Abstraction

Reduce (a huge) $TS$ to (a small) $\hat{TS}$ prior or during model checking

Relevant issues:

- What is the formal relationship between $TS$ and $\hat{TS}$?
- Can $\hat{TS}$ be obtained algorithmically and efficiently?
- Which logical fragment (of LTL, CTL, CTL*) is preserved?
- And in what sense?
  - “strong” preservation: positive and negative results carry over
  - “weak” preservation: only positive results carry over
  - “match”: logic equivalence coincides with formal relation
Summary of lecture #1

<table>
<thead>
<tr>
<th>formal relation</th>
<th>trace equivalence</th>
</tr>
</thead>
<tbody>
<tr>
<td>complexity</td>
<td>PSPACE-complete</td>
</tr>
<tr>
<td>logical fragment</td>
<td>LTL</td>
</tr>
<tr>
<td>preservation</td>
<td>strong</td>
</tr>
</tbody>
</table>
## Outlook of today’s lecture

<table>
<thead>
<tr>
<th>formal relation</th>
<th>trace equivalence</th>
<th>bisimulation</th>
</tr>
</thead>
<tbody>
<tr>
<td>complexity</td>
<td>PSPACE-complete</td>
<td>PTIME</td>
</tr>
<tr>
<td>logical fragment</td>
<td>LTL</td>
<td>CTL*</td>
</tr>
<tr>
<td>preservation</td>
<td>strong</td>
<td>match</td>
</tr>
</tbody>
</table>
Bisimulation

\( \mathcal{R} \subseteq S \times S \) is a *bisimulation* on \( TS \) if for any \( (s_1, s_2) \in \mathcal{R} \):

- \( L(s_1) = L(s_2) \)
- if \( s_1' \in Post(s_1) \) then there exists an \( s_2' \in Post(s_2) \) with \( (s_1', s_2') \in \mathcal{R} \)
- if \( s_2' \in Post(s_2) \) then there exists an \( s_1' \in Post(s_1) \) with \( (s_1', s_2') \in \mathcal{R} \)

\( s_1 \) and \( s_2 \) are *bisimilar*, \( s_1 \sim_{TS} s_2 \), if \( (s_1, s_2) \in \mathcal{R} \) for some bisimulation \( \mathcal{R} \) for \( TS \)
Bisimulation

\[
\begin{align*}
\text{s}_1 & \rightarrow \text{s}'_1 & \text{s}_1 & \rightarrow \text{s}'_1 \\
\mathcal{R} & \quad \text{can be completed to} & \mathcal{R} & \\
\text{s}_2 & \rightarrow \text{s}'_2 & \text{s}_2 & \rightarrow \text{s}'_2
\end{align*}
\]

and

\[
\begin{align*}
\text{s}_1 & \rightarrow \text{s}'_1 & \text{s}_1 & \rightarrow \text{s}'_1 \\
\mathcal{R} & \quad \text{can be completed to} & \mathcal{R} & \\
\text{s}_2 & \rightarrow \text{s}'_2 & \text{s}_2 & \rightarrow \text{s}'_2
\end{align*}
\]
Bisimulation on paths

Whenever we have:

\[ s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow s_4 \ldots \]
\[ \mathcal{R} \]
\[ t_0 \]

this can be completed to

\[ s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow s_4 \ldots \]
\[ \mathcal{R} \quad \mathcal{R} \quad \mathcal{R} \quad \mathcal{R} \quad \mathcal{R} \]
\[ t_0 \rightarrow t_1 \rightarrow t_2 \rightarrow t_3 \rightarrow t_4 \ldots \]

proof: by induction on the length of a path
Bisimulation of transition systems

\[
\begin{align*}
TS_1 \sim TS_2 \text{ iff } & \forall s_1 \in I_1. \exists s_2 \in I_2. s_1 \sim_{TS} s_2 \\
& \land \forall s_2 \in I_2. \exists s_1 \in I_1. s_1 \sim_{TS} s_2
\end{align*}
\]
Advanced model checking

