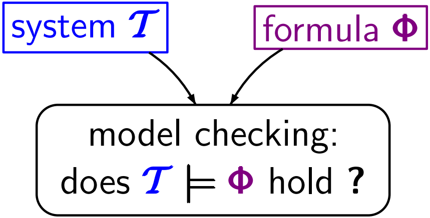
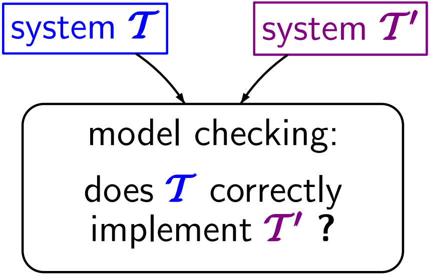
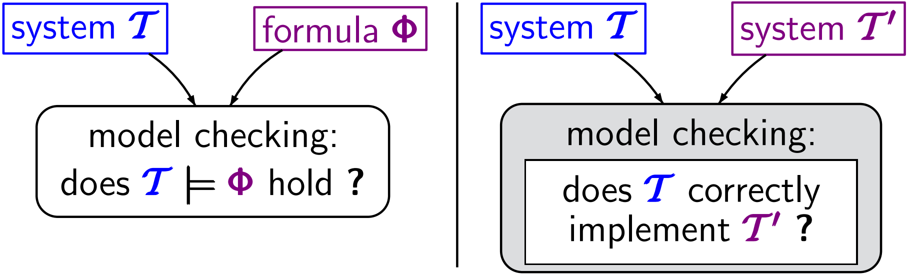
Heterogeneous/homogeneous model checking

Heterogeneous/homogeneous model checking GRM5.5-30



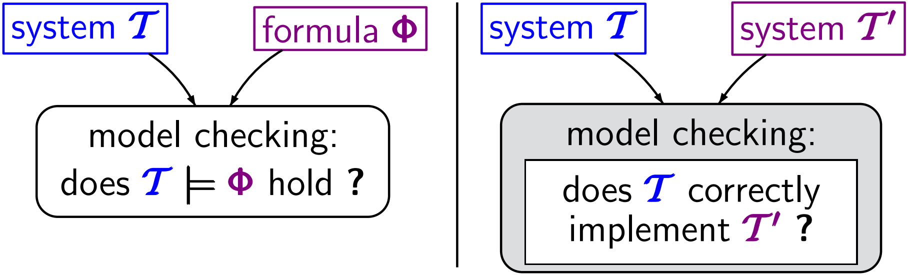


Heterogeneous/homogeneous model checking



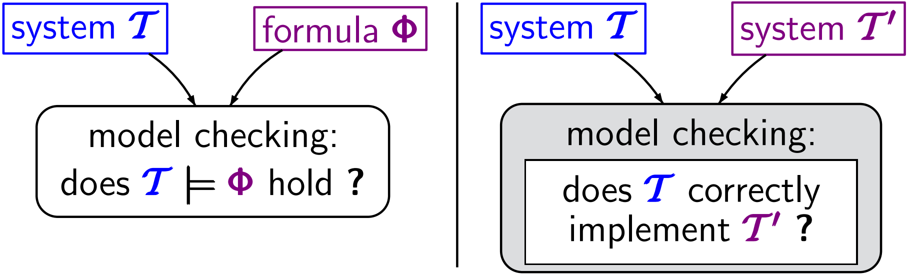
trace inclusion checking
trace equivalence checking

Heterogeneous/homogeneous model checking



trace inclusion checking
trace equivalence checking $\longleftarrow$ ***PSPACE*-complete**

Heterogeneous/homogeneous model checking

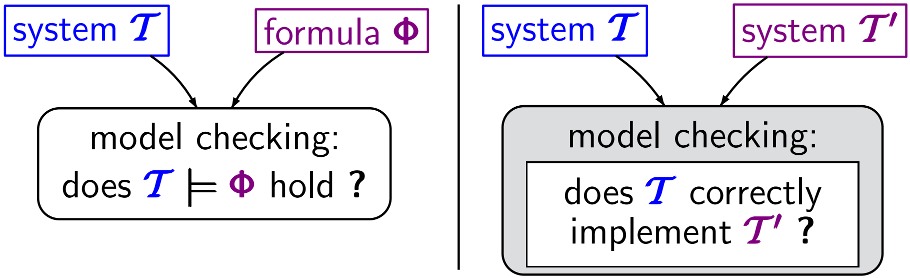


trace inclusion checking
trace equivalence checking $\longleftarrow$ ***PSPACE*-complete**

bisimulation equivalence checking

“does $\mathcal{T} \sim \mathcal{T}'$ hold ?”

Heterogeneous/homogeneous model checking

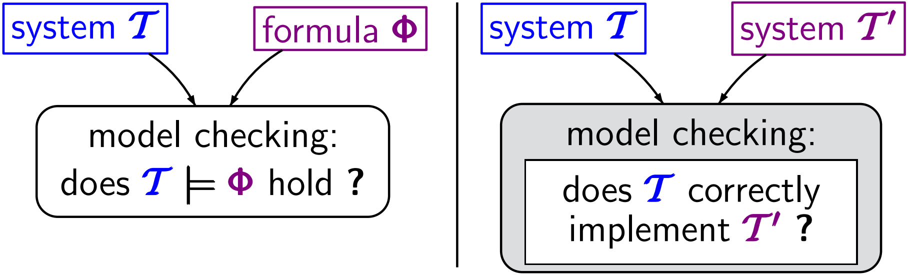


trace inclusion checking
trace equivalence checking $\longleftarrow$ ***PSPACE*-complete**

bisimulation equivalence checking $\longleftarrow$ $\mathcal{O}(m \cdot \log n)$
"does $\mathcal{T} \sim \mathcal{T}'$ hold ?"

n = #states
m = #transitions

Heterogeneous/homogeneous model checking



trace inclusion checking
trace equivalence checking ⟵ ***PSPACE*-complete**

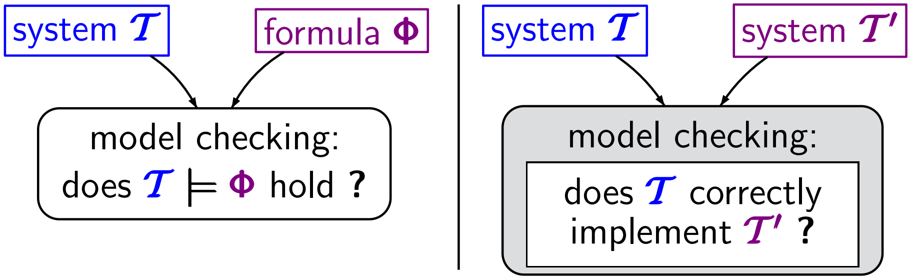
bisimulation equivalence checking ⟵ $\mathcal{O}(m \cdot \log n)$

"does $\mathcal{T} \sim \mathcal{T}'$ hold ?"

refinement checking via simulation

"does $\mathcal{T} \preceq \mathcal{T}'$ hold ?"

Heterogeneous/homogeneous model checking



trace inclusion checking
trace equivalence checking $\longleftarrow$ ***PSPACE*-complete**

bisimulation equivalence checking $\longleftarrow$ $\mathcal{O}(m \cdot \log n)$

"does $\mathcal{T} \sim \mathcal{T}'$ hold ?"

refinement checking via simulation $\longleftarrow$ $\mathcal{O}(m \cdot n)$

"does $\mathcal{T} \preceq \mathcal{T}'$ hold ?"

Refinement checking via simulation

given: 2 finite transition system $\mathcal{T}_1$ and $\mathcal{T}_2$ over the same set of propositions AP

question: does $\mathcal{T}_1 \preceq \mathcal{T}_2$ hold ?

Refinement checking via simulation

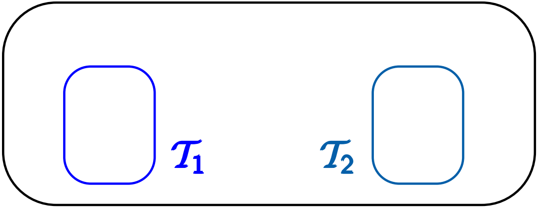
given: $\mathbf{2}$ finite transition system $\mathcal{T}_1$ and $\mathcal{T}_2$ over the same set of propositions AP

question: does $\mathcal{T}_1 \preceq \mathcal{T}_2$ hold ?

Refinement checking via simulation

given: 2 finite transition system $\mathcal{T}_1$ and $\mathcal{T}_2$ over the same set of propositions AP

question: does $\mathcal{T}_1 \preceq \mathcal{T}_2$ hold ?

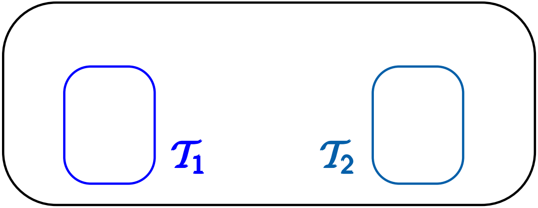


composite system $\mathcal{T} = \mathcal{T}_1 \uplus \mathcal{T}_2$

Refinement checking via simulation

given: 2 finite transition system $\mathcal{T}_1$ and $\mathcal{T}_2$ over the same set of propositions AP

question: does $\mathcal{T}_1 \preceq \mathcal{T}_2$ hold ?



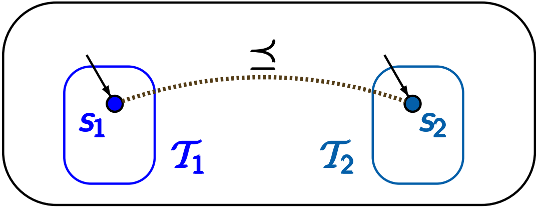
composite system $\mathcal{T} = \mathcal{T}_1 \uplus \mathcal{T}_2$

- compute the simulation preorder $\preceq_{\mathcal{T}}$ on $\mathcal{T}$

Refinement checking via simulation

given: 2 finite transition system $\mathcal{T}_1$ and $\mathcal{T}_2$ over the same set of propositions AP

question: does $\mathcal{T}_1 \preceq \mathcal{T}_2$ hold ?



composite system $\mathcal{T} = \mathcal{T}_1 \uplus \mathcal{T}_2$

- compute the simulation preorder $\preceq_{\mathcal{T}}$ on $\mathcal{T}$
- check whether for all initial states s_1 of $\mathcal{T}_1$ there is an initial state s_2 of $\mathcal{T}_2$ s.t. $s_1 \preceq_{\mathcal{T}} s_2$

Computing the simulation preorder $\preceq_{\mathcal{T}}$

given: finite TS $\mathcal{T} = (S, Act, \to, S_0, AP, L)$
possibly with terminal states

goal: compute the simulation preorder $\preceq_{\mathcal{T}}$

Computing the simulation preorder $\preceq_{\mathcal{T}}$

given: finite TS $\mathcal{T} = (S, Act, \rightarrow, S_0, AP, L)$
possibly with terminal states

goal: compute the simulation preorder $\preceq_{\mathcal{T}}$

$\rightsquigarrow$ simulation equivalence classes

$\rightsquigarrow$ simulation quotient $\mathcal{T}/\simeq$

Computing the simulation preorder $\preceq_{\mathcal{T}}$

given: finite TS $\mathcal{T} = (S, Act, \rightarrow, S_0, AP, L)$
possibly with terminal states

goal: compute the simulation preorder $\preceq_{\mathcal{T}}$

$\rightsquigarrow$ simulation equivalence classes

$\rightsquigarrow$ simulation quotient $\mathcal{T}/\simeq$

method: iterative refinement of relation $\mathcal{R} \subseteq S \times S$

Computing the simulation preorder $\preceq_{\mathcal{T}}$

$\mathcal{R} := \{ (s_1, s_2) \in S \times S : L(s_1) = L(s_2) \};$

WHILE $\mathcal{R}$ is no simulation DO

choose $(s_1, s_2) \in \mathcal{R}$ s.t. $s_1 \longrightarrow s_1'$, but there is no transition $s_2 \longrightarrow s_2'$ with $(s_1', s_2') \in \mathcal{R}$

$\mathcal{R} := \mathcal{R} \setminus \{(s_1, s_2)\}$

OD

return $\mathcal{R}$

Computing the simulation preorder $\preceq_{\mathcal{T}}$

$\mathcal{R} := \{ (s_1, s_2) \in S \times S : L(s_1) = L(s_2) \}$;

WHILE $\mathcal{R}$ is no simulation DO

choose $(s_1, s_2) \in \mathcal{R}$ s.t. $s_1 \rightarrow s_1'$, but there is no transition $s_2 \rightarrow s_2'$ with $(s_1', s_2') \in \mathcal{R}$

