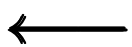
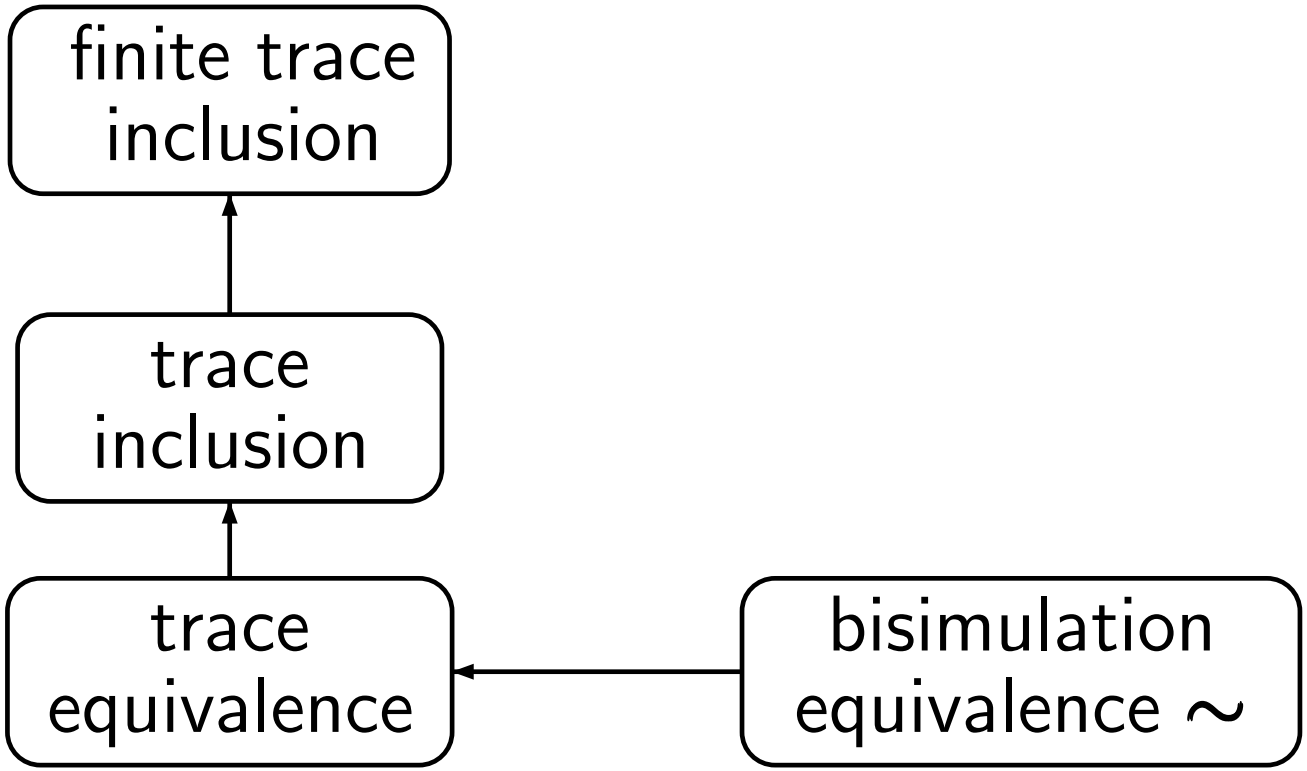
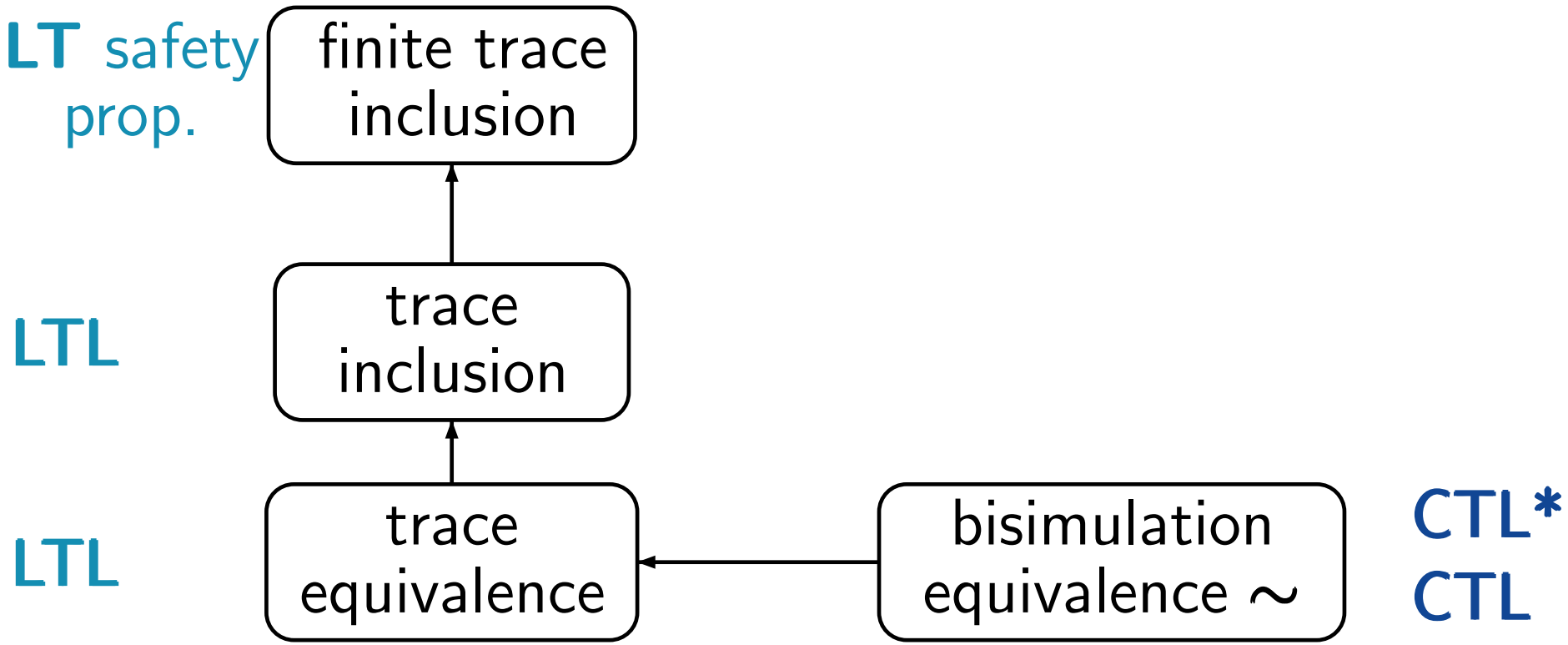
Overview



