$.

Detailed proof

$R = \{(s_1, s_2) \in S \times S \mid s_1 \equiv_{CTL} s_2\}$.

$\forall a \in AP.\ s_1 \models a \iff s_2 \models a$, thus $a \in L(s_1)$ if and only if $a \in L(s_2)$.

Suppose $(s_1, s_2) \in R$, let s_1' be such that $s_1 \longrightarrow s_1'$,
and assume for all s_2' with $s_2 \longrightarrow s_2'$. $(s_1', s_2') \notin R$.

TS is finitely branching: finitely many s_2', and for each exists a formula $\Psi_{s_2'}$ such that

$$s_1' \models \Psi_{s_2'} \quad \text{and} \quad s_2' \not\models \Psi_{s_2'}.$$

Let $\Phi = \exists \bigcirc \left(\bigwedge_{s_2''} \Psi_{s_2''}\right)$.

Then clearly $s_1' \models \bigwedge_{s_2''} \Psi_{s_2''}$ and $s_1 \models \Phi$, and $s_2' \not\models \bigwedge_{s_2''} \Psi_{s_2''}$ and thus $s_2 \not\models \Phi$.

Detailed proof

$R = \{(s_1, s_2) \in S \times S \mid s_1 \equiv_{CTL} s_2\}$.

$\forall a \in AP.\ s_1 \models a \iff s_2 \models a$, thus $a \in L(s_1)$ if and only if $a \in L(s_2)$.

Suppose $(s_1, s_2) \in R$, let s_1' be such that $s_1 \longrightarrow s_1'$,
and assume for all s_2' with $s_2 \longrightarrow s_2'$. $(s_1', s_2') \notin R$.

TS is finitely branching: finitely many s_2', and for each exists a formula $\Psi_{s_2'}$ such that

$$s_1' \models \Psi_{s_2'} \quad \text{and} \quad s_2' \not\models \Psi_{s_2'}.$$

Let $\Phi = \exists \bigcirc (\bigwedge_{s_2''} \Psi_{s_2''})$.

Then clearly $s_1' \models \bigwedge_{s_2''} \Psi_{s_2''}$ and $s_1 \models \Phi$, and $s_2' \not\models \bigwedge_{s_2''} \Psi_{s_2''}$ and thus $s_2 \not\models \Phi$.

A contradiction with $(s_1, s_2) \in R$. $\square$

Lemma 2

Lemma (Bisimulation is finer than CTL* equivalence)

For any transition system TS, we have for states $s_1, s_2 \in S$, and for paths π_1, π_2:

1. *If $s_1 \sim_{TS} s_2$, then for any CTL* state formula $\Phi : s_1 \models \Phi \iff s_2 \models \Phi$*
2. *If $\pi_1 \sim_{TS} \pi_2$, then for any CTL* path formula $\varphi : \pi_1 \models \varphi \iff \pi_2 \models \varphi$.*

That is, $\sim_{TS} \quad \subseteq \quad \equiv_{CTL^*}$.

Proof.

By induction over the structure of the formula. □

Detailed proof

Base case: Let $s_1 \sim_{TS} s_2$, then $L(s_1) = L(s_2)$ and thus for $\Phi = a \in AP$ we have

$$s_1 \models a \iff a \in L(s_1) \iff a \in L(s_2) \iff s_2 \models a.$$

Detailed proof

Base case: Let $s_1 \sim_{TS} s_2$, then $L(s_1) = L(s_2)$ and thus for $\Phi = a \in AP$ we have

$$s_1 \models a \iff a \in L(s_1) \iff a \in L(s_2) \iff s_2 \models a.$$

Induction step (state formulae): Assume Φ_1, Φ_2, Ψ are CTL* state formulae for which (1) holds, and φ a CTL* path formula for which (2) holds.

Let $s_1 \sim_{TS} s_2$, and do structural induction on Φ.