$\mathcal{R} := \mathcal{R} \setminus \{(s_1, s_2)\}$

OD

return $\mathcal{R}$ ⟵ $\mathcal{R}$ is the coarsest simulation on $\mathcal{T}$ and therefore $\mathcal{R} = \preceq_{\mathcal{T}}$

$\mathcal{R} := \{ (s_1, s_2) \in S \times S : L(s_1) = L(s_2) \};$

WHILE $\mathcal{R}$ is no simulation DO

choose $(s_1, s_2) \in \mathcal{R}$ s.t. $s_1 \longrightarrow s_1'$, but there is no transition $s_2 \longrightarrow s_2'$ with $(s_1', s_2') \in \mathcal{R}$

$\mathcal{R} := \mathcal{R} \setminus \{(s_1, s_2)\}$

OD

return $\mathcal{R}$

#iterations: $\mathcal{O}(|S|^2)$

$\mathcal{R} := \{ (s_1, s_2) \in S \times S : L(s_1) = L(s_2) \};$

WHILE $\mathcal{R}$ is no simulation DO

choose $(s_1, s_2) \in \mathcal{R}$ s.t. $s_1 \longrightarrow s_1'$, but there is no transition $s_2 \longrightarrow s_2'$ with $(s_1', s_2') \in \mathcal{R}$

$\mathcal{R} := \mathcal{R} \setminus \{(s_1, s_2)\}$

OD

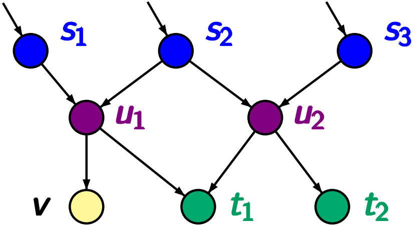
return $\mathcal{R}$

#iterations: $\mathcal{O}(|S|^2)$

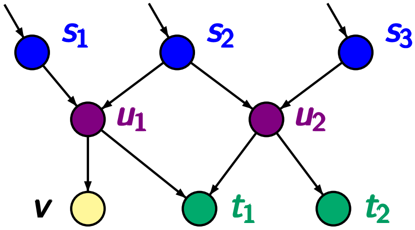
representation of $\mathcal{R}$ by simulator sets

$$Sim_{\mathcal{R}}(s_1) = \{ s_2 \in S : (s_1, s_2) \in \mathcal{R} \}$$

Example: computation of $\preceq$



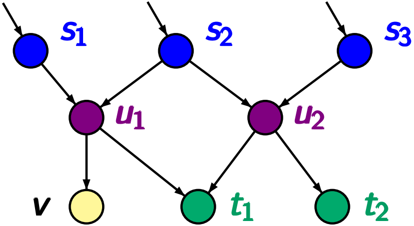
Example: computation of $\preceq$



initially:

$$
\begin{aligned}
Sim(s_i) &= \{s_1, s_2, s_3\} \\
Sim(u_1) &= Sim(u_2) = \{u_1, u_2\} \\
Sim(v) &= \{v\} \\
Sim(t_1) &= Sim(t_2) = \{t_1, t_2\}
\end{aligned}
$$

Example: computation of $\preceq$

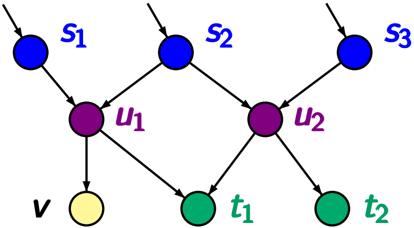


initially:

$$\begin{aligned}
Sim(s_i) &= \{s_1, s_2, s_3\} \\
Sim(u_1) &= Sim(u_2) = \{u_1, u_2\} \\
Sim(v) &= \{v\} \\
Sim(t_1) &= Sim(t_2) = \{t_1, t_2\}
\end{aligned}$$

$u_1 \not\preceq u_2$, as $u_1 \rightarrow v$, $u_2 \not\rightarrow Sim(v)$

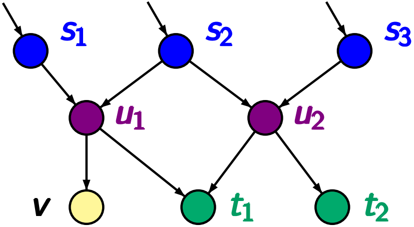
Example: computation of $\preceq$



initially:

$$
\begin{aligned}
Sim(s_i) &= \{s_1, s_2, s_3\} \\
Sim(u_1) &= Sim(u_2) = \{u_1, u_2\} \\
Sim(v) &= \{v\} \\
Sim(t_1) &= Sim(t_2) = \{t_1, t_2\}
\end{aligned}
$$

$u_1 \not\preceq u_2$, as $u_1 \rightarrow v$, $u_2 \not\rightarrow Sim(v)$ $\boxed{Sim(u_1) = \{u_1\}}$



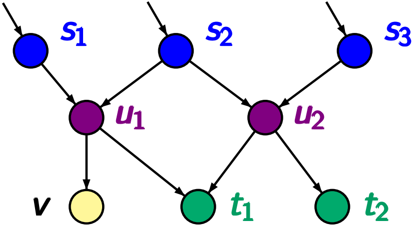
initially:

$$
\begin{aligned}
Sim(s_i) &= \{s_1, s_2, s_3\} \\
Sim(u_1) &= Sim(u_2) = \{u_1, u_2\} \\
Sim(v) &= \{v\} \\
Sim(t_1) &= Sim(t_2) = \{t_1, t_2\}
\end{aligned}
$$

$u_1 \not\preceq u_2$, as $u_1 \rightarrow v$, $u_2 \not\rightarrow Sim(v)$ $\quad$ $Sim(u_1) = \{u_1\}$

$s_1 \not\preceq s_3$, as $s_1 \rightarrow u_1$, $s_3 \not\rightarrow Sim(u_1)$

Example: computation of $\preceq$



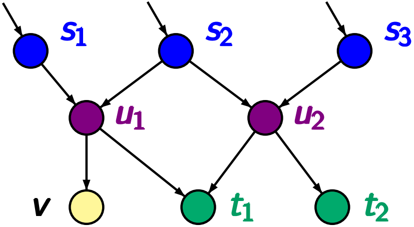
initially:

$$
\begin{aligned}
Sim(s_i) &= \{s_1, s_2, s_3\}\\
Sim(u_1) &= Sim(u_2) = \{u_1, u_2\}\\
Sim(v) &= \{v\}\\
Sim(t_1) &= Sim(t_2) = \{t_1, t_2\}
\end{aligned}
$$

$u_1 \not\preceq u_2$, as $u_1 \to v$, $u_2 \not\to Sim(v)$ $\quad$ $Sim(u_1) = \{u_1\}$

$s_1 \not\preceq s_3$, as $s_1 \to u_1$, $s_3 \not\to Sim(u_1)$ $\quad$ $Sim(s_1) = \{s_1, s_2\}$

Example: computation of $\preceq$



initially:

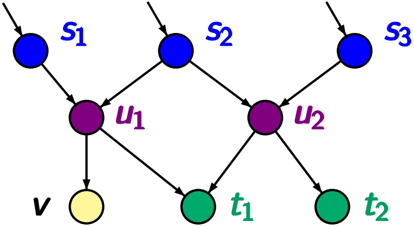
$$Sim(s_i) = \{s_1, s_2, s_3\}$$
$$Sim(u_1) = Sim(u_2) = \{u_1, u_2\}$$
$$Sim(v) = \{v\}$$
$$Sim(t_1) = Sim(t_2) = \{t_1, t_2\}$$

$u_1 \not\preceq u_2$, as $u_1 \to v$, $u_2 \not\to Sim(v)$ $\quad$ $Sim(u_1) = \{u_1\}$

$s_1 \not\preceq s_3$, as $s_1 \to u_1$, $s_3 \not\to Sim(u_1)$ $\quad$ $Sim(s_1) = \{s_1, s_2\}$

$s_2 \not\preceq s_3$, as $s_2 \to u_1$, $s_3 \not\to Sim(u_1)$

Example: computation of $\preceq$



initially:

$$
\begin{aligned}
Sim(s_i) &= \{s_1, s_2, s_3\} \\
Sim(u_1) &= Sim(u_2) = \{u_1, u_2\} \\
Sim(v) &= \{v\} \\
Sim(t_1) &= Sim(t_2) = \{t_1, t_2\}
\end{aligned}
$$

$u_1 \not\preceq u_2$, as $u_1 \to v$, $u_2 \not\to Sim(v)$ $\quad$ $Sim(u_1) = \{u_1\}$

$s_1 \not\preceq s_3$, as $s_1 \to u_1$, $s_3 \not\to Sim(u_1)$ $\quad$ $Sim(s_1) = \{s_1, s_2\}$

$s_2 \not\preceq s_3$, as $s_2 \to u_1$, $s_3 \not\to Sim(u_1)$ $\quad$ $Sim(s_2) = \{s_1, s_2\}$

Computing the simulation preorder $\preceq_{\mathcal{T}}$ GRM5.5-34

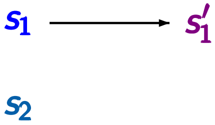
Computing the simulation preorder $\preceq_{\mathcal{T}}$


```
FOR ALL s1 ∈ S DO
      Sim(s1) := {s2 ∈ S : L(s1) = L(s2)}
OD
WHILE ∃s1 ∈ S ∃s2 ∈ Sim(s1) ∃s1' ∈ Post(s1)
                s.t. Post(s2) ∩ Sim(s1') = ∅        DO
      choose such states s1, s2
      Sim(s1) := Sim(s1) \ {s2}
OD
return {(s1, s2) : s2 ∈ Sim(s1)}
```

Computing the simulation preorder $\preceq_{\mathcal{T}}$

```
FOR ALL s1 ∈ S DO
    Sim(s1) := { s2 ∈ S : L(s1) = L(s2) }
OD
WHILE ∃s1 ∈ S ∃s2 ∈ Sim(s1) ∃s1' ∈ Post(s1)
               s.t. Post(s2) ∩ Sim(s1') = ∅        DO
    choose such states s1, s2
    Sim(s1) := Sim(s1) \ {s2}
OD
return {(s1, s2) : s2 ∈ Sim(s1)}
```

FOR ALL $s_1 \in S$ DO
    $Sim(s_1) := \{ s_2 \in S : L(s_1) = L(s_2) \}$
OD
WHILE $\exists s_1 \in S\ \exists s_2 \in Sim(s_1)\ \exists s_1' \in Post(s_1)$
    s.t. $Post(s_2) \cap Sim(s_1') = \varnothing$ DO
    choose such states s_1, s_2
    $Sim(s_1) := Sim(s_1) \setminus \{s_2\}$
OD
return $\{(s_1, s_2) : s_2 \in Sim(s_1)\}$



```
FOR ALL s1 ∈ S DO
    Sim(s1) := { s2 ∈ S : L(s1) = L(s2) }
OD
WHILE ∃s1 ∈ S ∃s2 ∈ Sim(s1) ∃s1' ∈ Post(s1)
            s.t. Post(s2) ∩ Sim(s1') = ∅          DO
    choose such states s1, s2
    Sim(s1) := Sim(s1) \ {s2}
OD
return {(s1, s2) : s2 ∈ Sim(s1)}
```

FOR ALL $s_1 \in S$ DO

$Sim(s_1) := \{ s_2 \in S : L(s_1) = L(s_2) \}$

OD

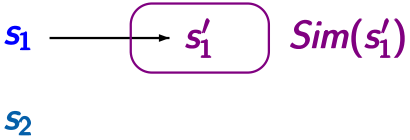
WHILE $\exists s_1 \in S \; \exists s_2 \in Sim(s_1) \; \exists s_1' \in Post(s_1)$ s.t. $Post(s_2) \cap Sim(s_1') = \varnothing$ DO

choose such states s_1, s_2

$Sim(s_1) := Sim(s_1) \setminus \{s_2\}$

OD

return $\{(s_1, s_2) : s_2 \in Sim(s_1)\}$



Computing the simulation preorder $\preceq_{\mathcal{T}}$

FOR ALL $s_1 \in S$ DO

  $Sim(s_1) := \{ s_2 \in S : L(s_1) = L(s_2) \}$

OD

WHILE $\exists s_1 \in S \; \exists s_2 \in Sim(s_1) \; \exists s_1' \in Post(s_1)$