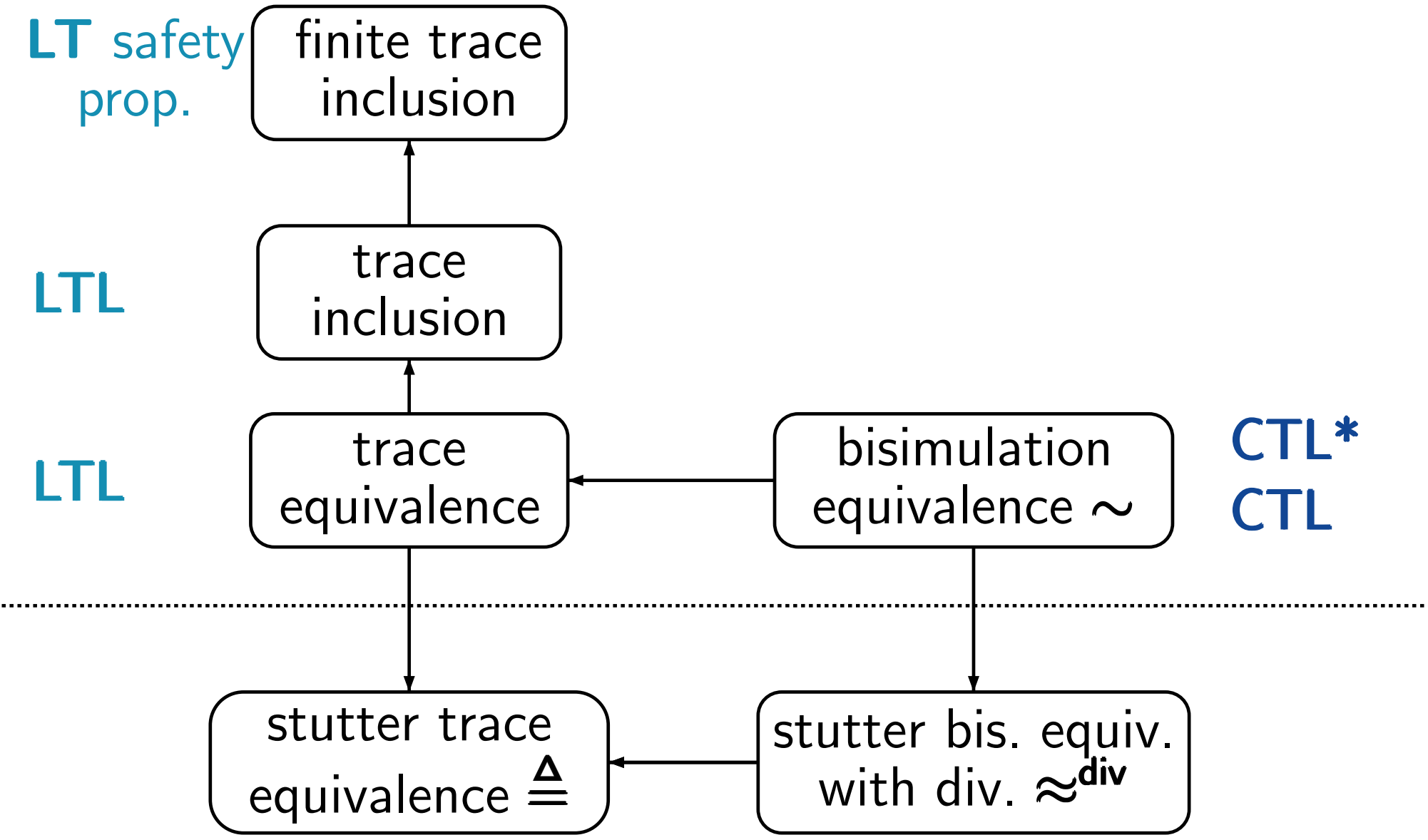
Equivalences and preorders



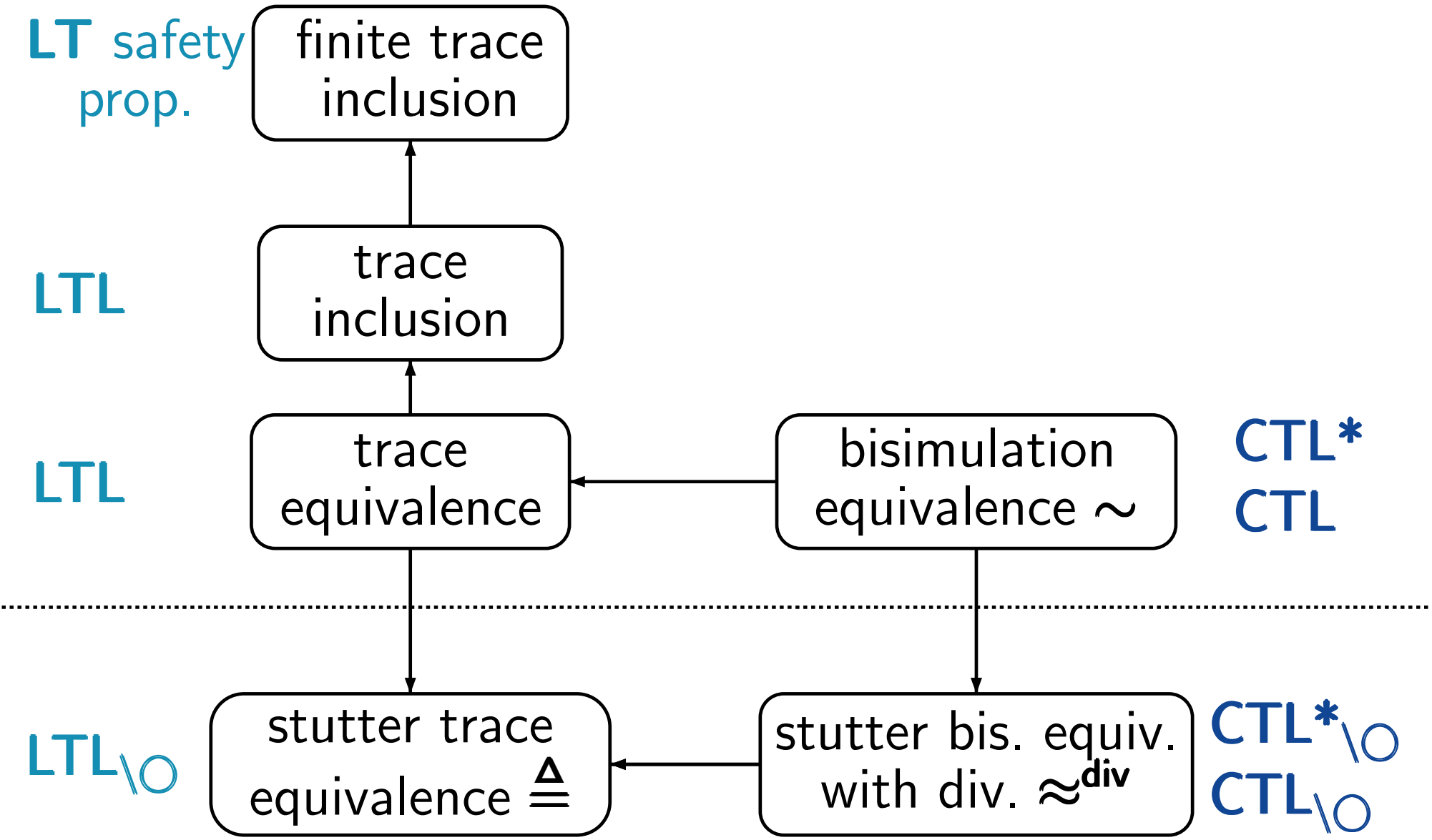
Logical characterizations



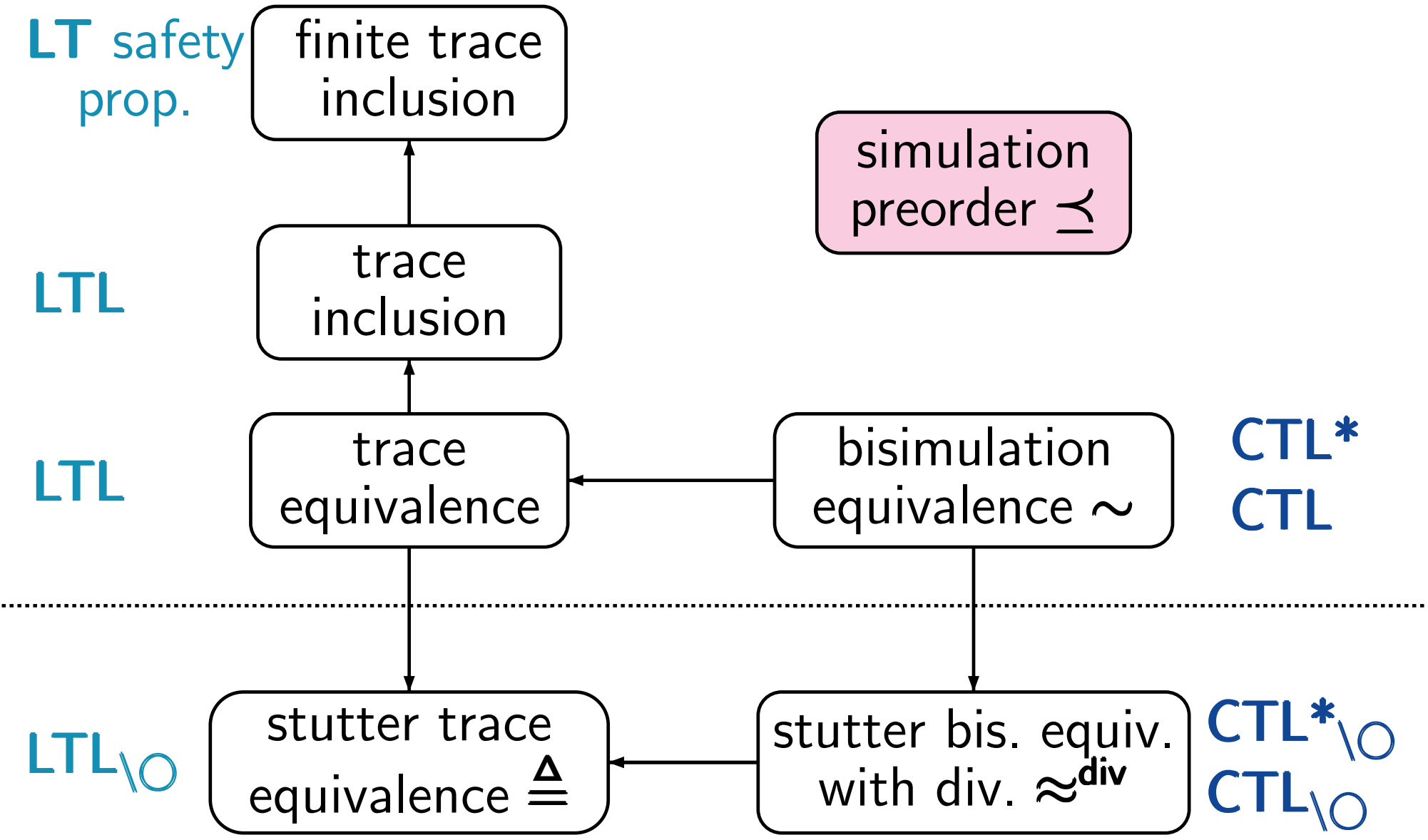
Logical characterizations



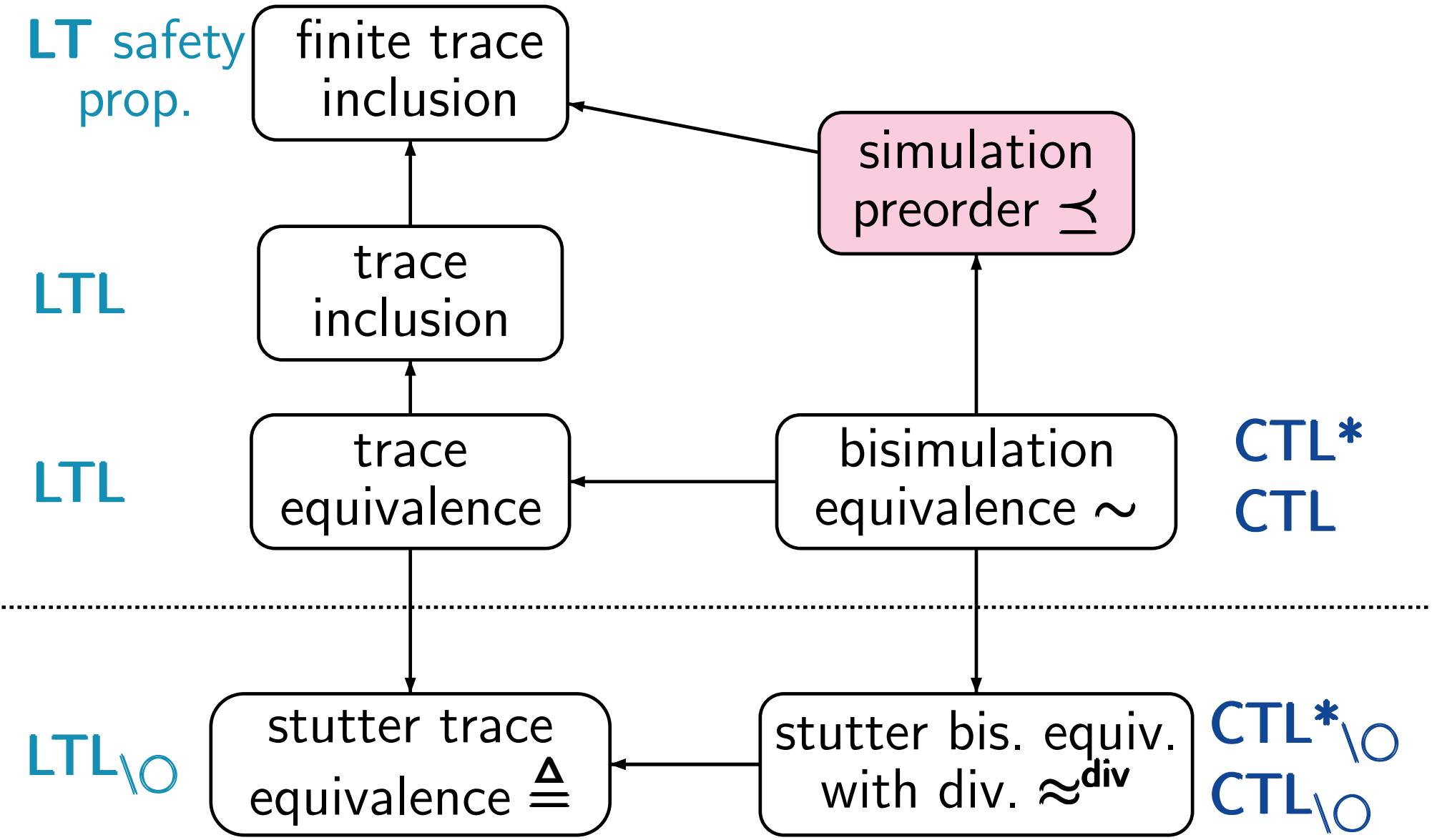
Logical characterizations



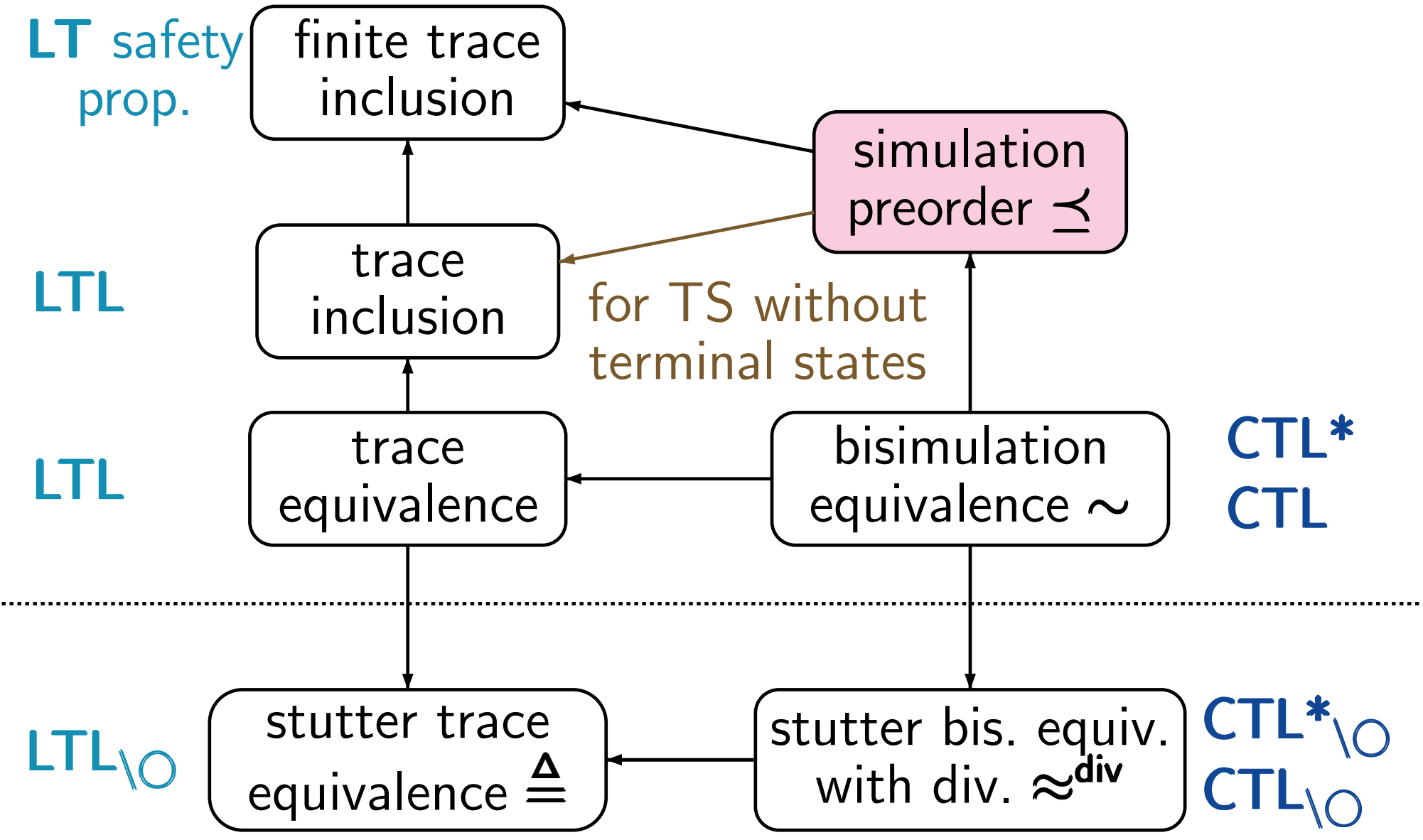
Logical characterizations



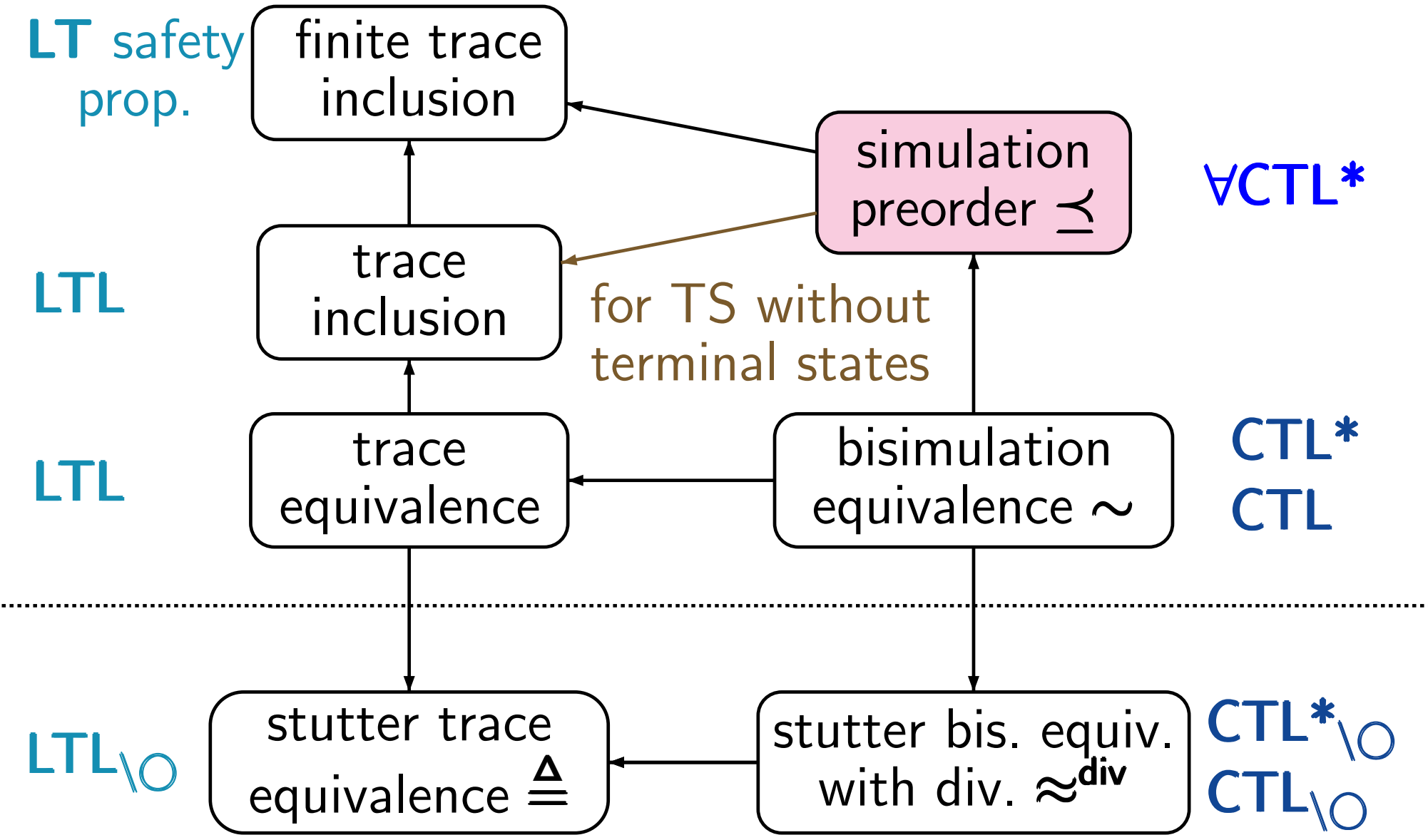
Logical characterizations



Logical characterizations



Logical characterizations



Logical characterization

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

$s_1 \sim_{\mathcal{T}} s_2$ iff s_1, s_2 satisfy the same **CTL*** formulas

iff s_1, s_2 satisfy the same **CTL** formulas

Logical characterization

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

$s_1 \sim_{\mathcal{T}} s_2$ iff s_1, s_2 satisfy the same **CTL*** formulas

iff s_1, s_2 satisfy the same **CTL** formulas

for the simulation preorder $\preceq_{\mathcal{T}}$:

Logical characterization

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

$s_1 \sim_{\mathcal{T}} s_2$ iff s_1, s_2 satisfy the same **CTL*** formulas

iff s_1, s_2 satisfy the same **CTL** formulas

for the simulation preorder $\preceq_{\mathcal{T}}$:

by a sublogic $\mathbb{L}$ of **CTL*** that subsumes **LTL**

Logical characterization

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

$s_1 \sim_{\mathcal{T}} s_2$ iff s_1, s_2 satisfy the same **CTL*** formulas

iff s_1, s_2 satisfy the same **CTL** formulas

for the simulation preorder $\preceq_{\mathcal{T}}$:

by a sublogic $\mathbb{L}$ of **CTL*** that subsumes **LTL**

$s_1 \preceq_{\mathcal{T}} s_2$ iff for all formulas $\Phi \in \mathbb{L}$:

$s_2 \models \Phi$ implies $s_1 \models \Phi$

Logical characterization

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

$s_1 \sim_{\mathcal{T}} s_2$ iff s_1, s_2 satisfy the same **CTL*** formulas

iff s_1, s_2 satisfy the same **CTL** formulas

for the simulation preorder $\preceq_{\mathcal{T}}$:

by a sublogic $\mathbb{L}$ of **CTL*** that subsumes **LTL**

$s_1 \preceq_{\mathcal{T}} s_2$ iff for all formulas $\Phi \in \mathbb{L}$:

$s_2 \models \Phi$ implies $s_1 \models \Phi$

observation: $\mathbb{L}$ cannot be closed under negation

The universal fragment ∀CTL* of CTL*

CTL* formulas in positive normal form, without ∃

Syntax of ∀CTL*

∀CTL* state formulas:

$$\Phi ::= \mathit{true} \mid \mathit{false} \mid a \mid \neg a \mid \Phi_1 \wedge \Phi_2 \mid \Phi_1 \vee \Phi_2 \mid \forall\varphi$$

∀CTL* path formulas:

$$\varphi ::= \Phi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \bigcirc\varphi \mid \varphi_1 \,\mathsf{U}\, \varphi_2 \mid \varphi_1 \,\mathsf{W}\, \varphi_2$$

Syntax of ∀CTL*

∀CTL* state formulas:

$$\Phi ::= \mathit{true} \mid \mathit{false} \mid a \mid \neg a \mid \Phi_1 \wedge \Phi_2 \mid \Phi_1 \vee \Phi_2 \mid \forall\varphi$$

∀CTL* path formulas:

$$\varphi ::= \Phi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \bigcirc\varphi \mid \varphi_1 \,\mathsf{U}\, \varphi_2 \mid \varphi_1 \,\mathsf{W}\, \varphi_2$$

eventually: $\Diamond\varphi \stackrel{\text{def}}{=} \mathit{true}\,\mathsf{U}\,\varphi$

Syntax of ∀CTL*

∀CTL* state formulas:

$$\Phi ::= \mathit{true} \mid \mathit{false} \mid a \mid \neg a \mid \Phi_1 \wedge \Phi_2 \mid \Phi_1 \vee \Phi_2 \mid \forall\varphi$$

∀CTL* path formulas:

$$\varphi ::= \Phi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \bigcirc\varphi \mid \varphi_1 \,\mathbf{U}\, \varphi_2 \mid \varphi_1 \,\mathbf{W}\, \varphi_2$$

eventually: $\Diamond\varphi \stackrel{\text{def}}{=} \mathit{true}\,\mathbf{U}\,\varphi$

always: $\Box\varphi \stackrel{\text{def}}{=} \varphi\,\mathbf{W}\,\mathit{false}$

Embedding of LTL in ∀CTL*

∀CTL* state formulas:

$$\Phi \;::=\; \mathit{true} \mid \mathit{false} \mid a \mid \neg a \mid \Phi_1 \wedge \Phi_2 \mid \Phi_1 \vee \Phi_2 \mid \forall\varphi$$

∀CTL* path formulas:

$$\varphi \;::=\; \Phi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \bigcirc\varphi \mid \varphi_1 \,\mathbf{U}\, \varphi_2 \mid \varphi_1 \,\mathbf{W}\, \varphi_2$$

for all **LTL** formulas φ in **PNF**:

Embedding of LTL in ∀CTL*

∀CTL* state formulas:

$$\Phi \;::=\; \mathit{true} \mid \mathit{false} \mid a \mid \neg a \mid \Phi_1 \wedge \Phi_2 \mid \Phi_1 \vee \Phi_2 \mid \forall\varphi$$

∀CTL* path formulas:

$$\varphi \;::=\; \Phi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \bigcirc\varphi \mid \varphi_1 \,\mathbf{U}\, \varphi_2 \mid \varphi_1 \,\mathbf{W}\, \varphi_2$$

for all **LTL** formulas φ in **PNF**:

$$s \models_{\mathrm{LTL}} \varphi \quad \text{iff} \quad s \models_{\forall\mathrm{CTL}^*} \forall\varphi$$

Embedding of LTL in ∀CTL*

∀CTL* state formulas:

$$\Phi \;::=\; \mathit{true} \mid \mathit{false} \mid a \mid \neg a \mid \Phi_1 \wedge \Phi_2 \mid \Phi_1 \vee \Phi_2 \mid \forall\varphi$$

∀CTL* path formulas:

$$\varphi \;::=\; \Phi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \bigcirc\varphi \mid \varphi_1 \,\mathsf{U}\, \varphi_2 \mid \varphi_1 \,\mathsf{W}\, \varphi_2$$

for all **LTL** formulas φ in **PNF**:

$$s \models_{\mathsf{LTL}} \varphi \quad \text{iff} \quad s \models_{\forall\mathsf{CTL}^*} \forall\varphi$$

but $\forall\Diamond\forall\Box a$ cannot be expressed in **LTL**

The universal fragments of CTL* and CTL

syntax of ∀**CTL***:

$$\Phi ::= \mathit{true} \mid \mathit{false} \mid a \mid \neg a \mid \Phi_1 \wedge \Phi_2 \mid \Phi_1 \vee \Phi_2 \mid \forall\varphi$$

$$\varphi ::= \Phi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \bigcirc\varphi \mid \varphi_1 \,\mathbf{U}\, \varphi_2 \mid \varphi_1 \,\mathbf{W}\, \varphi_2$$

∀**CTL**: sublogic of ∀**CTL***

- no Boolean operators for paths formulas
- the arguments of the temporal modalities $\bigcirc$, **U** and **W** are state formulas

The universal fragments of CTL* and CTL

syntax of ∀CTL*:

$$\Phi ::= \mathit{true} \mid \mathit{false} \mid a \mid \neg a \mid \Phi_1 \wedge \Phi_2 \mid \Phi_1 \vee \Phi_2 \mid \forall\varphi$$

$$\varphi ::= \Phi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \bigcirc\varphi \mid \varphi_1 \,\mathsf{U}\, \varphi_2 \mid \varphi_1 \,\mathsf{W}\, \varphi_2$$

∀CTL: sublogic of ∀CTL*

syntax of ∀CTL:

$$\Phi ::= \mathit{true} \mid \mathit{false} \mid a \mid \neg a \mid \Phi_1 \wedge \Phi_2 \mid \Phi_1 \vee \Phi_2 \mid \forall\bigcirc\Phi \mid \forall(\Phi_1 \,\mathsf{U}\, \Phi_2) \mid \forall(\Phi_1 \,\mathsf{W}\, \Phi_2)$$

Logical characterization of simulation

Logical characterization of simulation

Let $\mathcal{T}$ be a finite TS without terminal states. Then, for all states s_1 and s_2 in $\mathcal{T}$, the following statements are equivalent:

Logical characterization of simulation

Let $\mathcal{T}$ be a finite TS without terminal states. Then, for all states s_1 and s_2 in $\mathcal{T}$, the following statements are equivalent:

(1) $s_1 \preceq_{\mathcal{T}} s_2$

Logical characterization of simulation

Let $\mathcal{T}$ be a finite TS without terminal states. Then, for all states s_1 and s_2 in $\mathcal{T}$, the following statements are equivalent:

(1) $s_1 \preceq_{\mathcal{T}} s_2$

(2) for all $\forall$CTL state formulas Φ:

if $s_2 \models \Phi$ then $s_1 \models \Phi$

Logical characterization of simulation

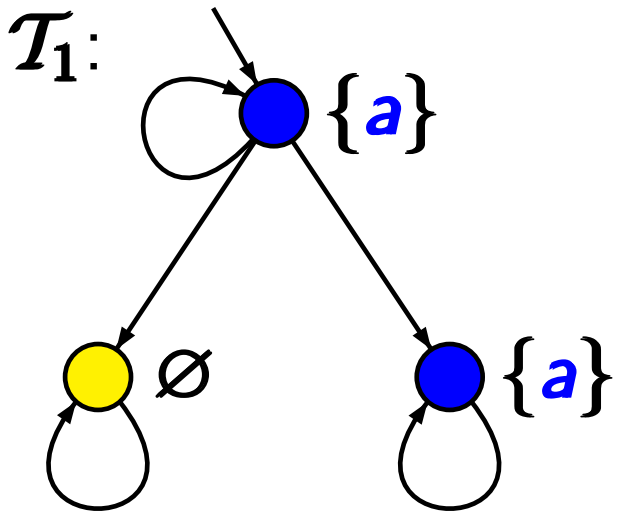
Let $\mathcal{T}$ be a finite TS without terminal states. Then, for all states s_1 and s_2 in $\mathcal{T}$, the following statements are equivalent:

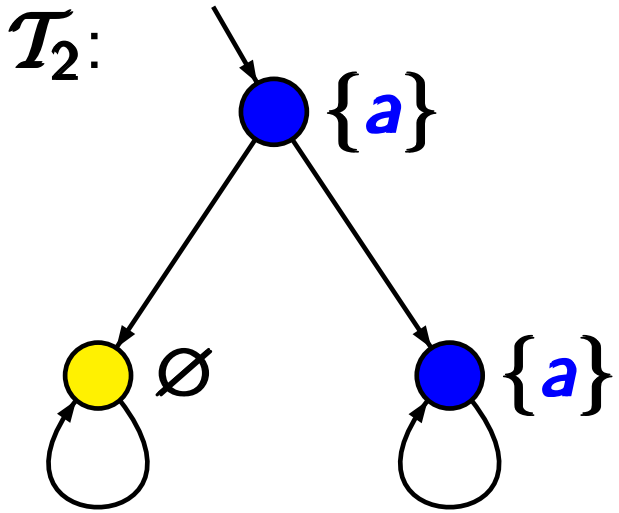
(1) $s_1 \preceq_{\mathcal{T}} s_2$

(2) for all $\forall$CTL state formulas Φ:
if $s_2 \models \Phi$ then $s_1 \models \Phi$

(3) for all $\forall$CTL* state formulas Φ:
if $s_2 \models \Phi$ then $s_1 \models \Phi$

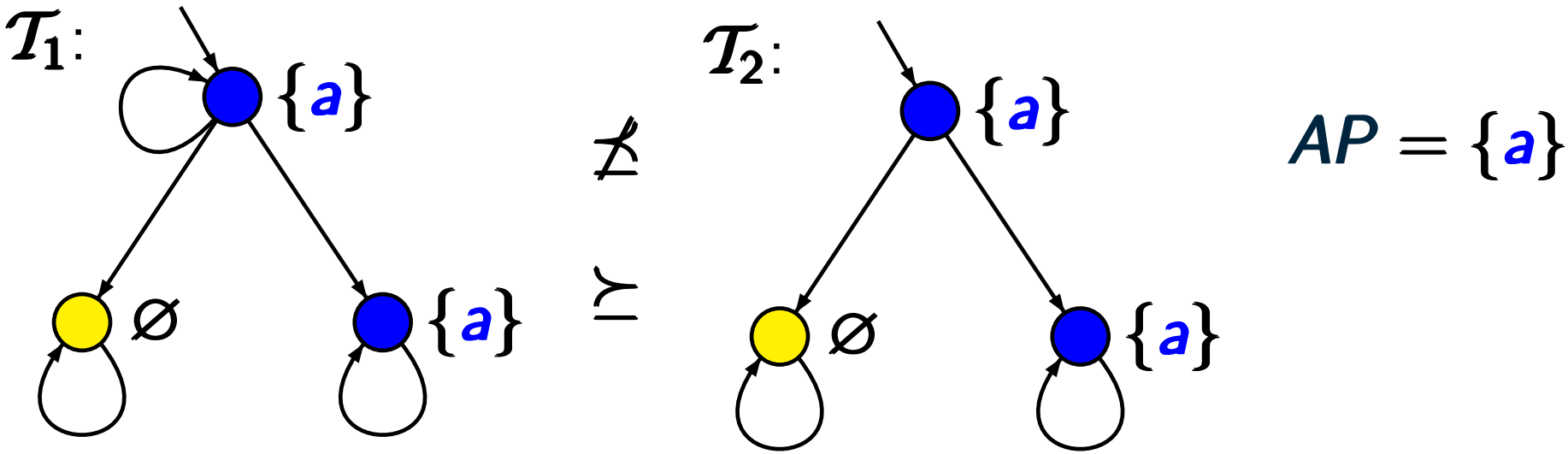
∀CTL and simulation



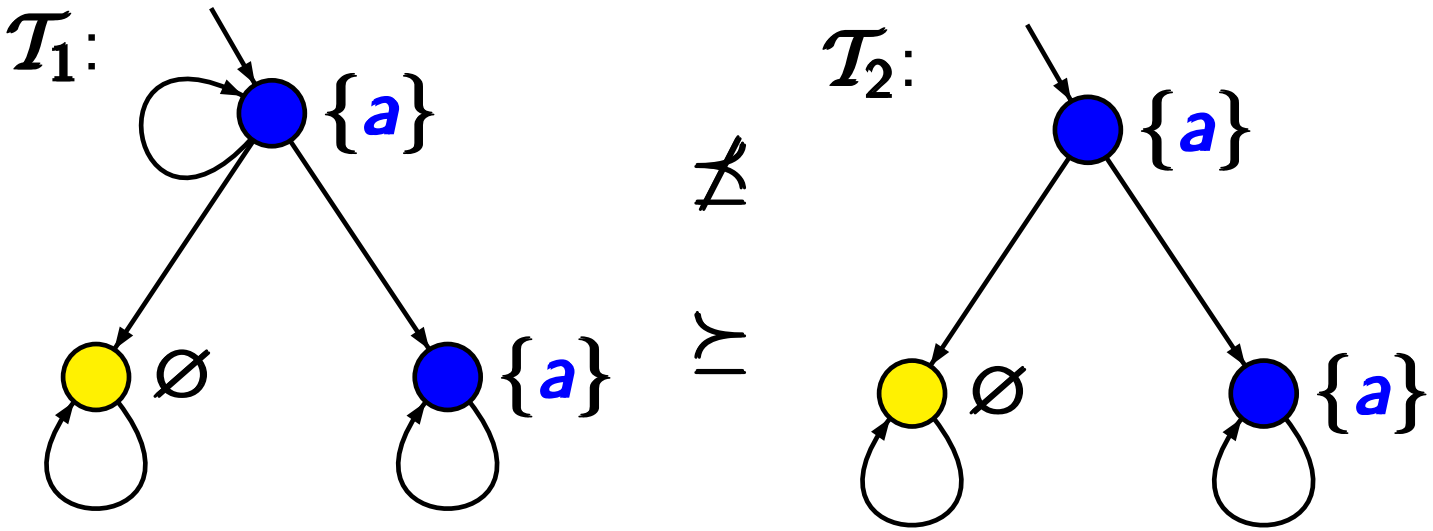


$AP = \{a\}$

∀CTL and simulation



∀CTL and simulation

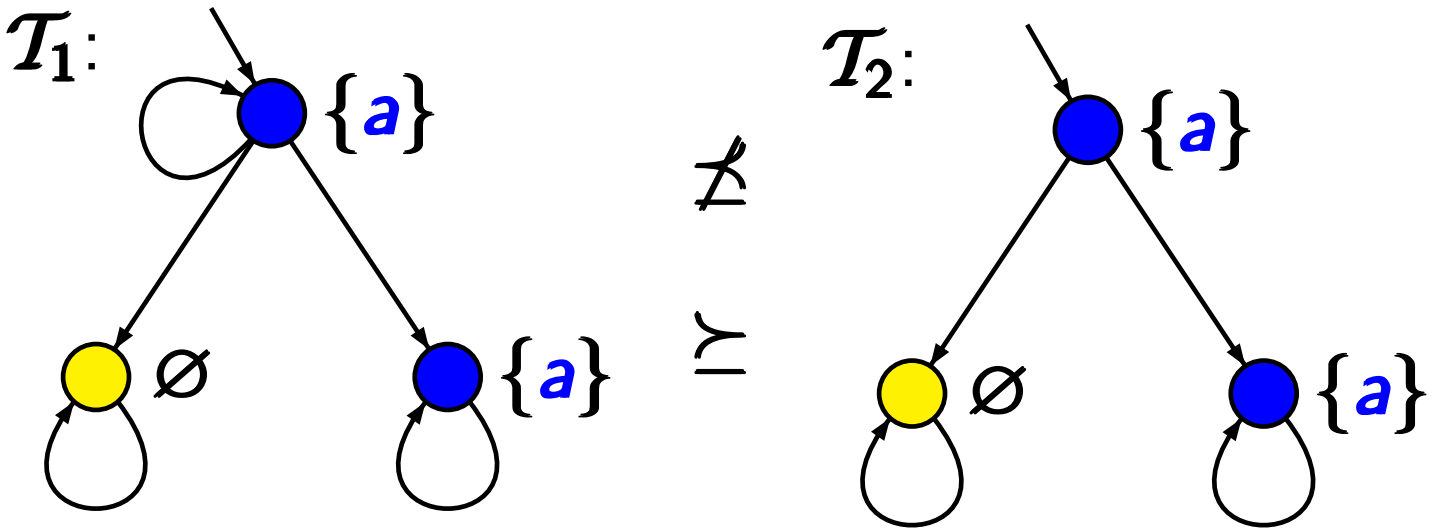


$AP = \{a\}$

e.g., $\mathcal{T}_1 \not\models \forall\bigcirc(\forall\bigcirc\neg a \vee \forall\bigcirc a)$

$\mathcal{T}_2 \models \forall\bigcirc(\forall\bigcirc\neg a \vee \forall\bigcirc a)$

∀CTL and simulation



$AP = \{a\}$

e.g., $\mathcal{T}_1 \not\models \forall\bigcirc(\forall\bigcirc\neg a \vee \forall\bigcirc a)$

$\mathcal{T}_2 \models \forall\bigcirc(\forall\bigcirc\neg a \vee \forall\bigcirc a)$

$\mathcal{T}_1 \not\models \forall\Diamond(\forall\Box\neg a \vee \forall\Box a)$

$\mathcal{T}_2 \models \forall\Diamond(\forall\Box\neg a \vee \forall\Box a)$

∀CTL/∀CTL* and the simulation preorder

For finite TS without terminal states, the following statements are equivalent:

(1) $s_1 \preceq_{\mathcal{T}} s_2$

(2) for all ∀CTL formulas Φ: $s_2 \models \Phi$ implies $s_1 \models \Phi$

(3) for all ∀CTL* formulas Φ: $s_2 \models \Phi$ implies $s_1 \models \Phi$

∀CTL/∀CTL* and the simulation preorder

For finite TS without terminal states, the following statements are equivalent:

(1) $s_1 \preceq_{\mathcal{T}} s_2$

(2) for all ∀CTL formulas Φ: $s_2 \models \Phi$ implies $s_1 \models \Phi$

(3) for all ∀CTL* formulas Φ: $s_2 \models \Phi$ implies $s_1 \models \Phi$

(3) $\Longrightarrow$ (2): obvious as ∀CTL is a sublogic of ∀CTL*

∀CTL/∀CTL* and the simulation preorder

For finite TS without terminal states, the following statements are equivalent:

(1) $s_1 \preceq_T s_2$

(2) for all ∀CTL formulas Φ: $s_2 \models \Phi$ implies $s_1 \models \Phi$

(3) for all ∀CTL* formulas Φ: $s_2 \models \Phi$ implies $s_1 \models \Phi$

(3) $\Longrightarrow$ (2): obvious as ∀CTL is a sublogic of ∀CTL*

(1) $\Longrightarrow$ (3): holds for arbitrary (possibly infinite) TS without terminal states

∀CTL/∀CTL* and the simulation preorder

For finite TS without terminal states, the following statements are equivalent:

(1) $s_1 \preceq_T s_2$

(2) for all ∀CTL formulas Φ: $s_2 \models \Phi$ implies $s_1 \models \Phi$

(3) for all ∀CTL* formulas Φ: $s_2 \models \Phi$ implies $s_1 \models \Phi$

(3) $\Longrightarrow$ (2): obvious as ∀CTL is a sublogic of ∀CTL*

(1) $\Longrightarrow$ (3): holds for arbitrary (possibly infinite) TS without terminal states

proof by structural induction

∀CTL/∀CTL* and the simulation preorder

For finite TS without terminal states, the following statements are equivalent:

(1) $s_1 \preceq_{\mathcal{T}} s_2$

(2) for all ∀CTL formulas Φ: $s_2 \models \Phi$ implies $s_1 \models \Phi$

(3) for all ∀CTL* formulas Φ: $s_2 \models \Phi$ implies $s_1 \models \Phi$

(1) $\Longrightarrow$ (3): show by structural induction:

∀CTL/∀CTL* and the simulation preorder

For finite TS without terminal states, the following statements are equivalent:

(1) $s_1 \preceq_{\mathcal{T}} s_2$

(2) for all ∀CTL formulas Φ: $s_2 \models \Phi$ implies $s_1 \models \Phi$

(3) for all ∀CTL* formulas Φ: $s_2 \models \Phi$ implies $s_1 \models \Phi$

(1) $\Longrightarrow$ (3): show by structural induction:

(i) for all ∀CTL* state formulas Φ and states s_1, s_2:
if $s_1 \preceq_{\mathcal{T}} s_2$ and $s_2 \models \Phi$ then $s_1 \models \Phi$

∀CTL/∀CTL* and the simulation preorder

For finite TS without terminal states, the following statements are equivalent:

(1) $s_1 \preceq_{\mathcal{T}} s_2$

(2) for all ∀CTL formulas Φ: $s_2 \models \Phi$ implies $s_1 \models \Phi$

(3) for all ∀CTL* formulas Φ: $s_2 \models \Phi$ implies $s_1 \models \Phi$

(1) $\Longrightarrow$ (3): show by structural induction:

(i) for all ∀CTL* state formulas Φ and states s_1, s_2:
if $s_1 \preceq_{\mathcal{T}} s_2$ and $s_2 \models \Phi$ then $s_1 \models \Phi$

(ii) for all ∀CTL* path formulas φ and paths π_1, π_2:
if $\pi_1 \preceq_{\mathcal{T}} \pi_2$ and $\pi_2 \models \varphi$ then $\pi_1 \models \varphi$

∀CTL/∀CTL* and the simulation preorder

For finite TS without terminal states, the following statements are equivalent:

(1) $s_1 \preceq_{\mathcal{T}} s_2$

(2) for all ∀CTL formulas Φ: $s_2 \models \Phi$ implies $s_1 \models \Phi$

(3) for all ∀CTL* formulas Φ: $s_2 \models \Phi$ implies $s_1 \models \Phi$

(2) $\Longrightarrow$ (1):

∀CTL/∀CTL* and the simulation preorder

For finite TS without terminal states, the following statements are equivalent:

(1) $s_1 \preceq_{\mathcal{T}} s_2$

(2) for all ∀CTL formulas Φ: $s_2 \models \Phi$ implies $s_1 \models \Phi$

(3) for all ∀CTL* formulas Φ: $s_2 \models \Phi$ implies $s_1 \models \Phi$

(2) $\Longrightarrow$ (1): show that

$$\mathcal{R} = \{ (s_1, s_2) : \text{for all ∀CTL formulas } \Phi: s_2 \models \Phi \text{ implies } s_1 \models \Phi \}$$

is a simulation.

∀CTL/∀CTL* and the simulation preorder

For finite TS without terminal states, the following statements are equivalent:

(1) $s_1 \preceq_{\mathcal{T}} s_2$

(2) for all ∀CTL formulas Φ: $s_2 \models \Phi$ implies $s_1 \models \Phi$

(3) for all ∀CTL* formulas Φ: $s_2 \models \Phi$ implies $s_1 \models \Phi$

(2) $\Longrightarrow$ (1): show that for finite TS:

$$\mathcal{R} = \{ (s_1, s_2) : \text{for all ∀CTL formulas } \Phi: s_2 \models \Phi \text{ implies } s_1 \models \Phi \}$$

is a simulation.

The existential fragment ∃CTL* of CTL*

The existential fragment ∃CTL* of CTL*

dual to ∀**CTL***, i.e., **CTL*** formulas in **PNF**, without ∀

The existential fragment ∃CTL* of CTL*

∃CTL* (state) formulas:

$$\psi \quad ::= \quad \mathit{true} \mid \mathit{false} \mid a \mid \neg a \mid \psi_1 \wedge \psi_2 \mid \psi_1 \vee \psi_2 \mid \exists \varphi$$

∃CTL* path formulas:

$$\varphi \quad ::= \quad \psi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \bigcirc \varphi \mid \varphi_1 \,\mathbf{U}\, \varphi_2 \mid \varphi_1 \,\mathbf{W}\, \varphi_2$$

The existential fragment ∃CTL* of CTL*

∃CTL* (state) formulas:

$$\Psi \quad ::= \quad \mathit{true} \mid \mathit{false} \mid a \mid \neg a \mid \Psi_1 \wedge \Psi_2 \mid \Psi_1 \vee \Psi_2 \mid \exists \varphi$$

∃CTL* path formulas:

$$\varphi \quad ::= \quad \Psi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \bigcirc \varphi \mid \varphi_1 \,\mathsf{U}\, \varphi_2 \mid \varphi_1 \,\mathsf{W}\, \varphi_2$$

analogous: ∃CTL

Duality of ∀CTL* and ∃CTL*

∃CTL* (state) formulas:

$$\Psi \quad ::= \quad \mathit{true} \mid \mathit{false} \mid a \mid \neg a \mid \Psi_1 \wedge \Psi_2 \mid \Psi_1 \vee \Psi_2 \mid \exists \varphi$$

∃CTL* path formulas:

$$\varphi \quad ::= \quad \Psi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \bigcirc \varphi \mid \varphi_1 \,\mathsf{U}\, \varphi_2 \mid \varphi_1 \,\mathsf{W}\, \varphi_2$$

analogous: ∃CTL

For each ∀CTL* formula Φ there is a ∃CTL* formula Ψ s.t. $\Phi \equiv \neg\Psi$

Duality of ∀CTL* and ∃CTL*

∃CTL* (state) formulas:

$$\Psi \quad ::= \quad \mathit{true} \mid \mathit{false} \mid a \mid \neg a \mid \Psi_1 \wedge \Psi_2 \mid \Psi_1 \vee \Psi_2 \mid \exists \varphi$$

∃CTL* path formulas:

$$\varphi \quad ::= \quad \Psi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \bigcirc \varphi \mid \varphi_1 \,\mathsf{U}\, \varphi_2 \mid \varphi_1 \,\mathsf{W}\, \varphi_2$$

analogous: ∃CTL

For each ∀CTL* formula Φ there is a ∃CTL* formula Ψ s.t. $\Phi \equiv \neg\Psi$ (and vice versa)

Duality of ∀CTL* and ∃CTL*

∃CTL* (state) formulas:

$$\Psi \quad ::= \quad \mathit{true} \mid \mathit{false} \mid a \mid \neg a \mid \Psi_1 \wedge \Psi_2 \mid \Psi_1 \vee \Psi_2 \mid \exists \varphi$$

∃CTL* path formulas:

$$\varphi \quad ::= \quad \Psi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \bigcirc \varphi \mid \varphi_1 \,\mathsf{U}\, \varphi_2 \mid \varphi_1 \,\mathsf{W}\, \varphi_2$$

analogous: ∃CTL

For each ∀CTL* formula Φ there is a ∃CTL* formula Ψ s.t. $\Phi \equiv \neg\Psi$ (and vice versa)

For each ∀CTL formula Φ there is a ∃CTL formula Ψ s.t. $\Phi \equiv \neg\Psi$ (and vice versa)

Logical characterization of simulation

Logical characterization of simulation

If s_1 and s_2 are states in a finite TS then the following statements are equivalent:

(1) $s_1 \preceq_T s_2$

(2) for all $\forall$CTL formulas Φ:
if $s_2 \models \Phi$ then $s_1 \models \Phi$

(3) for all $\forall$CTL* formulas Φ:
if $s_2 \models \Phi$ then $s_1 \models \Phi$

Logical characterization of simulation

If s_1 and s_2 are states in a finite TS then the following statements are equivalent:

(1) $s_1 \preceq_T s_2$

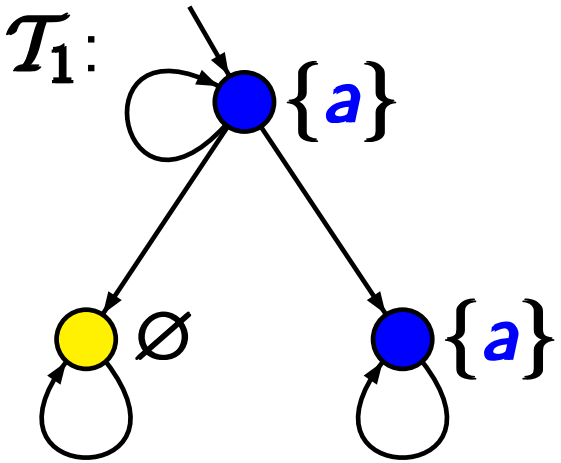
(2∀) for all ∀CTL formulas Φ:
if $s_2 \models \Phi$ then $s_1 \models \Phi$