Detailed proof

Case 1: $\Phi = \Phi_1 \wedge \Phi_2$.

$$
\begin{aligned}
s_1 \models \Phi_1 \wedge \Phi_2 &\iff s_1 \models \Phi_1 \text{ and } s_1 \models \Phi_2 \\
&\iff s_2 \models \Phi_1 \text{ and } s_2 \models \Phi_2 && \text{(by I.H.)} \\
&\iff s_2 \models \Phi_1 \wedge \Phi_2.
\end{aligned}
$$

Detailed proof

Case 1: $\Phi = \Phi_1 \wedge \Phi_2$.

$$
\begin{aligned}
s_1 \models \Phi_1 \wedge \Phi_2 &\iff s_1 \models \Phi_1 \text{ and } s_1 \models \Phi_2 \\
&\iff s_2 \models \Phi_1 \text{ and } s_2 \models \Phi_2 && \text{(by I.H.)} \\
&\iff s_2 \models \Phi_1 \wedge \Phi_2.
\end{aligned}
$$

Case 2: $\Phi = \neg\Psi$.

$$
\begin{aligned}
s_1 \models \neg\Psi &\iff s_1 \not\models \Psi \\
&\iff s_2 \not\models \Psi && \text{(by I.H.)} \\
&\iff s_2 \models \neg\Psi.
\end{aligned}
$$

Detailed proof

Case 3: $\Phi = \exists\varphi$.
It suffices to show $s_1 \models \exists\varphi \implies s_2 \models \exists\varphi$, the other direction will hold by symmetry.

Detailed proof

Case 3: $\Phi = \exists\varphi$.
It suffices to show $s_1 \models \exists\varphi \implies s_2 \models \exists\varphi$, the other direction will hold by symmetry.

Assume $s_1 \models \varphi$, then there exists a path $\pi_1 \in \mathit{Paths}(s_1)$ with $\pi_1 \models \varphi$.

Detailed proof

Case 3: $\Phi = \exists\varphi$.
It suffices to show $s_1 \models \exists\varphi \implies s_2 \models \exists\varphi$, the other direction will hold by symmetry.

Assume $s_1 \models \varphi$, then there exists a path $\pi_1 \in \mathit{Paths}(s_1)$ with $\pi_1 \models \varphi$.

Apply the **path lifting lemma**: then there also exists a path $\pi_2 \in \mathit{Paths}(s_2)$ such that $\pi_1 \sim_{TS} \pi_2$.

Detailed proof

Case 3: $\Phi = \exists\varphi$.
It suffices to show $s_1 \models \exists\varphi \implies s_2 \models \exists\varphi$, the other direction will hold by symmetry.

Assume $s_1 \models \varphi$, then there exists a path $\pi_1 \in \mathit{Paths}(s_1)$ with $\pi_1 \models \varphi$.

Apply the **path lifting lemma**: then there also exists a path $\pi_2 \in \mathit{Paths}(s_2)$ such that $\pi_1 \sim_{TS} \pi_2$.

From the I.H. it follows that $\pi_2 \models \varphi$ and thus $s_2 \models \exists\varphi$.

Detailed proof

Induction step (path formulae): Assume Φ is a CTL* state formula for which (1) holds, and $\varphi_1, \varphi_2, \psi$ are CTL* path formulae for which (2) holds.

Let $\pi_1 \sim_{TS} \pi_2$, and do structural induction on φ.

Detailed proof

Induction step (path formulae): Assume Φ is a CTL* state formula for which (1) holds, and $\varphi_1, \varphi_2, \psi$ are CTL* path formulae for which (2) holds.

Let $\pi_1 \sim_{TS} \pi_2$, and do structural induction on φ.

Case 1: $\varphi = \Phi$. We have that

$$\pi_1 \models \varphi \iff \pi_1[0] \models \Phi \iff \pi_2[0] \models \Phi \iff \pi_2 \models \varphi.$$

Detailed proof

Induction step (path formulae): Assume Φ is a CTL* state formula for which (1) holds, and $\varphi_1, \varphi_2, \psi$ are CTL* path formulae for which (2) holds.

Let $\pi_1 \sim_{TS} \pi_2$, and do structural induction on φ.