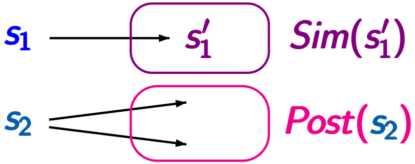
  s.t. $Post(s_2) \cap Sim(s_1') = \varnothing$ DO

  choose such states s_1, s_2

  $Sim(s_1) := Sim(s_1) \setminus \{s_2\}$

OD

return $\{(s_1, s_2) : s_2 \in Sim(s_1)\}$



Computing the simulation preorder $\preceq_{\mathcal{T}}$

FOR ALL $s_1 \in S$ DO

  $Sim(s_1) := \{s_2 \in S : L(s_1) = L(s_2)\}$

OD

WHILE $\exists s_1 \in S \ \exists s_2 \in Sim(s_1) \ \exists s_1' \in Post(s_1)$

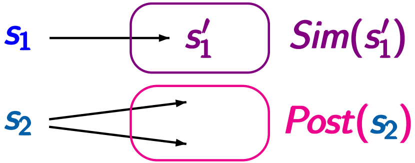
  s.t. $Post(s_2) \cap Sim(s_1') = \varnothing$ DO

  choose such states s_1, s_2

  $Sim(s_1) := Sim(s_1) \setminus \{s_2\}$ ⟵ $s_1 \not\preceq_{\mathcal{T}} s_2$

OD

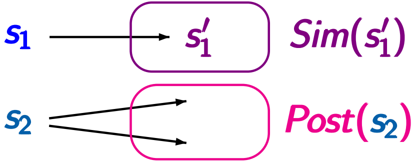
return $\{(s_1, s_2) : s_2 \in Sim(s_1)\}$



Computing the simulation preorder $\preceq_{\mathcal{T}}$


```
FOR ALL s1 ∈ S DO
     Sim(s1) := {s2 ∈ S : L(s1) = L(s2)}
OD
WHILE ∃s1 ∈ S ∃s2 ∈ Sim(s1) ∃s1' ∈ Post(s1)
                s.t. Post(s2) ∩ Sim(s1') = ∅        DO
     choose such states s1, s2
     Sim(s1) := Sim(s1) \ {s2}
OD
return {(s1, s2) : s2 ∈ Sim(s1)}
```

complexity:
$\mathcal{O}(m \cdot |S|^3)$



$m = \#\text{edges}$
$\geqslant |S|$

Computing the simulation preorder $\preceq_{\mathcal{T}}$

FOR ALL $s_1 \in S$ DO

$\quad Sim(s_1) \ := \ \{s_2 \in S : L(s_1) = L(s_2)\}$

OD

WHILE $\exists s_1 \in S \ \exists s_2 \in Sim(s_1) \ \exists s_1' \in Post(s_1)$

$\quad$ s.t. $Post(s_2) \cap Sim(s_1') = \varnothing$ DO

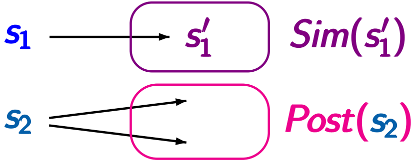
$\quad$ choose such states s_1, s_2

$\quad Sim(s_1) := Sim(s_1) \setminus \{s_2\}$

OD

return $\{(s_1, s_2) : s_2 \in Sim(s_1)\}$

complexity:
$\mathcal{O}(m \cdot |S|^2)$



$m = $ #edges
$\geqslant |S|$

$\mathcal{O}(m \cdot |S|^2)$-algorithm for computing $\preceq_{\mathcal{T}}$

$\mathcal{O}(m \cdot |S|^2)$-algorithm for computing $\preceq_{\mathcal{T}}$

reformulation of the algorithm to compute $\preceq_{\mathcal{T}}$
by means of

$\mathcal{O}(m \cdot |S|^2)$-algorithm for computing $\preceq_{\mathcal{T}}$

reformulation of the algorithm to compute $\preceq_{\mathcal{T}}$ by means of

- counters $\delta(s_1', s_2)$ for $|Post(s_2) \cap Sim(s_1')|$

$\mathcal{O}(m \cdot |S|^2)$-algorithm for computing $\preceq_{\mathcal{T}}$

reformulation of the algorithm to compute $\preceq_{\mathcal{T}}$
by means of

- counters $\delta(s_1', s_2)$ for $|Post(s_2) \cap Sim(s_1')|$
- a set V that organizes all pairs (s_1', s_2)
  where $\delta(s_1', s_2) = 0$

FOR ALL $s_1 \in S$ DO $Sim(s_1) := \{s_2 : L(s_1) = L(s_2)\}$ OD

OD

FOR ALL $s_1 \in S$ DO $Sim(s_1) := \{s_2 : L(s_1) = L(s_2)\}$ OD

FOR ALL s_1', s_2 DO $\delta(s_1', s_2) := \left|Post(s_2) \cap Sim(s_1')\right|$ OD

OD

FOR ALL $s_1 \in S$ DO $Sim(s_1) := \{s_2 : L(s_1) = L(s_2)\}$ OD

FOR ALL s_1', s_2 DO $\delta(s_1', s_2) := \left|Post(s_2) \cap Sim(s_1')\right|$ OD

$V := \{(s_1', s_2) : \delta(s_1', s_2) = 0\}$

OD

```
FOR ALL s_1 ∈ S DO  Sim(s_1) := {s_2 : L(s_1) = L(s_2)} OD
FOR ALL s_1', s_2 DO  δ(s_1', s_2) := |Post(s_2) ∩ Sim(s_1')| OD
V := {(s_1', s_2) : δ(s_1', s_2) = 0}
WHILE V ≠ ∅ DO



OD
```

$$\text{FOR ALL } s_1 \in S \text{ DO } Sim(s_1) := \{s_2 : L(s_1) = L(s_2)\} \text{ OD}$$

$$\text{FOR ALL } s_1', s_2 \text{ DO } \delta(s_1', s_2) := \left|Post(s_2) \cap Sim(s_1')\right| \text{ OD}$$

$$V := \{(s_1', s_2) : \delta(s_1', s_2) = 0\}$$

$$\text{WHILE } V \neq \varnothing \text{ DO}$$

OD

FOR ALL $s_1 \in S$ DO $Sim(s_1) := \{s_2 : L(s_1) = L(s_2)\}$ OD

FOR ALL s_1', s_2 DO $\delta(s_1', s_2) := \left| Post(s_2) \cap Sim(s_1') \right|$ OD

$V := \{(s_1', s_2) : \delta(s_1', s_2) = 0\}$

WHILE $V \neq \varnothing$ DO

choose $(s_1', s_2) \in V$ and remove (s_1', s_2) from V

OD

```
FOR ALL s1 ∈ S DO  Sim(s1) := {s2 : L(s1) = L(s2)} OD
FOR ALL s1', s2 DO  δ(s1', s2) := |Post(s2) ∩ Sim(s1')| OD
V := {(s1', s2) : δ(s1', s2) = 0}
WHILE V ≠ ∅ DO
   choose (s1', s2) ∈ V and remove (s1', s2) from V
   FOR ALL s1 ∈ Pre(s1') with s2 ∈ Sim(s1) DO
      Sim(s1) := Sim(s1) \ {s2}



   OD
OD
```

$$\text{FOR ALL } s_1 \in S \text{ DO } Sim(s_1) := \{s_2 : L(s_1) = L(s_2)\} \text{ OD}$$

FOR ALL $s_1 \in S$ DO $Sim(s_1) := \{s_2 : L(s_1) = L(s_2)\}$ OD

FOR ALL s_1', s_2 DO $\delta(s_1', s_2) := \left|Post(s_2) \cap Sim(s_1')\right|$ OD

$V := \{(s_1', s_2) : \delta(s_1', s_2) = 0\}$

WHILE $V \neq \varnothing$ DO

choose $(s_1', s_2) \in V$ and remove (s_1', s_2) from V

FOR ALL $s_1 \in Pre(s_1')$ with $s_2 \in Sim(s_1)$ DO

$Sim(s_1) := Sim(s_1) \setminus \{s_2\}$

FOR ALL $u_2 \in Pre(s_2)$ DO

OD

OD

FOR ALL $s_1 \in S$ DO $Sim(s_1) := \{s_2 : L(s_1) = L(s_2)\}$ OD

FOR ALL s_1', s_2 DO $\delta(s_1', s_2) := \left|Post(s_2) \cap Sim(s_1')\right|$ OD

$V := \{(s_1', s_2) : \delta(s_1', s_2) = 0\}$

WHILE $V \neq \varnothing$ DO

  choose $(s_1', s_2) \in V$ and remove (s_1', s_2) from V

  FOR ALL $s_1 \in Pre(s_1')$ with $s_2 \in Sim(s_1)$ DO

    $Sim(s_1) := Sim(s_1) \setminus \{s_2\}$

    FOR ALL $u_2 \in Pre(s_2)$ DO

      $\delta(s_1, u_2) := \delta(s_1, u_2) - 1$

  OD

OD

```
FOR ALL s1 ∈ S DO Sim(s1) := {s2 : L(s1) = L(s2)} OD
FOR ALL s1', s2 DO δ(s1', s2) := |Post(s2) ∩ Sim(s1')| OD
V := {(s1', s2) : δ(s1', s2) = 0}
WHILE V ≠ ∅ DO
  choose (s1', s2) ∈ V and remove (s1', s2) from V
  FOR ALL s1 ∈ Pre(s1') with s2 ∈ Sim(s1) DO
    Sim(s1) := Sim(s1) \ {s2}
    FOR ALL u2 ∈ Pre(s2) DO
      δ(s1, u2) := δ(s1, u2) − 1
      IF δ(s1, u2) = 0 THEN insert (s1, u2) in V FI
    OD
  OD
OD
```

```
FOR ALL s1 ∈ S DO  Sim(s1) := {s2 : L(s1) = L(s2)} OD
FOR ALL s1', s2 DO  δ(s1', s2) := |Post(s2) ∩ Sim(s1')| OD
V := {(s1', s2) : δ(s1', s2) = 0}
WHILE V ≠ ∅ DO
   choose (s1', s2) ∈ V and remove (s1', s2) from V
   FOR ALL s1 ∈ Pre(s1') with s2 ∈ Sim(s1) DO
      Sim(s1) := Sim(s1) \ {s2}
      FOR ALL u2 ∈ Pre(s2) DO
         δ(s1, u2) := δ(s1, u2) − 1
         IF δ(s1, u2) = 0 THEN insert (s1, u2) in V FI
      OD
   OD
OD
```

```
FOR ALL s_1 ∈ S DO  Sim(s_1) := {s_2 : L(s_1) = L(s_2)} OD
FOR ALL s_1', s_2 DO  δ(s_1', s_2) := |Post(s_2) ∩ Sim(s_1')| OD
V := {(s_1', s_2) : δ(s_1', s_2) = 0}
WHILE V ≠ ∅ DO
   choose (s_1', s_2) ∈ V and remove (s_1', s_2) from V
   FOR ALL s_1 ∈ Pre(s_1') with s_2 ∈ Sim(s_1) DO
     Sim(s_1) := Sim(s_1) \ {s_2}
     FOR ALL u_2 ∈ Pre(s_2) DO
       δ(s_1, u_2) := δ(s_1, u_2) − 1
       IF δ(s_1, u_2) = 0 THEN insert (s_1, u_2) in V FI
     OD
   OD
OD
```

in total: $\mathcal{O}(m \cdot |S|)$

```
FOR ALL s_1 ∈ S DO  Sim(s_1) := {s_2 : L(s_1) = L(s_2)} OD
FOR ALL s'_1, s_2 DO  δ(s'_1, s_2) := |Post(s_2) ∩ Sim(s'_1)| OD
V := {(s'_1, s_2) : δ(s'_1, s_2) = 0}
WHILE V ≠ ∅ DO
   choose (s'_1, s_2) ∈ V and remove (s'_1, s_2) from V
   FOR ALL s_1 ∈ Pre(s'_1) with s_2 ∈ Sim(s_1) DO
      Sim(s_1) := Sim(s_1) \ {s_2}
      FOR ALL u_2 ∈ Pre(s_2) DO
         δ(s_1, u_2) := δ(s_1, u_2) − 1
         IF δ(s_1, u_2) = 0 THEN insert (s_1, u_2) in V FI
      OD
   OD
OD
```

FOR ALL $s_1 \in S$ DO $Sim(s_1) := \{s_2 : L(s_1) = L(s_2)\}$ OD

FOR ALL s_1', s_2 DO $\delta(s_1', s_2) := \left|Post(s_2) \cap Sim(s_1')\right|$ OD