∼ vs. trace equivalence

\[ TS_1 \sim TS_2 \text{ implies } \text{Traces}(TS_1) = \text{Traces}(TS_2) \]

bisimilar transition systems thus satisfy the same LT properties!
Quotient transition system

Let $TS = (S, Act, \rightarrow, I, AP, L)$ and bisimulation $\mathcal{R} \subseteq S \times S$ be an *equivalence*. The *quotient* of $TS$ under $\mathcal{R}$ is defined by:

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

where

- $S' = S/\mathcal{R} = \{[s]_{\mathcal{R}} \mid s \in S\}$ with $[s]_{\mathcal{R}} = \{s' \in S \mid (s, s') \in \mathcal{R}\}$
- $I' = \{[s]_{\mathcal{R}} \mid s \in I\}$
- $L'([s]_{\mathcal{R}}) = L(s)$
- $\rightarrow'$ is defined by:

  $$\frac{s \xrightarrow{\alpha} s'}{[s]_{\mathcal{R}} \xrightarrow{\tau'} [s']_{\mathcal{R}}}$$

note that $TS \sim TS/\mathcal{R}$ Why?
Coarsest bisimulation

\( \sim_{TS} \) is a bisimulation, an equivalence, and the coarsest bisimulation for \( TS \)

The quotient under \( \sim_{TS} \) is the smallest under any bisimulation relation
The simplified bakery algorithm

Process 1:

\[
\text{\ldots \ldots} \quad \text{while true} \quad \{
\text{\ldots \ldots}
\]
\[
\begin{align*}
n_1 &: x_1 := x_2 + 1; \\
w_1 &: \text{wait until} \,(x_2 = 0 \lor x_1 < x_2) \quad \{
\text{\ldots \ldots \ldots critical section ...} \\
x_1 &: \quad 0; \\
\} \\
\end{align*}
\]

Process 2:

\[
\text{\ldots \ldots} \quad \text{while true} \quad \{
\text{\ldots \ldots}
\]
\[
\begin{align*}
n_2 &: x_2 := x_1 + 1; \\
w_2 &: \text{wait until} \,(x_1 = 0 \lor x_2 < x_1) \quad \{
\text{\ldots \ldots \ldots critical section ...} \\
x_2 &: \quad 0; \\
\} \\
\end{align*}
\]

this algorithm can be applied to arbitrarily many processes
### Example path fragment

<table>
<thead>
<tr>
<th>process $P_1$</th>
<th>process $P_2$</th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n_1$</td>
<td>$n_2$</td>
<td>0</td>
<td>0</td>
<td>$P_1$ requests access to critical section</td>
</tr>
<tr>
<td>$w_1$</td>
<td>$n_2$</td>
<td>1</td>
<td>0</td>
<td>$P_2$ requests access to critical section</td>
</tr>
<tr>
<td>$w_1$</td>
<td>$w_2$</td>
<td>1</td>
<td>2</td>
<td>$P_1$ enters the critical section</td>
</tr>
<tr>
<td>$c_1$</td>
<td>$w_2$</td>
<td>1</td>
<td>2</td>
<td>$P_1$ leaves the critical section</td>
</tr>
<tr>
<td>$n_1$</td>
<td>$w_2$</td>
<td>0</td>
<td>2</td>
<td>$P_1$ requests access to critical section</td>
</tr>
<tr>
<td>$w_1$</td>
<td>$w_2$</td>
<td>3</td>
<td>2</td>
<td>$P_2$ enters the critical section</td>
</tr>
<tr>
<td>$w_1$</td>
<td>$c_2$</td>
<td>3</td>
<td>2</td>
<td>$P_2$ leaves the critical section</td>
</tr>
<tr>
<td>$w_1$</td>
<td>$n_2$</td>
<td>3</td>
<td>0</td>
<td>$P_2$ requests access to critical section</td>
</tr>
<tr>
<td>$w_1$</td>
<td>$w_2$</td>
<td>3</td>
<td>4</td>
<td>$P_2$ enters the critical section</td>
</tr>
</tbody>
</table>