Case 1: $\varphi = \Phi$. We have that

$$\pi_1 \models \varphi \iff \pi_1[0] \models \Phi \iff \pi_2[0] \models \Phi \iff \pi_2 \models \varphi.$$

Cases 2 and 3: The cases where $\varphi = \varphi_1 \wedge \varphi_2$ and $\varphi = \neg\psi$ are similar to cases 2 and 3 of state formulae.

Detailed proof

Case 2: $\varphi = \varphi_1 \wedge \varphi_2$.

$$
\begin{aligned}
\pi_1 \models \varphi_1 \wedge \varphi_2 &\iff \pi_1 \models \varphi_1 \text{ and } \pi_1 \models \varphi_2 \\
&\iff \pi_2 \models \varphi_1 \text{ and } \pi_2 \models \varphi_2 && \text{(by I.H.)} \\
&\iff \pi_2 \models \varphi_1 \wedge \varphi_2.
\end{aligned}
$$

Detailed proof

Case 2: $\varphi = \varphi_1 \wedge \varphi_2$.

$$\begin{aligned}
\pi_1 \models \varphi_1 \wedge \varphi_2 &\iff \pi_1 \models \varphi_1 \text{ and } \pi_1 \models \varphi_2 \\
&\iff \pi_2 \models \varphi_1 \text{ and } \pi_2 \models \varphi_2 && \text{(by I.H.)} \\
&\iff \pi_2 \models \varphi_1 \wedge \varphi_2.
\end{aligned}$$

Case 3: $\varphi = \neg\psi$.

$$\begin{aligned}
\pi_1 \models \neg\psi &\iff \pi_1 \not\models \psi \\
&\iff \pi_2 \not\models \psi && \text{(by I.H.)} \\
&\iff \pi_2 \models \neg\psi.
\end{aligned}$$

Detailed proof

Case 4: $\varphi = \bigcirc\psi$:

$$\begin{aligned}
\pi_1 \models \bigcirc\psi &\iff \pi_1[1\ldots] \models \psi \\
&\iff \pi_2[1\ldots] \models \psi && \text{(by I.H.)} \\
&\iff \pi_2 \models \bigcirc\psi
\end{aligned}$$

Detailed proof

Case 4: $\varphi = \bigcirc\psi$:

$$\begin{aligned} \pi_1 \models \bigcirc\psi &\iff \pi_1[1\ldots] \models \psi \\ &\iff \pi_2[1\ldots] \models \psi && \text{(by I.H.)} \\ &\iff \pi_2 \models \bigcirc\psi \end{aligned}$$

Case 5: $\varphi = \varphi_1 \mathsf{U} \varphi_2$.

$$\begin{aligned} \pi_1 \models \varphi_1 \mathsf{U} \varphi_2 \quad &\iff \exists j \in \mathbb{N}.\quad \pi_1[j\ldots] \models \varphi_2 \quad \text{and} \\ &\qquad\qquad\quad \pi_1[i\ldots] \models \varphi_1, \quad i = 0, \ldots, j-1. \\ &\iff \exists j \in \mathbb{N}.\quad \pi_2[j\ldots] \models \varphi_2 \quad \text{and} \\ &\qquad\qquad\quad \pi_2[i\ldots] \models \varphi_1, \quad i = 0, \ldots, j-1. \\ &\iff \pi_2 \models \varphi_1 \mathsf{U} \varphi_2. \end{aligned}$$

$\square$

Lemma 3

Lemma (CTL* subsumes CTL)

We have $\equiv_{CTL^*} \;\subseteq\; \equiv_{CTL}$.

Proof.

Follows from that every CTL formula Φ is also a CTL* formula. $\square$

Back to our Theorem

Theorem (CTL / CTL* and bisimulation equivalence)

For a finitely branching transition system TS, we have

$$\sim_{TS} \quad = \quad \equiv_{CTL} \quad = \quad \equiv_{CTL^*}$$

Proof.