$V := \{(s_1', s_2) : \delta(s_1', s_2) = 0\}$

WHILE $V \neq \varnothing$ DO

choose $(s_1', s_2) \in V$ and remove (s_1', s_2) from V

FOR ALL $s_1 \in Pre(s_1')$ with $s_2 \in Sim(s_1)$ DO

$Sim(s_1) := Sim(s_1) \setminus \{s_2\}$

FOR ALL $u_2 \in Pre(s_2)$ DO

$\delta(s_1, u_2) := \delta(s_1, u_2) - 1$

IF $\delta(s_1, u_2) = 0$ THEN insert (s_1, u_2) in V FI

OD

OD

OD

cost per iteration
$\mathcal{O}(m)$

FOR ALL $s_1 \in S$ DO $Sim(s_1) := \{s_2 : L(s_1) = L(s_2)\}$ OD

FOR ALL s_1', s_2 DO $\delta(s_1', s_2) := \left|Post(s_2) \cap Sim(s_1')\right|$ OD

$V := \{(s_1', s_2) : \delta(s_1', s_2) = 0\}$

WHILE $V \neq \varnothing$ DO ⟵ $\#\text{iterations} \leqslant |S|^2$

choose $(s_1', s_2) \in V$ and remove (s_1', s_2) from V

FOR ALL $s_1 \in Pre(s_1')$ with $s_2 \in Sim(s_1)$ DO

$Sim(s_1) := Sim(s_1) \setminus \{s_2\}$

cost per iteration $\mathcal{O}(m)$

FOR ALL $u_2 \in Pre(s_2)$ DO

$\delta(s_1, u_2) := \delta(s_1, u_2) - 1$

IF $\delta(s_1, u_2) = 0$ THEN insert (s_1, u_2) in V FI

OD

OD

OD

An $\mathcal{O}(m \cdot |S|)$-algorithm for computing $\preceq_{\mathcal{T}}$

... algorithm by Henzinger, Henzinger, and Kopke

An $\mathcal{O}(m \cdot |S|)$-algorithm for computing $\preceq_{\mathcal{T}}$

... algorithm by Henzinger, Henzinger, and Kopke

relies on the following observations:

An $\mathcal{O}(m \cdot |S|)$-algorithm for computing $\preceq_\mathcal{T}$

... algorithm by Henzinger, Henzinger, and Kopke

relies on the following observations:

- suppose $s_1 \longrightarrow s_1'$ and $s_2 \longrightarrow s_2'$ are transitions s.t. $s_2 \in Sim(s_1)$ and $s_2' \in Sim(s_1')$.

An $\mathcal{O}(m \cdot |S|)$-algorithm for computing $\preceq_{\mathcal{T}}$

... algorithm by Henzinger, Henzinger, and Kopke

relies on the following observations:

- suppose $s_1 \longrightarrow s_1'$ and $s_2 \longrightarrow s_2'$ are transitions s.t. $s_2 \in Sim(s_1)$ and $s_2' \in Sim(s_1')$.
- suppose that s_2' will be removed from $Sim(s_1')$.

... algorithm by Henzinger, Henzinger, and Kopke

relies on the following observations:

- suppose $s_1 \longrightarrow s_1'$ and $s_2 \longrightarrow s_2'$ are transitions s.t. $s_2 \in Sim(s_1)$ and $s_2' \in Sim(s_1')$.
- suppose that s_2' will be removed from $Sim(s_1')$.

Then: if $Post(s_2) \cap Sim(s_1') = \{s_2'\}$ then $s_1 \not\preceq s_2$ and s_2 can be removed from $Sim(s_1)$.

... algorithm by Henzinger, Henzinger, and Kopke

relies on the following observations:

- suppose $s_1 \rightarrow s'_1$ and $s_2 \rightarrow s'_2$ are transitions s.t. $s_2 \in Sim(s_1)$ and $s'_2 \in Sim(s'_1)$.
- suppose that s'_2 will be removed from $Sim(s'_1)$.

Then: if $Post(s_2) \cap Sim(s'_1) = \{s'_2\}$ then $s_1 \not\preceq s_2$ and s_2 can be removed from $Sim(s_1)$.

↑

idea: collect all such states s_2 in $Remove(s'_1)$

Idea of the HHK-algorithm

If s_2' is removed from $Sim(s_1')$ then regard all direct predecessors s_1 of s_1' and remove all states in

$$Remove(s_1') \;=\; Pre(Sim_{old}(s_1')) \setminus Pre(Sim(s_1'))$$

from $Sim(s_1)$.

Idea of the HHK-algorithm

If s_2' is removed from $Sim(s_1')$ then regard all direct predecessors s_1 of s_1' and remove all states in

$$Remove(s_1') \;=\; Pre(Sim_{old}(s_1')) \setminus Pre(Sim(s_1'))$$

from $Sim(s_1)$. I.e., we put

$$Sim_{old}(s_1') := Sim(s_1')$$
$$Sim(s_1) := Sim(s_1) \setminus Remove(s_1') \text{ for } s_1 \in Pre(s_1')$$

Idea of the HHK-algorithm

If s_2' is removed from $Sim(s_1')$ then regard all direct predecessors s_1 of s_1' and remove all states in

$$Remove(s_1') \;=\; Pre(Sim_{old}(s_1')) \setminus Pre(Sim(s_1'))$$

from $Sim(s_1)$. I.e., we put

$$Sim_{old}(s_1') := Sim(s_1')$$
$$Sim(s_1) := Sim(s_1) \setminus Remove(s_1') \text{ for } s_1 \in Pre(s_1')$$

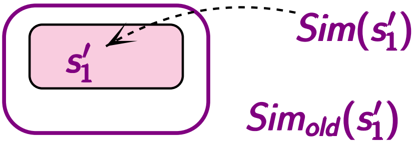
Idea of the HHK-algorithm

If s'_2 is removed from $Sim(s'_1)$ then regard all direct predecessors s_1 of s'_1 and remove all states in

$$Remove(s'_1) \;=\; Pre(Sim_{old}(s'_1)) \setminus Pre(Sim(s'_1))$$

from $Sim(s_1)$. I.e., we put

$$Sim_{old}(s'_1) := Sim(s'_1)$$
$$Sim(s_1) := Sim(s_1) \setminus Remove(s'_1) \text{ for } s_1 \in Pre(s'_1)$$



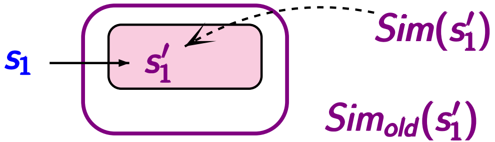
Idea of the HHK-algorithm

If s_2' is removed from $Sim(s_1')$ then regard all direct predecessors s_1 of s_1' and remove all states in

$$Remove(s_1') = Pre(Sim_{old}(s_1')) \setminus Pre(Sim(s_1'))$$

from $Sim(s_1)$. I.e., we put

$$Sim_{old}(s_1') := Sim(s_1')$$
$$Sim(s_1) := Sim(s_1) \setminus Remove(s_1') \text{ for } s_1 \in Pre(s_1')$$



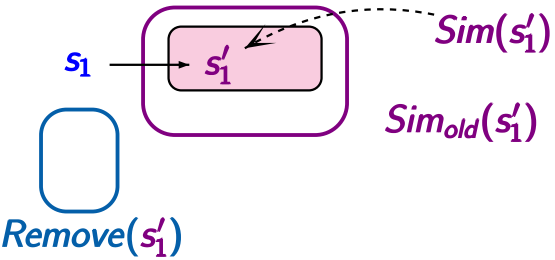
Idea of the HHK-algorithm

If s_2' is removed from $Sim(s_1')$ then regard all direct predecessors s_1 of s_1' and remove all states in

$$Remove(s_1') = Pre(Sim_{old}(s_1')) \setminus Pre(Sim(s_1'))$$

from $Sim(s_1)$. I.e., we put

$$Sim_{old}(s_1') := Sim(s_1')$$
$$Sim(s_1) := Sim(s_1) \setminus Remove(s_1') \text{ for } s_1 \in Pre(s_1')$$



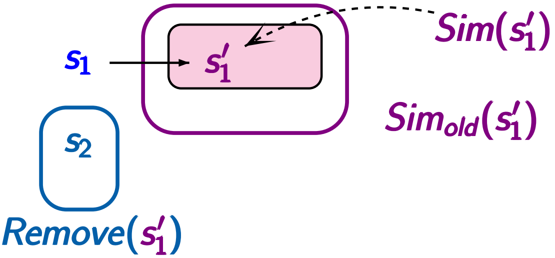
Idea of the HHK-algorithm

If s_2' is removed from $Sim(s_1')$ then regard all direct predecessors s_1 of s_1' and remove all states in

$$Remove(s_1') = Pre(Sim_{old}(s_1')) \setminus Pre(Sim(s_1'))$$

from $Sim(s_1)$. I.e., we put

$$Sim_{old}(s_1') := Sim(s_1')$$
$$Sim(s_1) := Sim(s_1) \setminus Remove(s_1') \text{ for } s_1 \in Pre(s_1')$$



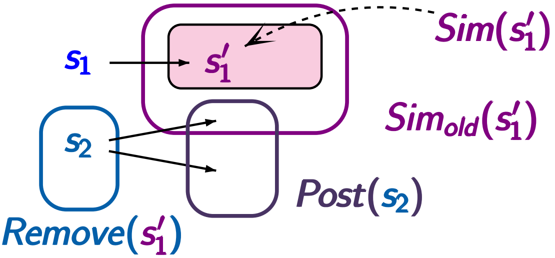
Idea of the HHK-algorithm

If s_2' is removed from $Sim(s_1')$ then regard all direct predecessors s_1 of s_1' and remove all states in

$$Remove(s_1') \;=\; Pre(Sim_{old}(s_1')) \setminus Pre(Sim(s_1'))$$

from $Sim(s_1)$. I.e., we put

$$Sim_{old}(s_1') := Sim(s_1')$$
$$Sim(s_1) := Sim(s_1) \setminus Remove(s_1') \text{ for } s_1 \in Pre(s_1')$$

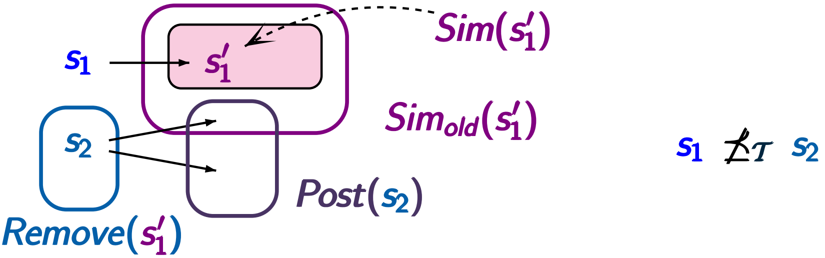


If s_2' is removed from $Sim(s_1')$ then regard all direct predecessors s_1 of s_1' and remove all states in

$$Remove(s_1') = Pre(Sim_{old}(s_1')) \setminus Pre(Sim(s_1'))$$

from $Sim(s_1)$. I.e., we put

$$Sim_{old}(s_1') := Sim(s_1')$$
$$Sim(s_1) := Sim(s_1) \setminus Remove(s_1') \text{ for } s_1 \in Pre(s_1')$$



HHK-algorithm (first version)


```
FOR ALL states s'_1 DO
    Sim_old(s'_1) := S
    Sim(s'_1) := { s'_2 ∈ S : L(s'_1) = L(s'_2) }
OD
```

FOR ALL states s'_1 DO

$Sim_{old}(s'_1) := S$

$Sim(s'_1) := \{ s'_2 \in S : L(s'_1) = L(s'_2) \}$

OD

HHK-algorithm (first version)

FOR ALL states s_1' DO

$\quad Sim_{old}(s_1') := S$

$\quad Sim(s_1') := \{s_2' \in S : L(s_1') = L(s_2')\}$

OD

WHILE $\exists$ state s_1' with $Sim(s_1') \neq Sim_{old}(s_1')$ DO

HHK-algorithm (first version)

FOR ALL states s_1' DO

$Sim_{old}(s_1') := S$

$Sim(s_1') := \{s_2' \in S : L(s_1') = L(s_2')\}$

OD

WHILE $\exists$ state s_1' with $Sim(s_1') \neq Sim_{old}(s_1')$ DO

choose such a state s_1';

HHK-algorithm (first version)


```
FOR ALL states s'_1 DO
    Sim_old(s'_1) := S
    Sim(s'_1) := { s'_2 ∈ S : L(s'_1) = L(s'_2) }
OD
WHILE ∃ state s'_1 with Sim(s'_1) ≠ Sim_old(s'_1) DO
    choose such a state s'_1;
    Remove(s'_1) := Pre(Sim_old(s'_1)) \ Pre(Sim(s'_1));
```