...
Bakery algorithm as transition system

infinite state space due to possible unbounded increase of counters
Bisimulation
Bisimulation quotient

\[ TS_{Bak}^{abs} = TS_{Bak} / \mathcal{R} \quad \text{for} \quad AP = \{ \text{crit}_1, \text{crit}_2, \text{wait}_1, \text{wait}_2 \} \]
Preservation of properties

- \( TS_{Bak}^{abs} \models \varphi \) with, e.g.,:
  - \( \Box(\neg \text{crit}_1 \lor \neg \text{crit}_2) \) and \( (\Box \Diamond \text{wait}_1 \Rightarrow \Box \Diamond \text{crit}_1) \land (\Box \Diamond \text{wait}_2 \Rightarrow \Box \Diamond \text{crit}_2) \)

- Since \( TS_{Bak}^{abs} \sim TS_{Bak} \), it follows \( \text{Traces}(TS_{Bak}^{abs}) = \text{Traces}(TS_{Bak}) \)

- Since \( \text{Traces}(TS_{Bak}^{abs}) = \text{Traces}(TS_{Bak}) \), it follows \( TS_{Bak} \models \varphi \)

- We thus have \( \text{Traces}(TS_{Bak}^{abs}) = \text{Traces}(TS_{Bak}) \)
Syntax of CTL* 

CTL* state-formulas are formed according to:

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

where $a \in AP$ and $\varphi$ is a path-formula

CTL* path-formulas are formed according to the grammar:

$$
\varphi ::= \Phi \mid \varphi_1 \land \varphi_2 \mid \neg \varphi \mid \Box \varphi \mid \varphi_1 \lor \varphi_2
$$

where $\Phi$ is a state-formula, and $\varphi$, $\varphi_1$ and $\varphi_2$ are path-formulas

in CTL*: $\forall \varphi = \neg \exists \neg \varphi$. This does not hold in CTL!
Relationship between LTL, CTL and CTL*
CTL* equivalence

States \( s_1 \) and \( s_2 \) in \( TS \) (over \( AP \)) are CTL*-equivalent:

\[
s_1 \equiv_{\text{CTL}^*} s_2 \text{ if and only if } (s_1 \models \Phi \iff s_2 \models \Phi)
\]

for all CTL* state formulas over \( AP \)

\[
TS_1 \equiv_{\text{CTL}^*} TS_2 \text{ if and only if } (TS_1 \models \Phi \iff TS_2 \models \Phi)
\]

for any sublogic of CTL*, logical equivalence is defined analogously
Bisimulation vs. CTL* and CTL equivalence

Let $TS$ be a finite transition system (without terminal states) and $s, s'$ states in $TS$.

The following statements are equivalent:

1. $s \sim_{TS} s'$
2. $s$ and $s'$ are CTL-equivalent, i.e., $s \equiv_{\text{CTL}} s'$
3. $s$ and $s'$ are CTL*-equivalent, i.e., $s \equiv_{\text{CTL}^*} s'$

this is proven in three steps: $\equiv_{\text{CTL}} \subseteq \sim \subseteq \equiv_{\text{CTL}^*} \subseteq \equiv_{\text{CTL}}$

important: equivalence is also obtained for any sub-logic containing $\neg$, $\land$ and $\bigcirc$
Example
Bisimulation vs. CTL*-equivalence

For any transition systems $TS$ and $TS'$ (over $AP$) without terminal states:

$TS \sim TS'$ if and only if $TS \equiv_{CTL} TS'$ if and only if $TS \equiv_{CTL^*} TS'$

$\Rightarrow$ prior to model-check $\Phi$, it is safe to first minimize $TS$ wrt. $\sim$

how to obtain such bisimulation quotients?
Basic fixpoint characterization

Consider the function $\mathcal{F} : 2^{S \times S} \rightarrow 2^{S \times S}$:

$$\mathcal{F}(\mathcal{R}) = \{ (s, t) \mid L(s) = L(t) \land \forall s' \in S. \ (s \rightarrow s' \Rightarrow \exists t' \in S. t \rightarrow t' \land (s', t') \in \mathcal{R}) \land \ (t \rightarrow s' \Rightarrow \exists u' \in S. s \rightarrow u' \land (s', u') \in \mathcal{R}) \}$$

$$\sim_{TS} = \mathcal{F}(\sim_{TS})$$ and for any $\mathcal{R}$ such that $\mathcal{F}(\mathcal{R}) = \mathcal{R}$ it holds $\mathcal{R} \subseteq \sim_{TS}$
How to compute the fixpoint of $\mathcal{F}$?