Applying the lemmas yields

$$\sim_{TS} \quad \subseteq \quad \equiv_{CTL^*} \quad \subseteq \quad \equiv_{CTL} \quad \subseteq \quad \sim_{TS} \; .$$

$\square$

The Theorem in practice

Why is this Theorem useful in practice?

1. If we know two states / TS are bisimilar, they satisfy the same CTL / CTL* formulae,
2. If we find a CTL / CTL* formula that holds in one state / TS but not in the other, they are **not bisimilar**.

Parallel Composition of Transition Systems

Parallel composition - intuition

Real-world systems are big and complex, hard to model as a TS.

Parallel composition - intuition

Real-world systems are big and complex, hard to model as a TS.

But they often consist of smaller components.

Parallel composition - intuition

Real-world systems are big and complex, hard to model as a TS.

But they often consist of smaller components.

Idea: model each component as TS, and combine.

Parallel composition - intuition

Real-world systems are big and complex, hard to model as a TS.

But they often consist of smaller components.

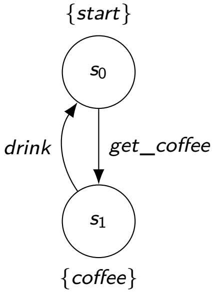
Idea: model each component as TS, and combine.

How to combine? **Parallel composition**.

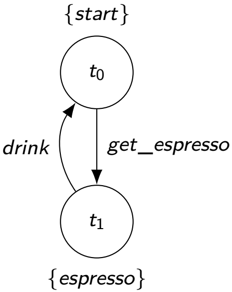
Let's try this intuitively

How would you combine a TS for coffee and a TS for espresso?

TS for coffee.



TS for espresso.



Notations

Structural operational semantics notation:

$$\frac{\text{premise}}{\text{conclusion}}$$

means "if premise is true, then conclusion is true."

Notations

Structural operational semantics notation:

$$\frac{\text{premise}}{\text{conclusion}}$$

means "if premise is true, then conclusion is true."

In the following, we assume transition systems TS_1, TS_2 with $TS_i = (S_i, Act_i, \longrightarrow_i, I_i, AP_i, L_i)$.

Handshaking

Definition (Handshake actions)

We define the set H of handshake actions for TS_1 and TS_2 as the set of all actions that occur in both systems (except τ):

$$H = (Act_1 \cap Act_2) \setminus \{\tau\}.$$

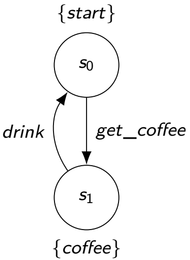
Handshaking

Definition (Handshake actions)

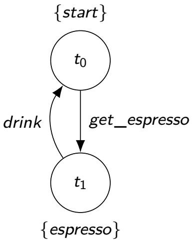
We define the set H of handshake actions for TS_1 and TS_2 as the set of all actions that occur in both systems (except τ):

$$H = (Act_1 \cap Act_2) \setminus \{\tau\}.$$

TS for coffee.



TS for espresso.



$$\begin{aligned} H &= (Act_1 \cap Act_2) \setminus \{\tau\} \\ &= (\{drink, get\_coffee\} \cap \\ &\qquad \{drink, get\_espresso\}) \setminus \{\tau\} \\ &= \{drink\}. \end{aligned}$$

Parallel composition

Definition (Parallel composition)

The parallel composition $TS_1 \parallel_H TS_2$ is defined as
$TS_1 \parallel_H TS_2 = (S_1 \times S_2, Act_1 \cup Act_2, \longrightarrow, I_1 \times I_2, AP_1 \cup AP_2, L)$
where

- $L(\langle s_1, s_2 \rangle) = L_1(s_1) \cup L_2(s_2)$,
- $\longrightarrow$ is defined via the **rules for handshaking actions** H.