HHK-algorithm (first version)

FOR ALL states s_1' DO

$Sim_{old}(s_1') := S$

$Sim(s_1') := \{s_2' \in S : L(s_1') = L(s_2')\}$

OD

WHILE $\exists$ state s_1' with $Sim(s_1') \neq Sim_{old}(s_1')$ DO

choose such a state s_1';

$Remove(s_1') := Pre(Sim_{old}(s_1')) \setminus Pre(Sim(s_1'))$;

FOR ALL $s_1 \in Pre(s_1')$ DO

HHK-algorithm (first version)


```
FOR ALL states s'_1 DO
    Sim_old(s'_1) := S
    Sim(s'_1) := { s'_2 ∈ S : L(s'_1) = L(s'_2) }
OD
WHILE ∃ state s'_1 with Sim(s'_1) ≠ Sim_old(s'_1) DO
    choose such a state s'_1;
    Remove(s'_1) := Pre(Sim_old(s'_1)) \ Pre(Sim(s'_1));
    FOR ALL s_1 ∈ Pre(s'_1) DO
        Sim(s_1) := Sim(s_1) \ Remove(s'_1)
    OD ;
```

HHK-algorithm (first version)

FOR ALL states s_1' DO

$Sim_{old}(s_1') := S$

$Sim(s_1') := \{ s_2' \in S : L(s_1') = L(s_2') \}$

OD

WHILE $\exists$ state s_1' with $Sim(s_1') \neq Sim_{old}(s_1')$ DO

choose such a state s_1';

$Remove(s_1') := Pre(Sim_{old}(s_1')) \setminus Pre(Sim(s_1'))$;

FOR ALL $s_1 \in Pre(s_1')$ DO

$Sim(s_1) := Sim(s_1) \setminus Remove(s_1')$

OD ;

$Sim_{old}(s_1') := Sim(s_1')$

OD

HHK-algorithm (first version)


```
FOR ALL states s'_1 DO
    Sim_old(s'_1) := S
    Sim(s'_1) := { s'_2 ∈ S : L(s'_1) = L(s'_2) }
OD
WHILE ∃ state s'_1 with Sim(s'_1) ≠ Sim_old(s'_1) DO
    choose such a state s'_1;
    Remove(s'_1) := Pre(Sim_old(s'_1)) \ Pre(Sim(s'_1));
    FOR ALL s_1 ∈ Pre(s'_1) DO
        Sim(s_1) := Sim(s_1) \ Remove(s'_1)
    OD ;
    Sim_old(s'_1) := Sim(s'_1)
OD
return {(s_1, s'_2) : s'_2 ∈ Sim(s'_1)}
```

HHK-algorithm (first version)

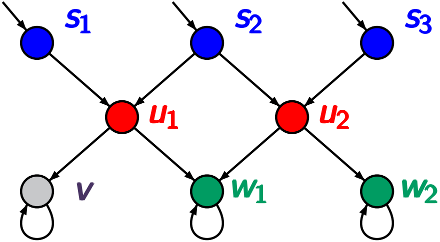
> if s_2' is terminal then so is s_1'

```
FOR ALL states s'_1 DO
    Sim_old(s'_1) := S
    Sim(s'_1) := { s'_2 ∈ S : L(s'_1) = L(s'_2) and ... }
OD
WHILE ∃ state s'_1 with Sim(s'_1) ≠ Sim_old(s'_1) DO
    choose such a state s'_1;
    Remove(s'_1) := Pre(Sim_old(s'_1)) \ Pre(Sim(s'_1));
    FOR ALL s_1 ∈ Pre(s'_1) DO
        Sim(s_1) := Sim(s_1) \ Remove(s'_1)
    OD ;
    Sim_old(s'_1) := Sim(s'_1)
OD
return {(s_1, s'_2) : s'_2 ∈ Sim(s'_1)}
```

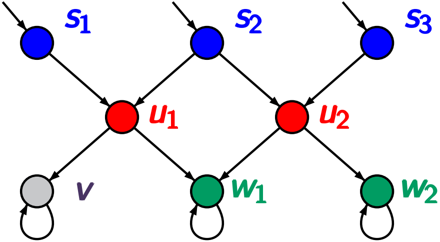
HHK-algorithm (first version)


```
FOR ALL states s'_1 DO
    Sim_old(s'_1) := "undefined"
    Sim(s'_1) := { s'_2 ∈ s : L(s'_1) = L(s'_2) and ... }
OD
WHILE ∃ state s'_1 with Sim(s'_1) ≠ Sim_old(s'_1) DO
  choose such a state s'_1
  IF Sim_old(s'_1) = "undefined"
    THEN Remove(s'_1) := S \ Pre(Sim(s'_1))
    ELSE Remove(s'_1) := Pre(Sim_old(s'_1)) \ Pre(Sim(s'_1))
  FI
  FOR ALL s_1 ∈ Pre(s'_1) DO
  ...
```

Example: HHK-algorithm



Example: HHK-algorithm



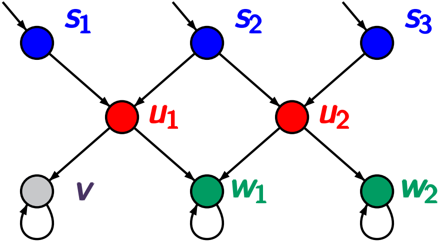
initially:

$Sim_{old}(t) = \bot$ for all states t

$Sim(s_1) = \{s_1, s_2, s_3\}$

$\vdots$

Example: HHK-algorithm



initially:

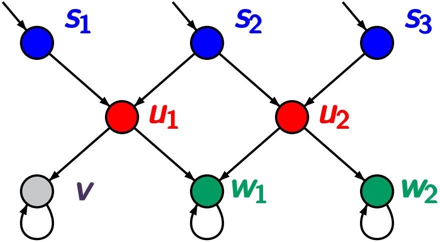
$Sim_{old}(t) = \bot$ for all states t

$Sim(s_1) = \{s_1, s_2, s_3\}$

$\vdots$

choose state $s_1' = v$ with $Sim_{old}(v) \neq Sim(v)$:

Example: HHK-algorithm



initially:

$Sim_{old}(t) = \bot$ for all states t

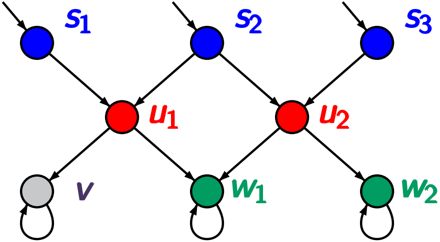
$Sim(s_1) = \{s_1, s_2, s_3\}$

$\vdots$

choose state $s_1' = v$ with $Sim_{old}(v) \neq Sim(v)$:

$$Remove(v) = S \setminus Pre(Sim(v)) = S \setminus \{u_1\}$$

Example: HHK-algorithm



initially:

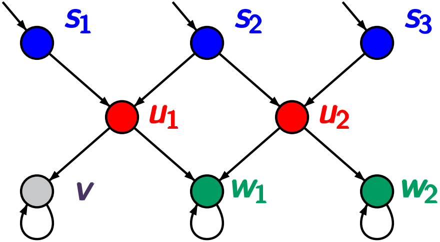
$Sim_{old}(t) = \perp$ for all states t

$Sim(s_1) = \{s_1, s_2, s_3\}$

$\vdots$

choose state $s'_1 = v$ with $Sim_{old}(v) \neq Sim(v)$:

$$Remove(v) = S \setminus Pre(Sim(v)) = S \setminus \{u_1\}$$
$$Sim(u_1) \;:=\; Sim(u_1) \setminus Remove(v) = \{u_1\}$$



initially:

$Sim_{old}(t) = \perp$ for all states t

$Sim(s_1) = \{s_1, s_2, s_3\}$

$\vdots$

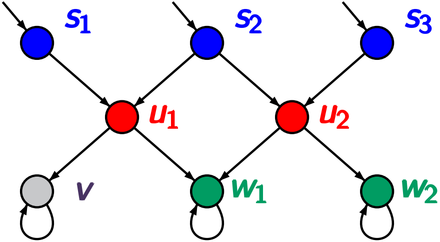
choose state $s_1' = v$ with $Sim_{old}(v) \neq Sim(v)$:

$$Remove(v) = S \setminus Pre(Sim(v)) = S \setminus \{u_1\}$$
$$Sim(u_1) := Sim(u_1) \setminus Remove(v) = \{u_1\}$$

$\uparrow$

$u_1 \longrightarrow v$ can't be simulated by any of the states in $Remove(v)$

Example: HHK-algorithm



initially:

$Sim_{old}(t) = \bot$ for all states t

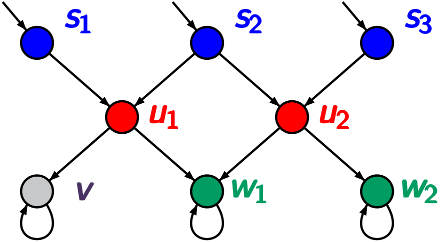
$Sim(s_1) = \{s_1, s_2, s_3\}$

$\vdots$

choose state $s_1' = v$ with $Sim_{old}(v) \neq Sim(v)$:

$$Remove(v) = S \setminus Pre(Sim(v)) = S \setminus \{u_1\}$$
$$Sim(u_1) := Sim(u_1) \setminus Remove(v) = \{u_1\}$$
$$Sim_{old}(v) := Sim(v) = \{v\}$$

Example: HHK-algorithm



initially:

$Sim_{old}(t) = \bot$ for all states t

$Sim(s_1) = \{s_1, s_2, s_3\}$

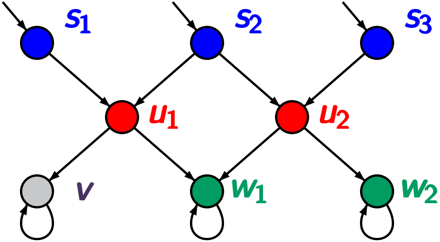
$\vdots$

choose state $s_1' = v$ with $Sim_{old}(v) \neq Sim(v)$:

$$Remove(v) = S \setminus Pre(Sim(v)) = S \setminus \{u_1\}$$
$$Sim(u_1) := Sim(u_1) \setminus Remove(v) = \{u_1\}$$
$$Sim_{old}(v) := Sim(v) = \{v\}$$

choose next state s_1'

Example: HHK-algorithm



initially:

$Sim_{old}(t) = \bot$ for all states t

$Sim(s_1) = \{s_1, s_2, s_3\}$

$\vdots$

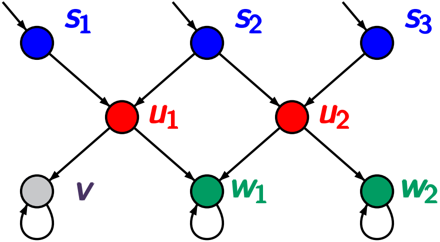
choose state $s_1' = v$ with $Sim_{old}(v) \neq Sim(v)$:

$$Remove(v) = S \setminus Pre(Sim(v)) = S \setminus \{u_1\}$$
$$Sim(u_1) \; := \; Sim(u_1) \setminus Remove(v) = \{u_1\}$$
$$Sim_{old}(v) := Sim(v) = \{v\}$$

choose next state $s_1' = s_1$ with $Sim_{old}(s_1) \neq Sim(s_1)$:

no change in $Sim(\ldots)$, as $Pre(s_1) = \varnothing$

Example: HHK-algorithm



initially:

$Sim_{old}(t) = \perp$ for all states t

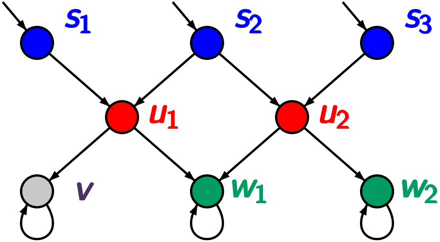
$Sim(s_1) = \{s_1, s_2, s_3\}$

⋮

$s_1' = v$: $Sim(u_1) = \{u_1\}$, $Sim_{old}(v) = Sim(v) = \{v\}$

$s_1' = s_i$: $Sim_{old}(s_i) = Sim(s_i) = \{s_1, s_2, s_3\}$

Example: HHK-algorithm



initially:

$Sim_{old}(t) = \bot$ for all states t

$Sim(s_1) = \{s_1, s_2, s_3\}$

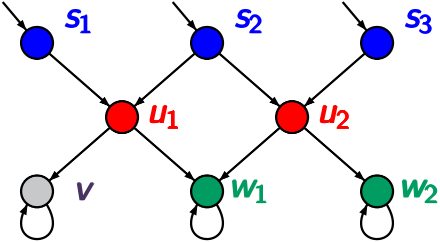
$\vdots$

$s_1' = v$: $Sim(u_1) = \{u_1\}$, $Sim_{old}(v) = Sim(v) = \{v\}$

$s_1' = s_i$: $Sim_{old}(s_i) = Sim(s_i) = \{s_1, s_2, s_3\}$

choose next state $s_1' = u_1$ with $Sim_{old}(u_1) \neq Sim(u_1)$:

Example: HHK-algorithm



initially:

$Sim_{old}(t) = \bot$ for all states t

$Sim(s_1) = \{s_1, s_2, s_3\}$

$\vdots$

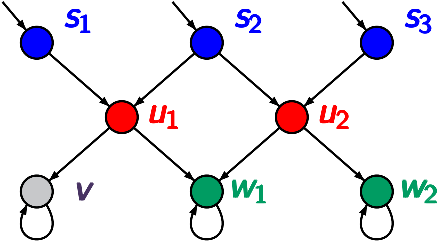
$s_1' = v$: $Sim(u_1) = \{u_1\}$, $Sim_{old}(v) = Sim(v) = \{v\}$

$s_1' = s_i$: $Sim_{old}(s_i) = Sim(s_i) = \{s_1, s_2, s_3\}$

choose next state $s_1' = u_1$ with $Sim_{old}(u_1) \neq Sim(u_1)$:

$Remove(u_1) = S \setminus Pre(Sim(u_1)) =$

Example: HHK-algorithm



initially:

$Sim_{old}(t) = \perp$ for all states t

$Sim(s_1) = \{s_1, s_2, s_3\}$

$\vdots$

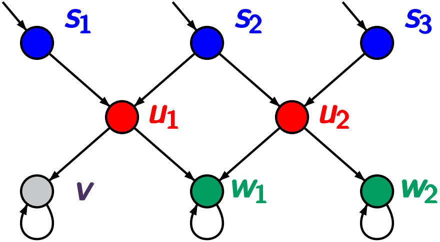
$s_1' = v$: $Sim(u_1) = \{u_1\}$, $Sim_{old}(v) = Sim(v) = \{v\}$

$s_1' = s_i$: $Sim_{old}(s_i) = Sim(s_i) = \{s_1, s_2, s_3\}$

choose next state $s_1' = u_1$ with $Sim_{old}(u_1) \neq Sim(u_1)$:

$Remove(u_1) = S \setminus Pre(Sim(u_1)) = S \setminus \{s_1, s_2\}$

Example: HHK-algorithm



initially:

$Sim_{old}(t) = \bot$ for all states t

$Sim(s_1) = \{s_1, s_2, s_3\}$

$\vdots$

$s_1' = v$: $Sim(u_1) = \{u_1\}$, $Sim_{old}(v) = Sim(v) = \{v\}$

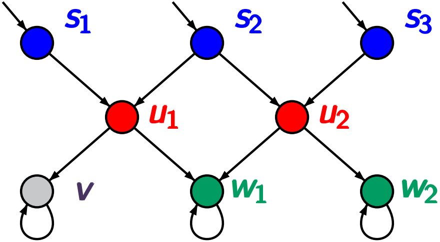
$s_1' = s_i$: $Sim_{old}(s_i) = Sim(s_i) = \{s_1, s_2, s_3\}$

choose next state $s_1' = u_1$ with $Sim_{old}(u_1) \neq Sim(u_1)$:

$Remove(u_1) = S \setminus Pre(Sim(u_1)) = S \setminus \{s_1, s_2\}$

$Sim(s_1) := Sim(s_1) \setminus Remove(u_1) = \{s_1, s_2\}$

Example: HHK-algorithm



initially:

$Sim_{old}(t) = \bot$ for all states t

$Sim(s_1) = \{s_1, s_2, s_3\}$

$\vdots$

$s_1' = v$: $Sim(u_1) = \{u_1\}$, $Sim_{old}(v) = Sim(v) = \{v\}$

$s_1' = s_i$: $Sim_{old}(s_i) = Sim(s_i) = \{s_1, s_2, s_3\}$

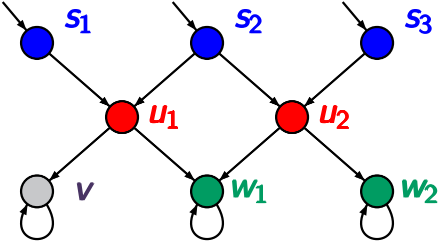
choose next state $s_1' = u_1$ with $Sim_{old}(u_1) \neq Sim(u_1)$:

$Remove(u_1) = S \setminus Pre(Sim(u_1)) = S \setminus \{s_1, s_2\}$

$Sim(s_1) := Sim(s_1) \setminus Remove(u_1) = \{s_1, s_2\}$

$s_1 \longrightarrow u_1$ can't be simulated by any state $t \in Remove(u_1)$

Example: HHK-algorithm



initially:

$Sim_{old}(t) = \bot$ for all states t

$Sim(s_1) = \{s_1, s_2, s_3\}$

$\vdots$

$s_1' = v$: $Sim(u_1) = \{u_1\}$, $Sim_{old}(v) = Sim(v) = \{v\}$

$s_1' = s_i$: $Sim_{old}(s_i) = Sim(s_i) = \{s_1, s_2, s_3\}$

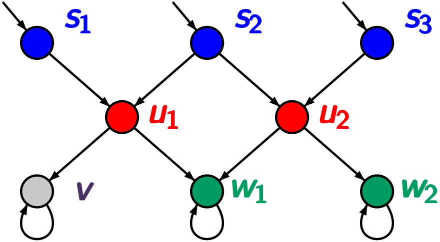
choose next state $s_1' = u_1$ with $Sim_{old}(u_1) \neq Sim(u_1)$:

$Remove(u_1) = S \setminus Pre(Sim(u_1)) = S \setminus \{s_1, s_2\}$

$Sim(s_1) := Sim(s_1) \setminus Remove(u_1) = \{s_1, s_2\}$

$Sim(s_2) := Sim(s_2) \setminus Remove(u_1) = \{s_1, s_2\}$

Example: HHK-algorithm



initially:

$Sim(s_1) = \{s_1, s_2, s_3\}$

⋮

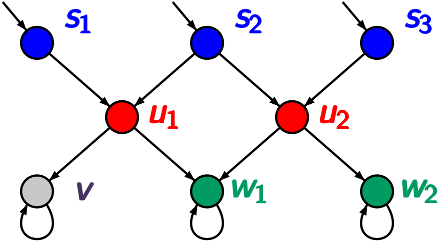
$Sim(u_2) = \{u_1, u_2\}$

v: $Sim(u_1) = \{u_1\}$, $Sim_{old}(v) = Sim(v) = \{v\}$

u_1: $Sim(s_i) = \{s_1, s_2\}$, $i{=}1,2$, $Sim_{old}(u_1) = Sim(u_1) = \{u_1\}$

choose state $s_1' = u_2$:

Example: HHK-algorithm



initially:

$Sim(s_1) = \{s_1, s_2, s_3\}$
$\vdots$
$Sim(u_2) = \{u_1, u_2\}$

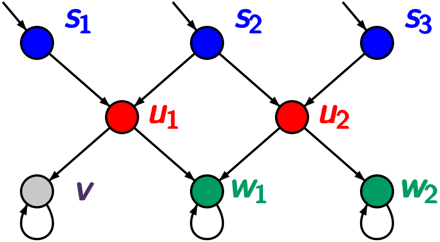
v: $Sim(u_1) = \{u_1\}$, $Sim_{old}(v) = Sim(v) = \{v\}$

u_1: $Sim(s_i) = \{s_1, s_2\}$, $i{=}1,2$, $Sim_{old}(u_1) = Sim(u_1) = \{u_1\}$

choose state $s_1' = u_2$:

$Remove(u_2) = S \setminus Pre(Sim(u_2))$

Example: HHK-algorithm



initially:

$Sim(s_1) = \{s_1, s_2, s_3\}$

$\vdots$

$Sim(u_2) = \{u_1, u_2\}$

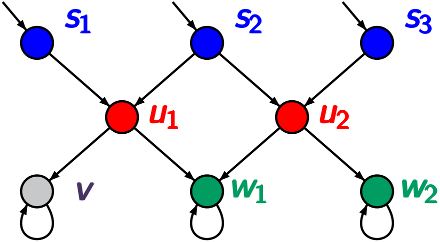
v: $Sim(u_1) = \{u_1\}$, $Sim_{old}(v) = Sim(v) = \{v\}$

u_1: $Sim(s_i) = \{s_1, s_2\}$, $i{=}1,2$, $Sim_{old}(u_1) = Sim(u_1) = \{u_1\}$

choose state $s_1' = u_2$:

$Remove(u_2) = S \setminus Pre(Sim(u_2)) = S \setminus \{s_1, s_2, s_3\}$

Example: HHK-algorithm



initially:

$Sim(s_1) = \{s_1, s_2, s_3\}$

⋮

$Sim(u_2) = \{u_1, u_2\}$

v: $Sim(u_1) = \{u_1\}$, $Sim_{old}(v) = Sim(v) = \{v\}$

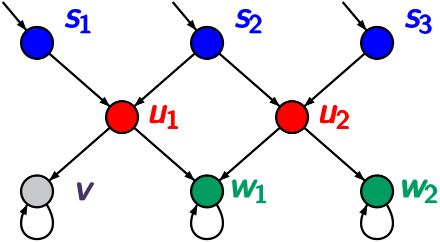
u_1: $Sim(s_i) = \{s_1, s_2\}$, i=1, 2, $Sim_{old}(u_1) = Sim(u_1) = \{u_1\}$

choose state $s'_1 = u_2$:

$Remove(u_2) = S \setminus Pre(Sim(u_2)) = S \setminus \{s_1, s_2, s_3\}$

$Sim(s_i) := Sim(s_2) \setminus Remove(u_2) = \{s_1, s_2\}, \ i = 1, 2$

Example: HHK-algorithm



initially:

$Sim(s_1) = \{s_1, s_2, s_3\}$

$\vdots$

$Sim(u_2) = \{u_1, u_2\}$

v: $Sim(u_1) = \{u_1\}$, $Sim_{old}(v) = Sim(v) = \{v\}$

u_1: $Sim(s_i) = \{s_1, s_2\}$, $i{=}1, 2$, $Sim_{old}(u_1) = Sim(u_1) = \{u_1\}$

choose state $s_1' = u_2$:

$Remove(u_2) = S \setminus Pre(Sim(u_2)) = S \setminus \{s_1, s_2, s_3\}$

$Sim(s_i) := Sim(s_2) \setminus Remove(u_2) = \{s_1, s_2\}$, $i = 1, 2$

$Sim(s_3) := Sim(s_3) \setminus Remove(u_2) = \{s_1, s_2, s_3\}$

HHK-algorithm (second version)

HHK-algorithm (second version)

- an $\mathcal{O}(m \cdot |S|)$-algorithm for computing $\preceq_{\mathcal{T}}$

HHK-algorithm (second version)

- an $\mathcal{O}(\boldsymbol{m} \cdot |S|)$-algorithm for computing $\preceq_{\mathcal{T}}$
- relies on the techniques sketched so far

HHK-algorithm (second version)

- an $\mathcal{O}(m \cdot |S|)$-algorithm for computing $\preceq_{\mathcal{T}}$
- relies on the techniques sketched so far
- but avoids the explicit use of the "old" simulator sets $Sim_{old}(s)$,

HHK-algorithm (second version)

- an $\mathcal{O}(m \cdot |S|)$-algorithm for computing $\preceq_{\mathcal{T}}$
- relies on the techniques sketched so far
- but avoids the explicit use of the "old" simulator sets $Sim_{old}(s)$,
- adds dynamically elements to $Remove(s)$