For finite transition system $TS = (S, Act, \to, I, AP, L)$:

$$\sim_{TS} = \bigcap_{i=0}^{\infty} \sim_i$$

that is: $s \sim_{TS} s'$ iff $s \sim_i s'$ for all $i \geq 0$

where $\sim_i$ is defined by:

$$\sim_0 = \{(s, t) \in S \times S \mid L(s) = L(t)\}$$

$$\sim_{i+1} = \mathcal{F}(\sim_i)$$

$this constitutes the basis for the algorithms to follow$
Partitions

- A partition $\Pi = \{B_1, \ldots, B_k\}$ of $S$ satisfies:
  - $B_i$ is non-empty; $B_i$ is called a block
  - $B_i \cap B_j = \emptyset$ for all $i, j$ with $i \neq j$
  - $B_1 \cup \ldots \cup B_k = S$

- $C \subseteq S$ is a super-block of partition $\Pi$ of $S$ if
  $$C = B_{i_1} \cup \ldots \cup B_{i_l} \quad \text{for } B_{i_j} \in \Pi \text{ for } 0 < j \leq l$$

- Partition $\Pi$ is finer than partition $\Pi'$ if:
  $$\forall B \in \Pi. \ (\exists B' \in \Pi'. \ B \subseteq B')$$

  $\Rightarrow$ each block of $\Pi'$ equals the disjoint union of a set of blocks in $\Pi$
  - $\Pi$ is strictly finer than $\Pi'$ if it is finer than $\Pi'$ and $\Pi \neq \Pi'$
Partitions and equivalences

- $\mathcal{R}$ is an equivalence on $S$  $\Rightarrow$  $S/\mathcal{R}$ is a partition of $S$

- Partition $\Pi = \{B_1, \ldots, B_k\}$ of $S$ induces the equivalence relation

  $$\mathcal{R}_\Pi = \{(s, t) \mid \exists B_i \in \Pi. s \in B_i \land t \in B_i\}$$

- $S/\mathcal{R}_\Pi = \Pi$

$\Rightarrow$  there is a one-to-one relationship between partitions and equivalences
Skeleton for bisimulation checking

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

• Iteratively compute a partition of $S$

• Initially: $\Pi_0$ equals $\Pi_{AP} = \{ (s, t) \in S \times S \mid L(s) = L(t) \}$

• Repeat until no change: $\Pi_{i+1} := \text{Refine}(\Pi_i)$

  – loop invariant: $\Pi_i$ is coarser than $S/\sim$ and finer than $\{ S \}$

• Return $\Pi_i$

  – termination: $S \times S \supseteq R_{\Pi_0} \supsetneq R_{\Pi_1} \supsetneq R_{\Pi_2} \supsetneq \ldots \supsetneq R_{\Pi_i} = \sim_{TS}$
  
  – time complexity: maximally $|S|$ iterations needed (why?)

  this is a partition-refinement algorithm
Computing the initial partition $\Pi_{AP}$

- Main idea: construct a *decision tree* of height $k$ for $AP = \{a_1, \ldots, a_k\}$

- Node at depth $i < k$ of the tree: $a_i \in L(s)$ or $a_i \notin L(s)$?