Rules for handshaking

Definition (Rules for handshaking)

- ('Interleaving') For actions $\alpha \notin H$:

$$\frac{s_1 \xrightarrow{\alpha}_1 s_1'}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s_1', s_2 \rangle} \quad \text{and} \quad \frac{s_2 \xrightarrow{\alpha}_2 s_2'}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s_1, s_2' \rangle}$$

Rules for handshaking

Definition (Rules for handshaking)

- ('Interleaving') For actions $\alpha \notin H$:

$$\frac{s_1 \xrightarrow{\alpha}_1 s_1'}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s_1', s_2 \rangle} \quad \text{and} \quad \frac{s_2 \xrightarrow{\alpha}_2 s_2'}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s_1, s_2' \rangle}$$

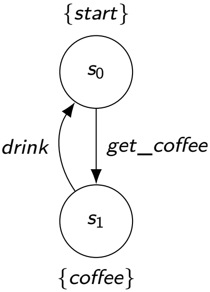
- ('Handshaking') For actions $\alpha \in H$:

$$\frac{s_1 \xrightarrow{\alpha}_1 s_1' \ \wedge \ s_2 \xrightarrow{\alpha}_2 s_2'}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s_1', s_2' \rangle}$$

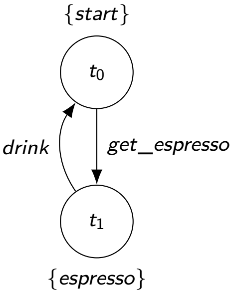
Example

Give the parallel composition $TS_1 \parallel_H TS_2$ for the TS given by:

TS for coffee.



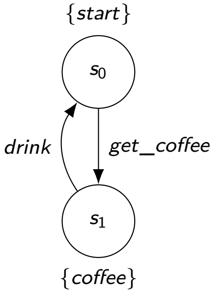
TS for espresso.



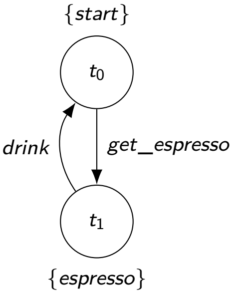
Example

Give the parallel composition $TS_1 \parallel_H TS_2$ for the TS given by:

TS for coffee.

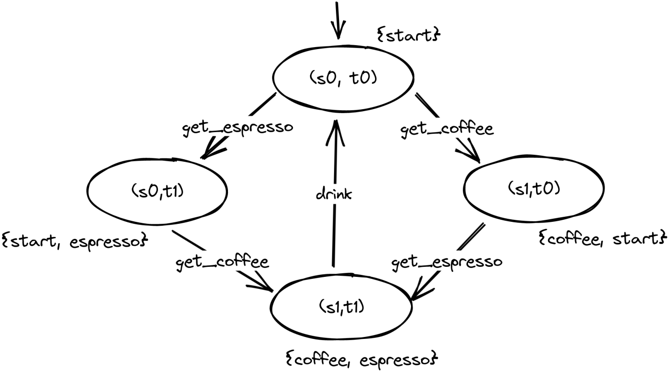


TS for espresso.

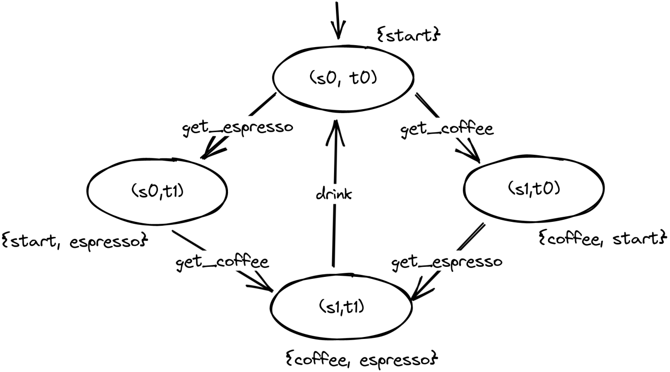


Handshaking actions: $H = \{drink\}$.