Loop invariant of the HHK-algorithm

Loop invariant of the HHK-algorithm

(1) $Sim(s_1') \supseteq \{s_2' \in S : s_1' \preceq_{\mathcal{T}} s_2'\}$

Loop invariant of the HHK-algorithm

(1) $Sim(s_1') \supseteq \{s_2' \in S : s_1' \preceq_{\mathcal{T}} s_2'\}$

(2) $Remove(s_1') \subseteq S \setminus Pre(Sim(s_1'))$,

Loop invariant of the HHK-algorithm

(1) $Sim(s_1') \supseteq \{s_2' \in S : s_1' \preceq_{\mathcal{T}} s_2'\}$

(2) $Remove(s_1') \subseteq S \setminus Pre(Sim(s_1'))$,

i.e., for all $s_2 \in Remove(s_1')$:

$$Post(s_2) \cap Sim(s_1') = \varnothing$$

Loop invariant of the HHK-algorithm

(1) $Sim(s_1') \supseteq \{s_2' \in S : s_1' \preceq_{\mathcal{T}} s_2'\}$

(2) $Remove(s_1') \subseteq S \setminus Pre(Sim(s_1'))$,

i.e., for all $s_2 \in Remove(s_1')$:

$$Post(s_2) \cap Sim(s_1') = \varnothing$$

hence: if $s_1 \rightarrow s_1'$ then $s_1 \not\preceq_{\mathcal{T}} s_2$

Loop invariant of the HHK-algorithm

(1) $Sim(s_1') \supseteq \{s_2' \in S : s_1' \preceq_{\mathcal{T}} s_2'\}$

(2) $Remove(s_1') \subseteq S \setminus Pre(Sim(s_1'))$,

i.e., for all $s_2 \in Remove(s_1')$:

$$Post(s_2) \cap Sim(s_1') = \varnothing$$

hence: if $s_1 \to s_1'$ then $s_1 \not\preceq_{\mathcal{T}} s_2$

(3) if $s_2 \in Sim(s_1)$ and $s_1 \to s_1'$ then

- either $Post(s_2) \cap Sim(s_1') \neq \varnothing$
- or $s_2 \in Remove(s_1')$

Loop invariant of the HHK-algorithm

(1) $Sim(s_1') \supseteq \{s_2' \in S : s_1' \preceq_{\mathcal{T}} s_2'\}$

(2) $Remove(s_1') \subseteq S \setminus Pre(Sim(s_1'))$

(3) if $s_2 \in Sim(s_1)$ and $s_1 \to s_1'$ then

- either $Post(s_2) \cap Sim(s_1') \neq \varnothing$
- or $s_2 \in Remove(s_1')$

Loop invariant of the HHK-algorithm

(1) $Sim(s_1') \supseteq \{s_2' \in S : s_1' \preceq_{\mathcal{T}} s_2'\}$

(2) $Remove(s_1') \subseteq S \setminus Pre(Sim(s_1'))$

(3) if $s_2 \in Sim(s_1)$ and $s_1 \rightarrow s_1'$ then

- either $Post(s_2) \cap Sim(s_1') \neq \varnothing$
- or $s_2 \in Remove(s_1')$

if $Remove(s_1') = \varnothing$ for all states s_1' then by (3):

$$s_2 \in Sim(s_1) \wedge s_1 \rightarrow s_1' \implies Post(s_2) \cap Sim(s_1') \neq \varnothing$$

(1) $Sim(s_1') \supseteq \{s_2' \in S : s_1' \preceq_{\mathcal{T}} s_2'\}$

(2) $Remove(s_1') \subseteq S \setminus Pre(Sim(s_1'))$

(3) if $s_2 \in Sim(s_1)$ and $s_1 \to s_1'$ then

- either $Post(s_2) \cap Sim(s_1') \neq \varnothing$
- or $s_2 \in Remove(s_1')$

if $Remove(s_1') = \varnothing$ for all states s_1' then by (3):

$$s_2 \in Sim(s_1) \wedge s_1 \to s_1' \Longrightarrow Post(s_2) \cap Sim(s_1') \neq \varnothing$$

hence: $\{(s_1', s_2') : s_2' \in Sim(s_1')\}$ is a simulation

(1) $Sim(s_1') \supseteq \{s_2' \in S : s_1' \preceq_{\mathcal{T}} s_2'\}$

(2) $Remove(s_1') \subseteq S \setminus Pre(Sim(s_1'))$

(3) if $s_2 \in Sim(s_1)$ and $s_1 \rightarrow s_1'$ then

- either $Post(s_2) \cap Sim(s_1') \neq \varnothing$
- or $s_2 \in Remove(s_1')$

if $Remove(s_1') = \varnothing$ for all states s_1' then by (3):

$$s_2 \in Sim(s_1) \wedge s_1 \rightarrow s_1' \implies Post(s_2) \cap Sim(s_1') \neq \varnothing$$

hence: $\{(s_1', s_2') : s_2' \in Sim(s_1')\}$ is a simulation

since $\preceq_{\mathcal{T}}$ is the coarsest simulation, (1) yields:

$$s_2' \in Sim(s_1') \quad \text{iff} \quad s_1' \preceq_{\mathcal{T}} s_2'$$

HHK-algorithm (second version)

HHK-algorithm (second version)


```
FOR ALL states s'_1 DO
        Sim(s'_1) := {s'_2 ∈ S : L(s'_1) = L(s'_2)}
OD
```

FOR ALL states s'_1 DO

$$Sim(s'_1) := \{s'_2 \in S : L(s'_1) = L(s'_2)\}$$

OD

HHK-algorithm (second version)

FOR ALL states s_1' DO

$$Sim(s_1') := \{s_2' \in S : L(s_1') = L(s_2')\}$$

$$Remove(s_1') := S \setminus Pre(Sim(s_1'))$$

OD

HHK-algorithm (second version)


```
FOR ALL states s'_1 DO
        Sim(s'_1) := {s'_2 ∈ S : L(s'_1) = L(s'_2)}
        Remove(s'_1) := S \ Pre(Sim(s'_1))
OD
WHILE there exists a state s'_1 with Remove(s'_1) ≠ ∅ DO
```

HHK-algorithm (second version)


```
FOR ALL states s'_1 DO
        Sim(s'_1) := {s'_2 ∈ S : L(s'_1) = L(s'_2)}
        Remove(s'_1) := S \ Pre(Sim(s'_1))
OD
WHILE there exists a state s'_1 with Remove(s'_1) ≠ ∅ DO
      choose such a state s'_1
```

HHK-algorithm (second version)


```
FOR ALL states s'_1 DO
        Sim(s'_1) := {s'_2 ∈ S : L(s'_1) = L(s'_2)}
        Remove(s'_1) := S \ Pre(Sim(s'_1))
OD
WHILE there exists a state s'_1 with Remove(s'_1) ≠ ∅ DO
      choose such a state s'_1
      FOR ALL s_2 ∈ Remove(s'_1) DO
```

FOR ALL states s'_1 DO

$Sim(s'_1) := \{s'_2 \in S : L(s'_1) = L(s'_2)\}$

$Remove(s'_1) := S \setminus Pre(Sim(s'_1))$

OD

WHILE there exists a state s'_1 with $Remove(s'_1) \neq \varnothing$ DO

choose such a state s'_1

FOR ALL $s_2 \in Remove(s'_1)$ DO

HHK-algorithm (second version)


```
FOR ALL states s'_1 DO
        Sim(s'_1) := {s'_2 ∈ S : L(s'_1) = L(s'_2)}
        Remove(s'_1) := S \ Pre(Sim(s'_1))
OD
WHILE there exists a state s'_1 with Remove(s'_1) ≠ ∅ DO
      choose such a state s'_1
      FOR ALL s_2 ∈ Remove(s'_1) DO
         FOR ALL s_1 ∈ Pre(s'_1) DO
```

FOR ALL states s_1' DO

$Sim(s_1') := \{s_2' \in S : L(s_1') = L(s_2')\}$

$Remove(s_1') := S \setminus Pre(Sim(s_1'))$

OD

WHILE there exists a state s_1' with $Remove(s_1') \neq \varnothing$ DO

choose such a state s_1'

FOR ALL $s_2 \in Remove(s_1')$ DO

FOR ALL $s_1 \in Pre(s_1')$ DO

HHK-algorithm (second version)


```
FOR ALL states s'_1 DO
        Sim(s'_1) := {s'_2 ∈ S : L(s'_1) = L(s'_2)}
        Remove(s'_1) := S \ Pre(Sim(s'_1))
OD
WHILE there exists a state s'_1 with Remove(s'_1) ≠ ∅ DO
      choose such a state s'_1
      FOR ALL s_2 ∈ Remove(s'_1) DO
         FOR ALL s_1 ∈ Pre(s'_1) DO
            IF s_2 ∈ Sim(s_1) THEN
```

HHK-algorithm (second version)

FOR ALL states s_1' DO

$Sim(s_1') := \{s_2' \in S : L(s_1') = L(s_2')\}$

$Remove(s_1') := S \setminus Pre(Sim(s_1'))$

OD

WHILE there exists a state s_1' with $Remove(s_1') \neq \varnothing$ DO

choose such a state s_1'

FOR ALL $s_2 \in Remove(s_1')$ DO

FOR ALL $s_1 \in Pre(s_1')$ DO

IF $s_2 \in Sim(s_1)$ THEN

$Sim(s_1) := Sim(s_1) \setminus \{s_2\}$

HHK-algorithm (second version)

```
FOR ALL states s'_1 DO
        Sim(s'_1) := { s'_2 ∈ S : L(s'_1) = L(s'_2) }
        Remove(s'_1) := S \ Pre(Sim(s'_1))
OD
WHILE there exists a state s'_1 with Remove(s'_1) ≠ ∅ DO
    choose such a state s'_1
    FOR ALL s_2 ∈ Remove(s'_1) DO
      FOR ALL s_1 ∈ Pre(s'_1) DO
        IF s_2 ∈ Sim(s_1) THEN
          Sim(s_1) := Sim(s_1) \ {s_2}
          Remove(s_1) := Remove(s_1) ∪
            { s ∈ Pre(s_2) : Post(s) ∩ Sim(s_1) = ∅ } FI
      OD
    OD
```

HHK-algorithm (second version)

FOR ALL states s_1' DO

$Sim(s_1') := \{s_2' \in S : L(s_1') = L(s_2')\}$

OD $Remove(s_1') := S \setminus Pre(Sim(s_1'))$

WHILE there exists a state s_1' with $Remove(s_1') \neq \varnothing$ DO

choose such a state s_1'

FOR ALL $s_2 \in Remove(s_1')$ DO

FOR ALL $s_1 \in Pre(s_1')$ DO

IF $s_2 \in Sim(s_1)$ THEN

$Sim(s_1) := Sim(s_1) \setminus \{s_2\}$

$Remove(s_1) := Remove(s_1) \cup$

$\{s \in Pre(s_2) : Post(s) \cap Sim(s_1) = \varnothing\}$ FI

OD

OD

DO $Remove(s_1') := \varnothing$

HHK-algorithm (second version)

⋮

choose such a state s_1' with $Remove(s_1') \neq \varnothing$

FOR ALL $s_2 \in Remove(s_1')$ DO

FOR ALL $s_1 \in Pre(s_1')$ DO

    IF $s_2 \in Sim(s_1)$ THEN

        $Sim(s_1) := Sim(s_1) \setminus \{s_2\}$

        $Remove(s_1) := Remove(s_1) \cup$

            $\{s \in Pre(s_2) : Post(s) \cap Sim(s_1) = \varnothing\}$

    FI

...

HHK-algorithm (second version)

⋮

choose such a state s_1' with $Remove(s_1') \neq \varnothing$

FOR ALL $s_2 \in Remove(s_1')$ DO

FOR ALL $s_1 \in Pre(s_1')$ DO

  IF $s_2 \in Sim(s_1)$ THEN

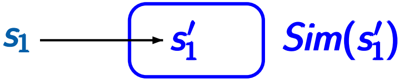
    $Sim(s_1) := Sim(s_1) \setminus \{s_2\}$

    $Remove(s_1) := Remove(s_1) \cup$

      $\{s \in Pre(s_2) : Post(s) \cap Sim(s_1) = \varnothing\}$

  FI

...



HHK-algorithm (second version)

⋮

choose such a state s_1' with $Remove(s_1') \neq \varnothing$

FOR ALL $s_2 \in Remove(s_1')$ DO

FOR ALL $s_1 \in Pre(s_1')$ DO

  IF $s_2 \in Sim(s_1)$ THEN

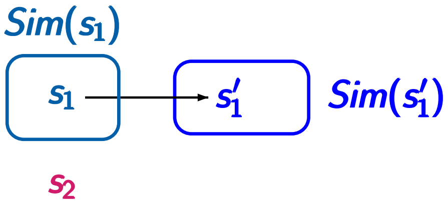
    $Sim(s_1) := Sim(s_1) \setminus \{s_2\}$

    $Remove(s_1) := Remove(s_1) \cup$

      $\{s \in Pre(s_2) : Post(s) \cap Sim(s_1) = \varnothing\}$

  FI

...