- Leaf $v$ represents equally labeled states:
  - $s \in states(v)$ if and only if decision path for $L(s)$ leads from root to $v$

- Decision tree is created step-by-step
  - new nodes are created when a state is encountered with a new labeling

- Time complexity $\Theta(|S| \cdot |AP|)$
  - a single tree traversal is needed for each state
Example
1. $S/\sim$ is the coarsest partition $\Pi$ of $S$ such that
   (i) $\Pi$ is finer than the initial partition $\Pi_{AP}$, and
   (ii) $B \cap \text{Pre}(C) = \emptyset$ or $B \subseteq \text{Pre}(C)$ for all $B, C \in \Pi$
   i.e., either no or all states in $B$ have a direct successor in $C$

2. If (ii) holds for $\Pi$, then it holds for all $B \in \Pi$ and all superblocks $C'$ of $\Pi$
Proof
How to compute the fixpoint of $\mathcal{F}$?

For finite transition system $TS = (S, Act, \rightarrow, I, AP, L)$:

$$\sim = \bigcap_{i=0}^{\infty} \sim_i$$

where $\sim_i$ is defined by:

$$\sim_0 = \{ (s, t) \in S \times S \mid L(s) = L(t) \}$$
$$\sim_{i+1} = \sim_i \cap \{ (s, t) \mid \forall C \in S/\sim_i . s \in \text{Pre}(C) \text{ iff } t \in \text{Pre}(C) \}$$

the block $C'$ is called a splitter

each relation $\sim_i$ is an equivalence relation
The refinement operator

- Let: \( \text{Refine}(\Pi, C) = \bigcup_{B \in \Pi} \text{Refine}(B, C) \) for \( C \) a superblock of \( \Pi \)
  - where \( \text{Refine}(B, C) = \left\{ B \cap \text{Pre}(C), B \setminus \text{Pre}(C) \right\} \setminus \{\emptyset\} \)

- Basic properties:
  - for \( \Pi \) finer than \( \Pi_{AP} \) and coarser than \( S/\sim \):
    \( \text{Refine}(\Pi, C) \) is finer than \( \Pi \) and \( \text{Refine}(\Pi, C) \) is coarser than \( S/\sim \)
  - \( \Pi \) is strictly coarser than \( S/\sim \) if and only if there exists a splitter for \( \Pi \)
Splitters

• Let $\Pi$ be a partition of $S$ and $C$ a superblock of $\Pi$

• $C$ is a splitter of $\Pi$ if for some $B \in \Pi$:

$$B \cap {\text{Pre}}(C) \neq \emptyset \land B \setminus {\text{Pre}}(C) \neq \emptyset$$

• Block $B$ is stable wrt. $C$ if

$$B \cap {\text{Pre}}(C) = \emptyset \land B \setminus {\text{Pre}}(C) = \emptyset$$

• $\Pi$ is stable wrt. $C$ if any $B \in \Pi$ is stable wrt. $C$
Algorithm skeleton

Input: finite transition system $TS$ over $AP$ with state space $S$
Output: bisimulation quotient space $S/\sim$

\[
\Pi := \Pi_{AP}; \\
\text{while there exists a splitter for } \Pi \text{ do} \\
\quad \text{choose a splitter } C \text{ for } \Pi; \\
\quad \Pi := \text{Refine}(\Pi, C); \\
\text{od} \\
\text{return } \Pi
\]

(* \text{Refine}(\Pi, C) is strictly finer than } \Pi *\)
Example
Which splitter to take?

How to determine a splitter for partition $\Pi_{i+1}$?

1. Simple strategy: $\mathcal{O}(|S| \cdot M)$  
   use any block of $\Pi_i$ as splitter candidate

2. Advanced strategy: $\mathcal{O}(\log |S| \cdot M)$  
   use only “smaller” blocks of $\Pi_i$ as splitter candidates 
   and apply “simultaneous” refinement
A partition-refinement algorithm

[Kanellakis & Smolka, 1983]

*Input:* finite transition system $TS$ with state space $S$

*Output:* bisimulation quotient space $S/\sim$

\[
\begin{align*}
\Pi & := \Pi_{AP}; \\
\Pi_{old} & := \{ S \};
\end{align*}
\]

(* $\Pi_{old}$ is the “previous” partition *)