Example

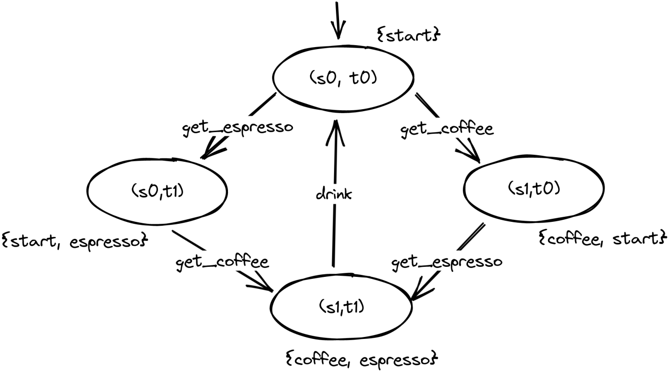

{start}
(s0, t0)
get_espresso
get_coffee
(s0,t1)
drink
(s1,t0)
{start, espresso}
{coffee, start}
get_coffee
get_espresso
(s1,t1)
{coffee, espresso}

Example



We simply followed the rules to compute $TS_1 \parallel_{\{drink\}} TS_2$,
but is this what we want?

Example



We simply followed the rules to compute $TS_1 \parallel_{\{drink\}} TS_2$,
but is this what we want?

We don't want to handshake over *drink*.
Solution: rename *drink* to two actions *drink_coffee* and *drink_espresso*.

Properties

Parallel composition can also be used on more than two transition systems.

In general, we have that

- $TS_1 \parallel_H TS_2 = TS_2 \parallel_H TS_1$,
- $TS_1 \parallel_H (TS_2 \parallel_{H'} TS_3) \neq (TS_1 \parallel_H TS_2) \parallel_{H'} TS_3$ for $H \neq H'$.

Compatibility handshaking and bisimulation

Lemma

For transition system T_1, T_1' *over* Act *and* T_2, T_2' *over* Act', *and* $H \subseteq Act \cup Act'$, *it holds that:*

$$T_1 \sim_a T_1' \text{ and } T_2 \sim_a T_2' \Rightarrow T_1 \parallel_H T_2 \sim_a T_1' \parallel_H T_2'$$

Compatibility handshaking and bisimulation

Lemma

For transition system T_1, T_1' over Act and T_2, T_2' over Act', and $H \subseteq Act \cup Act'$, it holds that:

$$T_1 \sim_a T_1' \text{ and } T_2 \sim_a T_2' \Rightarrow T_1 \parallel_H T_2 \sim_a T_1' \parallel_H T_2'$$

An example of a consequence:

$$(A \parallel_H B \parallel_H C)/_\sim \sim ((A \parallel_H B)/_\sim \parallel_H C)/_\sim$$

Final remarks

Other definitions of bisimulation

For model checking, only looking at the labels is a sufficient notion of behavioral equivalence.

Other definitions of bisimulation

For model checking, only looking at the labels is a sufficient notion of behavioral equivalence.

Other types of bisimilarity can easily be defined, for example action-based bisimilarity:

Other definitions of bisimulation

For model checking, only looking at the labels is a sufficient notion of behavioral equivalence.

Other types of bisimilarity can easily be defined, for example action-based bisimilarity:

Definition (Action-based bisimilarity)

Let $TS_i = (S_i, Act, \longrightarrow_i, AP_i, L_i)$ for $i = 1, 2$ be transition systems with a shared set of actions Act. An action-based bisimulation for (TS_1, TS_2) is a relation $R \subseteq S_1 \times S_2$ with:

1. $\forall s_1 \in I_1. \exists s_2 \in I_2. (s_1, s_2) \in R$, and $\forall s_2 \in I_2. \exists s_1 \in I_1. (s_1, s_2) \in R$.
2. for all $(s_1, s_2) \in R$ we have:
   - If $s_1 \xrightarrow{\alpha} s_1'$ then there exists $s_2' \in S_2$ with $s_2 \xrightarrow{\alpha} s_2'$ and $(s_1', s_2') \in R$,
   - If $s_2 \xrightarrow{\alpha} s_2'$ then there exists $s_1' \in S_1$ with $s_1 \xrightarrow{\alpha} s_1'$ and $(s_1', s_2') \in R$.

Bisimulations for other types of models

We only considered transition systems.