HHK-algorithm (second version)

⋮

choose such a state s_1' with $Remove(s_1') \neq \varnothing$

FOR ALL $s_2 \in Remove(s_1')$ DO

FOR ALL $s_1 \in Pre(s_1')$ DO

  IF $s_2 \in Sim(s_1)$ THEN

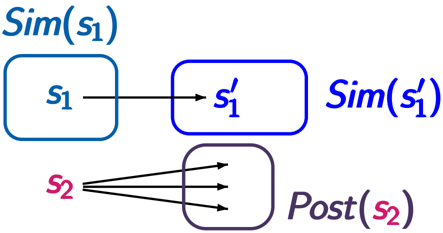
    $Sim(s_1) := Sim(s_1) \setminus \{s_2\}$

    $Remove(s_1) := Remove(s_1) \cup$

      $\{s \in Pre(s_2) : Post(s) \cap Sim(s_1) = \varnothing\}$

  FI

...



HHK-algorithm (second version)

⋮

choose such a state s_1' with $Remove(s_1') \neq \varnothing$

FOR ALL $s_2 \in Remove(s_1')$ DO

FOR ALL $s_1 \in Pre(s_1')$ DO

IF $s_2 \in Sim(s_1)$ THEN

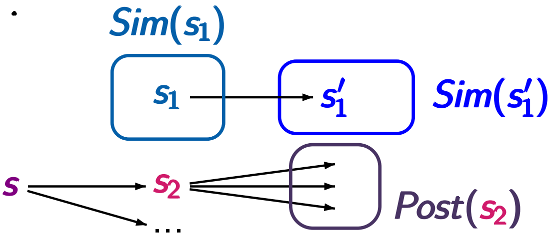
$Sim(s_1) := Sim(s_1) \setminus \{s_2\}$

$Remove(s_1) := Remove(s_1) \cup$

$\{s \in Pre(s_2) : Post(s) \cap Sim(s_1) = \varnothing\}$

FI

...



HHK-algorithm (second version)

⋮

choose such a state s_1' with $Remove(s_1') \neq \varnothing$

FOR ALL $s_2 \in Remove(s_1')$ DO

FOR ALL $s_1 \in Pre(s_1')$ DO

  IF $s_2 \in Sim(s_1)$ THEN

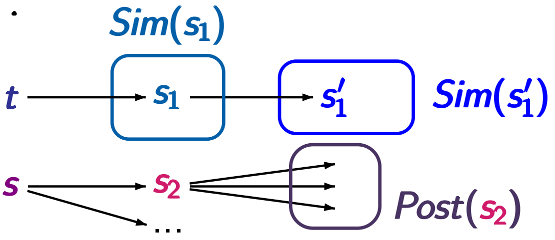
    $Sim(s_1) := Sim(s_1) \setminus \{s_2\}$

    $Remove(s_1) := Remove(s_1) \cup$

      $\{s \in Pre(s_2) : Post(s) \cap Sim(s_1) = \varnothing\}$

  FI

...



HHK-algorithm (second version)

⋮

choose such a state s_1' with $Remove(s_1') \neq \varnothing$

FOR ALL $s_2 \in Remove(s_1')$ DO

FOR ALL $s_1 \in Pre(s_1')$ DO

  IF $s_2 \in Sim(s_1)$ THEN

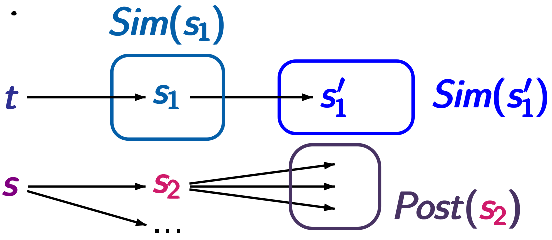
    $Sim(s_1) := Sim(s_1) \setminus \{s_2\}$

    $Remove(s_1) := Remove(s_1) \cup$

      $\{s \in Pre(s_2) : Post(s) \cap Sim(s_1) = \varnothing\}$

  FI

...



$t \not\lesssim_{\mathcal{T}} s$

Termination of the HHK-algorithm

Termination of the HHK-algorithm

for each pair (s_2, s_1') of states: s_2 is inserted in (and removed from) ***Remove***(s_1') at most once

Termination of the HHK-algorithm

for each pair (s_2, s_1') of states: s_2 is inserted in (and removed from) $Remove(s_1')$ at most once

$\vdots$
IF $s_2 \in Sim(s_1)$ THEN
$Sim(s_1) := Sim(s_1) \setminus \{s_2\}$;
$Remove(s_1) := Remove(s_1) \cup$
$\{s \in Pre(s_2) : Post(s) \cap Sim(s_1) = \varnothing\}$
FI
$\vdots$

Termination of the HHK-algorithm

for each pair (s_2, s_1') of states: s_2 is inserted in (and removed from) $Remove(s_1')$ at most once

$$\vdots$$

IF $s_2 \in Sim(s_1)$ THEN

$Sim(s_1) := Sim(s_1) \setminus \{s_2\}$;

$Remove(s_1) := Remove(s_1) \cup$

$\{s \in Pre(s_2) : Post(s) \cap Sim(s_1) = \varnothing\}$

FI

$$\vdots$$

if s is inserted in $Remove(s_1)$ then there exists a state s_2 s.t. ...

for each pair (s_2, s_1') of states: s_2 is inserted in (and removed from) $Remove(s_1')$ at most once

```
⋮
IF  s₂ ∈ Sim(s₁) THEN
    Sim(s₁) := Sim(s₁) \ {s₂};
    Remove(s₁) := Remove(s₁) ∪
       {s ∈ Pre(s₂) : Post(s) ∩ Sim(s₁) = ∅}
FI
⋮
```

if s is inserted in $Remove(s_1)$ then there exists a state s_2 s.t. $s \rightarrow s_2$ and ...

for each pair (s_2, s_1') of states: s_2 is inserted in (and removed from) $Remove(s_1')$ at most once

$$\begin{array}{l}
\vdots \\
\texttt{IF } s_2 \in Sim(s_1) \texttt{ THEN} \\
\quad Sim(s_1) := Sim(s_1) \setminus \{s_2\}; \\
\quad Remove(s_1) := Remove(s_1) \cup \\
\qquad \{s \in Pre(s_2) : Post(s) \cap Sim(s_1) = \varnothing\} \\
\texttt{FI} \\
\vdots
\end{array}$$

if s is inserted in $Remove(s_1)$ then there exists a state s_2 s.t. $s \rightarrow s_2$ and $Post(s) \cap Sim(s_1) = \{s_2\}$ immediately before s_2 has been removed from $Sim(s_1)$

Complexity of the HHK-algorithm

Complexity of the HHK-algorithm

show that the HHK-algorithm can be realized in time:

$$\mathcal{O}(m \cdot |S|)$$

where m = number of edges

S = state space

Complexity of the HHK-algorithm

show that the HHK-algorithm can be realized in time:

$$\mathcal{O}(m \cdot |S|)$$

where m = number of edges

S = state space

$m \geqslant |S|$

Complexity of the HHK-algorithm

show that the HHK-algorithm can be realized in time:

$$\mathcal{O}(m \cdot |S|)$$

where m = number of edges

S = state space

$m \geqslant |S|$

and AP is fixed

Complexity of the initialization

Complexity of the initialization


```
FOR ALL states s'_1 DO
```

$Remove(s'_1) := S \setminus Pre(Sim(s'_1))$

$Sim(s'_1) := \{ s'_2 \in S : L(s'_1) = L(s'_2)$ and if s'_2 is terminal then so is s'_1 $\}$

```
OD
```

Complexity of the initialization

```
FOR ALL states s'_1 DO
```

$Remove(s_1') := S \setminus Pre(Sim(s_1'))$

$Sim(s_1') := \{ \ s_2' \in S : L(s_1') = L(s_2')$ and if s_2' is terminal then so is s_1' $\}$

```
OD
```

time complexity: $\mathcal{O}(|S| \cdot AP) = \mathcal{O}(|S|)$
(as in the bisimulation algorithms)

Complexity of the while-loop

Complexity of the while-loop

WHILE there exists a state s_1' with $Remove(s_1') \neq \varnothing$ DO
  choose such a state s_1'
  FOR ALL $s_2 \in Remove(s_1')$ DO
    FOR ALL $s_1 \in Pre(s_1')$ DO
      IF $s_2 \in Sim(s_1)$ THEN
        $Sim(s_1) := Sim(s_1) \setminus \{s_2\}$
        $Remove(s_1) := Remove(s_1) \cup$
          $\{s \in Pre(s_2) : Post(s) \cap Sim(s_1) = \varnothing\}$
      FI
    OD
  OD
  $Remove(s_1') := \varnothing$
DO

Complexity of the while-loop


```
WHILE there exists a state s1' with Remove(s1') ≠ ∅ DO
  choose such a state s1'
  FOR ALL s2 ∈ Remove(s1') DO
    FOR ALL s1 ∈ Pre(s1') DO
      IF s2 ∈ Sim(s1) THEN
        Sim(s1) := Sim(s1) \ {s2}
        Remove(s1) := Remove(s1) ∪
          {s ∈ Pre(s2) : Post(s) ∩ Sim(s1) = ∅}
      FI
    OD
  OD
  Remove(s1') := ∅
DO
```

Complexity of the while-loop

```
WHILE there exists a state s1' with Remove(s1') ≠ ∅ DO
  choose such a state s1'
  FOR ALL s2 ∈ Remove(s1') DO
    FOR ALL s1 ∈ Pre(s1') DO
      IF s2 ∈ Sim(s1) THEN
        Sim(s1) := Sim(s1) \ {s2}
        Remove(s1) := Remove(s1) ∪
          {s ∈ Pre(s2) : Post(s) ∩ Sim(s1) = ∅}
      FI
    OD
  OD
  Remove(s1') := ∅
DO
```

in total: $\mathcal{O}(m \cdot |S|)$

Complexity of the while-loop

```
WHILE there exists a state s'_1 with Remove(s'_1) ≠ ∅ DO
  choose such a state s'_1
  FOR ALL s_2 ∈ Remove(s'_1) DO
    FOR ALL s_1 ∈ Pre(s'_1) DO
      IF s_2 ∈ Sim(s_1) THEN
        Sim(s_1) := Sim(s_1) \ {s_2}
        Remove(s_1) := Remove(s_1) ∪
          {s ∈ Pre(s_2) : Post(s) ∩ Sim(s_1) = ∅}
      FI
    OD
  OD
  Remove(s'_1) := ∅
DO
```

in total: $\mathcal{O}(m \cdot |S|)$

Complexity of the while-loop

WHILE there exists a state s_1' with $\mathit{Remove}(s_1') \neq \varnothing$ DO
  choose such a state s_1'

FOR ALL $s_2 \in \mathit{Remove}(s_1')$ DO
  FOR ALL $s_1 \in \mathit{Pre}(s_1')$ DO

in total: $\mathcal{O}(m \cdot |S|)$

IF $s_2 \in \mathit{Sim}(s_1)$ THEN
  $\mathit{Sim}(s_1) := \mathit{Sim}(s_1) \setminus \{s_2\}$
  $\mathit{Remove}(s_1) := \mathit{Remove}(s_1) \cup$
    $\{s \in \mathit{Pre}(s_2) : \mathit{Post}(s) \cap \mathit{Sim}(s_1) = \varnothing\}$
FI

in total: $\mathcal{O}(m \cdot |S|)$

  OD
OD

$\mathit{Remove}(s_1') := \varnothing$

DO

Summary: linear vs. branching time

Summary: linear vs. branching time

	linear time	branching time
temporal logic	**LTL**	**CTL**
implementation relation	trace equivalence trace inclusion	bisimulation simulation

Summary: linear vs. branching time

	linear time	branching time
temporal logic	**LTL** **PSPACE**-complete	**CTL** in **P**
implementation relation	trace equivalence trace inclusion	bisimulation simulation

Summary: linear vs. branching time

	linear time	branching time
temporal logic	**LTL** **PSPACE**-complete	**CTL** in **P**
implementation relation	trace equivalence trace inclusion **PSPACE**-complete	bisimulation simulation in **P**

Summary: linear vs. branching time

	linear time	branching time
temporal logic	**LTL** **PSPACE**-complete	**CTL** in **P**
implementation relation	trace equivalence trace inclusion **PSPACE**-complete	bisimulation simulation in **P**

bisimulation $\sim$: $\mathcal{O}(m \cdot \log |S|)$

stutter bisimulation $\approx$ or $\approx^{\mathbf{div}}$: $\mathcal{O}(m \cdot |S|)$

simulation $\preceq$: $\mathcal{O}(m \cdot |S|)$