(* loop invariant: $\Pi$ is coarser than $S/\sim$ and finer than $\Pi_{AP}$ and $\Pi_{old}$ *)

repeat

$\Pi_{old} := \Pi$;

for all $C \in \Pi_{old}$ do

$\Pi := \text{Refine}(\Pi, C)$;

od

until $\Pi = \Pi_{old}$

return $\Pi$
Time complexity

For $TS = (S, Act, \rightarrow, I, AP, L)$ with $M \geq |S|$, the number of edges in $TS$: the partition-refinement algorithm to compute $TS/\sim$ has a worst-case time complexity in $O(|S| \cdot |AP| + |S| \cdot M)$.
Proof
Advanced model checking

An efficiency improvement

• Not necessary to refine with respect to all blocks $C \in \Pi_{old}$

⇒ Consider only the “smaller” subblocks of a previous refinement

• Step $i$: refine $C'$ into $C_1 = C' \cap Pre(D)$ and $C_2 = C' \setminus Pre(D)$

• Step $i+1$: use the smallest $C \in \{ C_1, C_2 \}$ as splitter candidate
  – let $C$ be such that $|C| \leq |C'|/2$, thus $|C| \leq |C' \setminus C|$
  – combine the refinement steps with respect to $C$ and $C' \setminus C$

• $\text{Refine}(\Pi, C, C' \setminus C) = \text{Refine}\left( \text{Refine}(\Pi, C), C' \setminus C \right)$ where $|C| \leq |C' \setminus C|$
  – the decomposed blocks are stable with respect to $C$ and $C' \setminus C$
The new refinement operator

- Let: $\text{Refine}(\Pi, C, C' \setminus C) = \bigcup_{B \in \Pi} \text{Refine}(B, C, C' \setminus C)$
  
  where $\text{Refine}(B, C, C' \setminus C) = \{ B_1, B_2, B_3 \} \setminus \{ \emptyset \}$ with:

  $B_1 = B \cap \text{Pre}(C') \cap \text{Pre}(C' \setminus C)$ to both $C$ and $C \setminus C'$
  $B_2 = (B \cap \text{Pre}(C')) \setminus \text{Pre}(C' \setminus C)$ only to $C$
  $B_3 = (B \cap \text{Pre}(C' \setminus C)) \setminus \text{Pre}(C)$ only to $C' \setminus C$

$\Rightarrow$ blocks $B_1, B_2, B_3$ are stable with respect to $C$ and $C' \setminus C$
Advanced model checking

Improved partition-refinement algorithm

[Paige & Tarjan, 1987]

Input: finite transition system $TS$ with state space $S$

Output: bisimulation quotient space $S/\sim$

\[
\Pi_{old} := \{ S \}; \\
\Pi := \text{Refine}(\Pi_{AP}, S);
\]

(* loop invariant: $\Pi$ is coarser than $S/\sim$ and finer than $\Pi_{AP}$ and $\Pi_{old}$, *)

(* and $\Pi$ is stable with respect to any block in $\Pi_{old}$ *)

repeat

choose block $C' \in \Pi_{old} \setminus \Pi$ and block $C \in \Pi$ with $C \subseteq C'$ and $|C| \leq \frac{|C'|}{2}$;

$\Pi_{old} := \Pi$;

$\Pi := \text{Refine}(\Pi, C, C' \setminus C)$;

until $\Pi = \Pi_{old}$

return $\Pi$
Example
Advanced model checking

Time complexity

For $TS = (S, Act, \rightarrow, I, AP, L)$ with $M \geq |S|$, the \# edges in $TS$:

\[ \text{Time complexity of computing } TS/\sim \text{ is } O(|S| \cdot |AP| + \log |S| \cdot M) \]
Proof
## Summary of today’s lecture

<table>
<thead>
<tr>
<th>formal relation</th>
<th>trace equivalence</th>
<th>bisimulation</th>
</tr>
</thead>
<tbody>
<tr>
<td>complexity</td>
<td>PSPACE-complete</td>
<td>$O(\log</td>
</tr>
<tr>
<td>logical fragment</td>
<td>LTL</td>
<td>CTL*</td>
</tr>
<tr>
<td>preservation</td>
<td>strong</td>
<td>match</td>
</tr>
</tbody>
</table>