Other types of models (deterministic automata, mealy machines, Markov chains, ...) also have bisimulations defined for them.

Bisimulations for other types of models

We only considered transition systems.

Other types of models (deterministic automata, mealy machines, Markov chains, ...) also have bisimulations defined for them.

There is also a **general coalgebraic definition** of bisimilarity.

Definition 33 (**F-bisimulation**). Let $F\colon \mathsf{Set} \to \mathsf{Set}$ be a functor and let (S, α) and (T, β) be two F-coalgebras. A relation $R \subseteq S \times T$ is an *F-bisimulation* if there exists an F-coalgebra structure $\gamma\colon R \to F(R)$ such that the projections $\pi_1\colon R \to S$ and $\pi_2\colon R \to T$ are F-homomorphisms:

$$\begin{tikzcd}
S \arrow[d, "\alpha"'] & R \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] \arrow[d, dashed, "\exists\,\gamma"] & T \arrow[d, "\beta"] \\
F(S) & F(R) \arrow[l, "F(\pi_1)"] \arrow[r, "F(\pi_2)"'] & F(T)
\end{tikzcd}$$

Definition taken from Jan Rutten, The Method of Coalgebra: Exercises in Coinduction, 2019.

Take the course on category theory and coalgebra for more about this!

Bisimulations for other types of models

We used probabilistic bisimulations in our recent AAAI 2023 paper.

Safe Policy Improvement for POMDPs via Finite-State Controllers

Thiago D. Simão,* Marnix Suilen,* Nils Jansen

Department of Software Science
Radboud University
Nijmegen, The Netherlands
{thiago.simao, marnix.suilen, nils.jansen}@ru.nl

Abstract

We study *safe policy improvement* (SPI) for partially observable Markov decision processes (POMDPs). SPI is an offline reinforcement learning (RL) problem that assumes access to (1) historical data about an environment, and (2) the so-called *behavior policy* that previously generated this data by interacting with the environment. SPI methods neither require access to a model nor the environment itself, and aim to reliably improve the behavior policy in an offline manner. Existing methods make the strong assumption that the environment is fully observable. In our novel approach to the SPI problem for POMDPs, we assume that a finite-state controller (FSC) represents the behavior policy and that finite memory is sufficient to derive optimal policies. This assumption allows us to map the POMDP to a finite-state fully observable MDP, the *history MDP*. We estimate this MDP by combining the

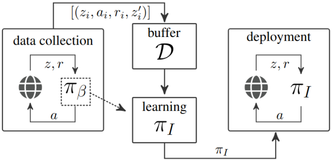


Figure 1: Illustration of the offline reinforcement learning problem in partially observable environments (adapted from Levine et al. 2020). The dashed arrow indicates the setting where the behavior policy is available during learning.

Apartness

We have seen that two states / TS that behave the same are bisimilar.
What about states / TS that **are not bisimilar**?

Apartness

We have seen that two states / TS that behave the same are bisimilar.
What about states / TS that **are not bisimilar**?

This is called **apartness**, and is the exact opposite of bisimilarity.
Recently used for a **new approach to model learning**!

A New Approach for Active Automata Learning Based on Apartness*

Frits Vaandrager✉, Bharat Garhewal, Jurriaan Rot, and Thorsten Wißmann

Institute for Computing and Information Sciences, Radboud University, Nijmegen, the Netherlands

Abstract. We present $L^{\#}$, a new and simple approach to active automata learning. Instead of focusing on equivalence of observations, like the L^* algorithm and its descendants, $L^{\#}$ takes a different perspective: it tries to establish *apartness*, a constructive form of inequality. $L^{\#}$ does not require auxiliary notions such as observation tables or discrimination trees, but operates directly on tree-shaped automata. $L^{\#}$ has the same asymptotic query and symbol complexities as the best existing learning

Material discussed

- Parallel composition: Sections 2.1 (up to 2.1.1), and 2.2 (up to 2.2.4).
- Bisimulations: Sections 7.1, 7.2, and 7.3 (up to 7.3.4).