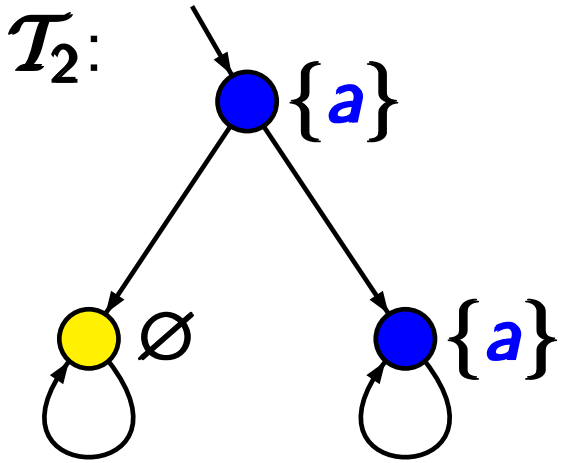
(3∀) for all ∀CTL* formulas Φ:
if $s_2 \models \Phi$ then $s_1 \models \Phi$

(2∃) for all ∃CTL formulas Ψ:
if $s_1 \models \Psi$ then $s_2 \models \Psi$

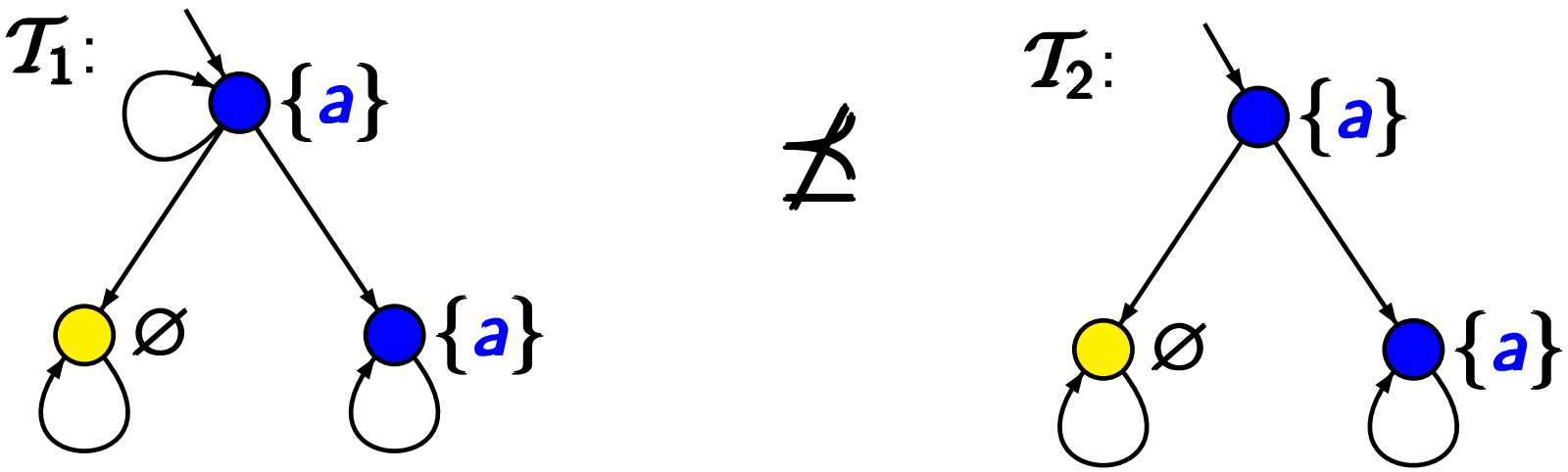
(3∃) for all ∃CTL formulas Ψ:
if $s_1 \models \Psi$ then $s_2 \models \Psi$

Example: ∀CTL/∃CTL and simulation

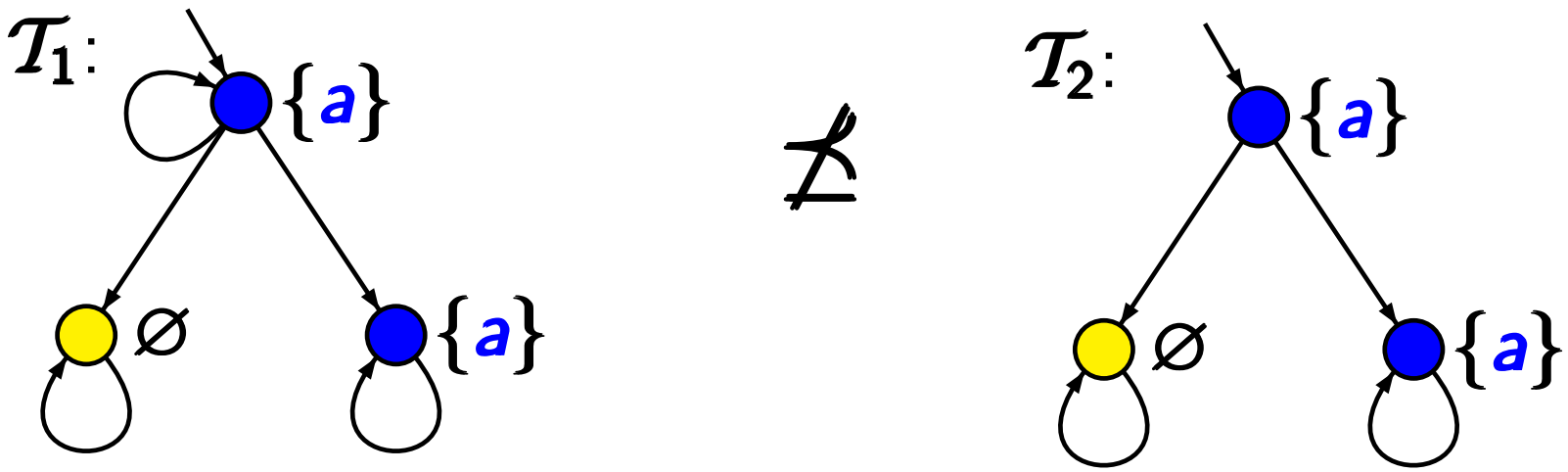




Example: ∀CTL/∃CTL and simulation



Example: ∀CTL/∃CTL and simulation

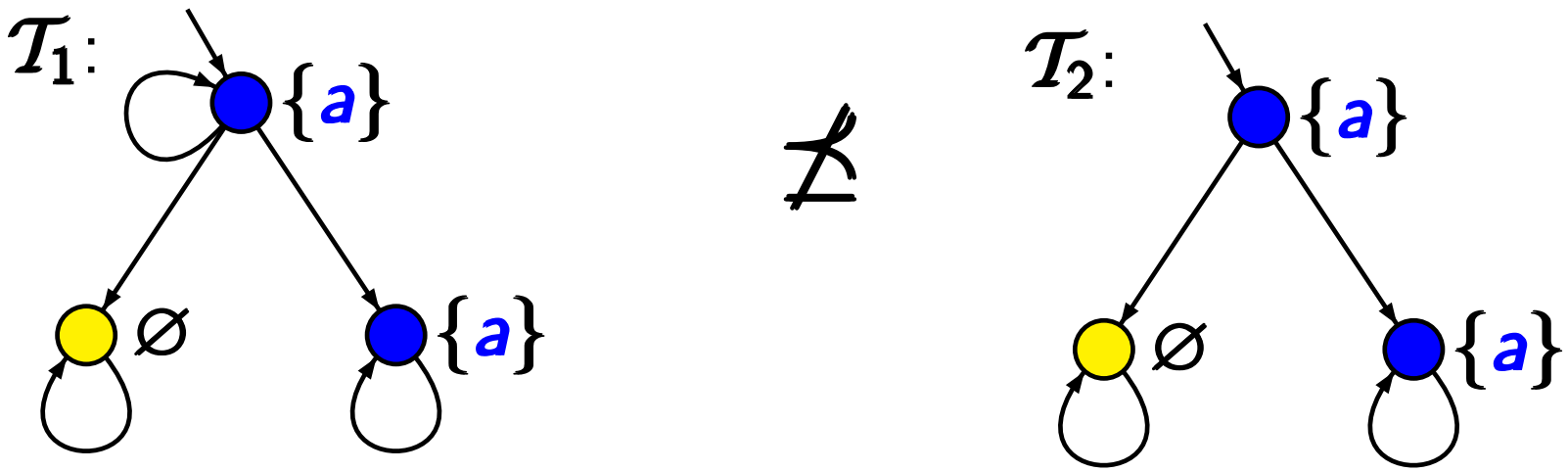


$\mathcal{T}_1 \not\models \forall\bigcirc(\forall\bigcirc\neg a \vee \forall\bigcirc a)$

$\mathcal{T}_2 \models \forall\bigcirc(\forall\bigcirc\neg a \vee \forall\bigcirc a)$

∀CTL formula

Example: ∀CTL/∃CTL and simulation



$\mathcal{T}_1 \not\models \forall\bigcirc(\forall\bigcirc\neg a \vee \forall\bigcirc a)$

$\mathcal{T}_2 \models \forall\bigcirc(\forall\bigcirc\neg a \vee \forall\bigcirc a)$

∀CTL formula

$\mathcal{T}_1 \models \exists\bigcirc(\exists\bigcirc\neg a \wedge \exists\bigcirc a)$

$\mathcal{T}_2 \not\models \exists\bigcirc(\exists\bigcirc\neg a \wedge \exists\bigcirc a)$

∃CTL formula

Characterizations of simulation equivalence

Characterizations of simulation equivalence

for finite TS without terminal states:

Characterizations of simulation equivalence

for finite TS without terminal states:

$\mathcal{T}_1 \simeq \mathcal{T}_2$ iff $\mathcal{T}_1 \preceq \mathcal{T}_2$ and $\mathcal{T}_2 \preceq \mathcal{T}_1$

Characterizations of simulation equivalence

for finite TS without terminal states:

$\mathcal{T}_1 \simeq \mathcal{T}_2$ iff $\mathcal{T}_1 \preceq \mathcal{T}_2$ and $\mathcal{T}_2 \preceq \mathcal{T}_1$

iff $\mathcal{T}_1, \mathcal{T}_2$ satisfy the same $\forall$CTL* formulas

Characterizations of simulation equivalence

for finite TS without terminal states:

$\mathcal{T}_1 \simeq \mathcal{T}_2$ iff $\mathcal{T}_1 \preceq \mathcal{T}_2$ and $\mathcal{T}_2 \preceq \mathcal{T}_1$

iff $\mathcal{T}_1, \mathcal{T}_2$ satisfy the same $\forall$CTL* formulas

iff $\mathcal{T}_1, \mathcal{T}_2$ satisfy the same $\forall$CTL formulas

Characterizations of simulation equivalence

for finite TS without terminal states:

$\mathcal{T}_1 \simeq \mathcal{T}_2$ iff $\mathcal{T}_1 \preceq \mathcal{T}_2$ and $\mathcal{T}_2 \preceq \mathcal{T}_1$

iff $\mathcal{T}_1, \mathcal{T}_2$ satisfy the same $\forall$CTL* formulas

iff $\mathcal{T}_1, \mathcal{T}_2$ satisfy the same $\forall$CTL formulas

iff $\mathcal{T}_1, \mathcal{T}_2$ satisfy the same $\exists$CTL* formulas

Characterizations of simulation equivalence

for finite TS without terminal states:

$\mathcal{T}_1 \simeq \mathcal{T}_2$ iff $\mathcal{T}_1 \preceq \mathcal{T}_2$ and $\mathcal{T}_2 \preceq \mathcal{T}_1$

iff $\mathcal{T}_1, \mathcal{T}_2$ satisfy the same $\forall$CTL* formulas

iff $\mathcal{T}_1, \mathcal{T}_2$ satisfy the same $\forall$CTL formulas

iff $\mathcal{T}_1, \mathcal{T}_2$ satisfy the same $\exists$CTL* formulas

iff $\mathcal{T}_1, \mathcal{T}_2$ satisfy the same $\exists$CTL formulas

Characterizations of simulation equivalence

for finite TS without terminal states:

$\mathcal{T}_1 \simeq \mathcal{T}_2$ iff $\mathcal{T}_1 \preceq \mathcal{T}_2$ and $\mathcal{T}_2 \preceq \mathcal{T}_1$

iff $\mathcal{T}_1, \mathcal{T}_2$ satisfy the same $\forall$CTL* formulas

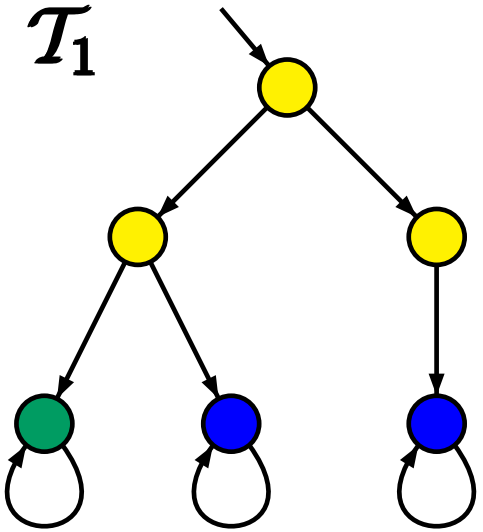
iff $\mathcal{T}_1, \mathcal{T}_2$ satisfy the same $\forall$CTL formulas

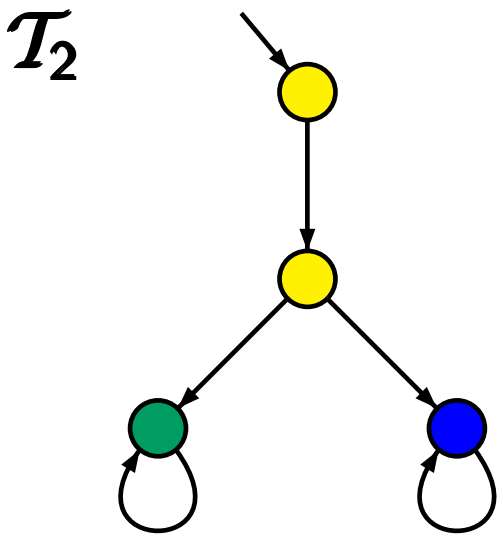
iff $\mathcal{T}_1, \mathcal{T}_2$ satisfy the same $\exists$CTL* formulas

iff $\mathcal{T}_1, \mathcal{T}_2$ satisfy the same $\exists$CTL formulas

... even holds for $\forall\text{CTL}^*_{\setminus U,W}$, $\forall\text{CTL}_{\setminus U,W}$, $\exists\text{CTL}^*_{\setminus U,W}$, $\exists\text{CTL}_{\setminus U,W}$

Simulation equivalence

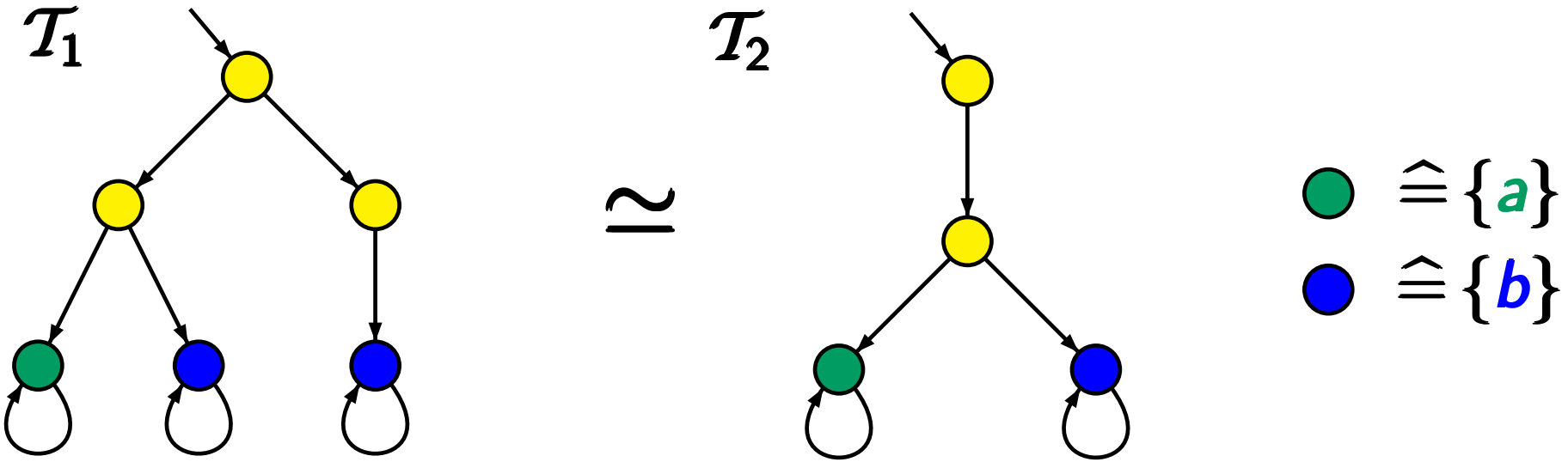




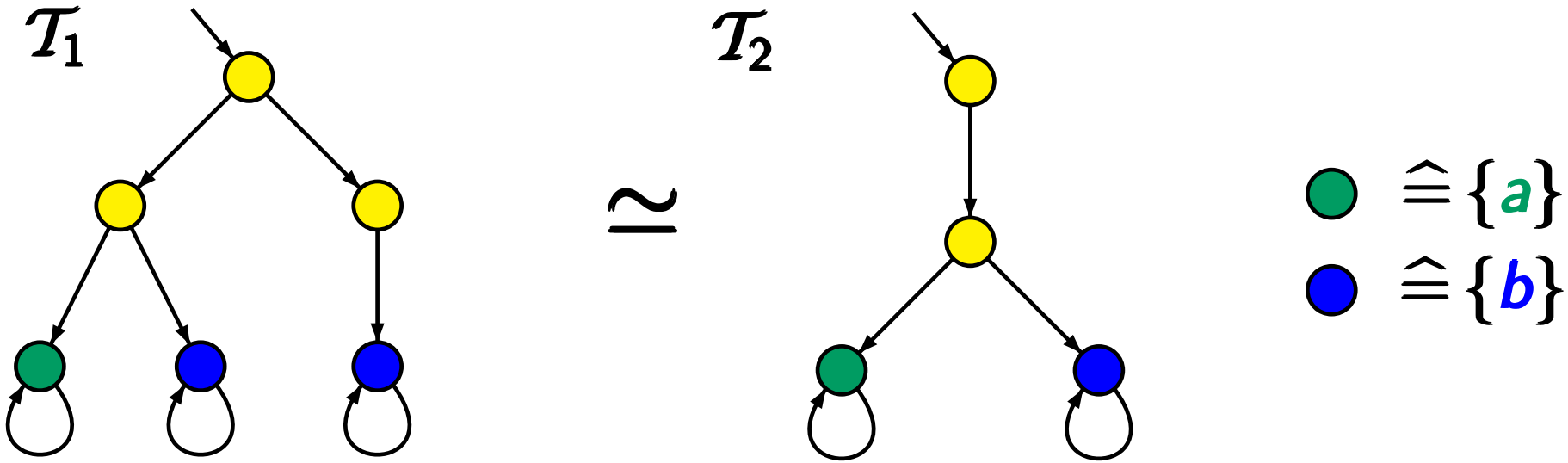
$\widehat{=} \{a\}$

$\widehat{=} \{b\}$

Simulation equivalence

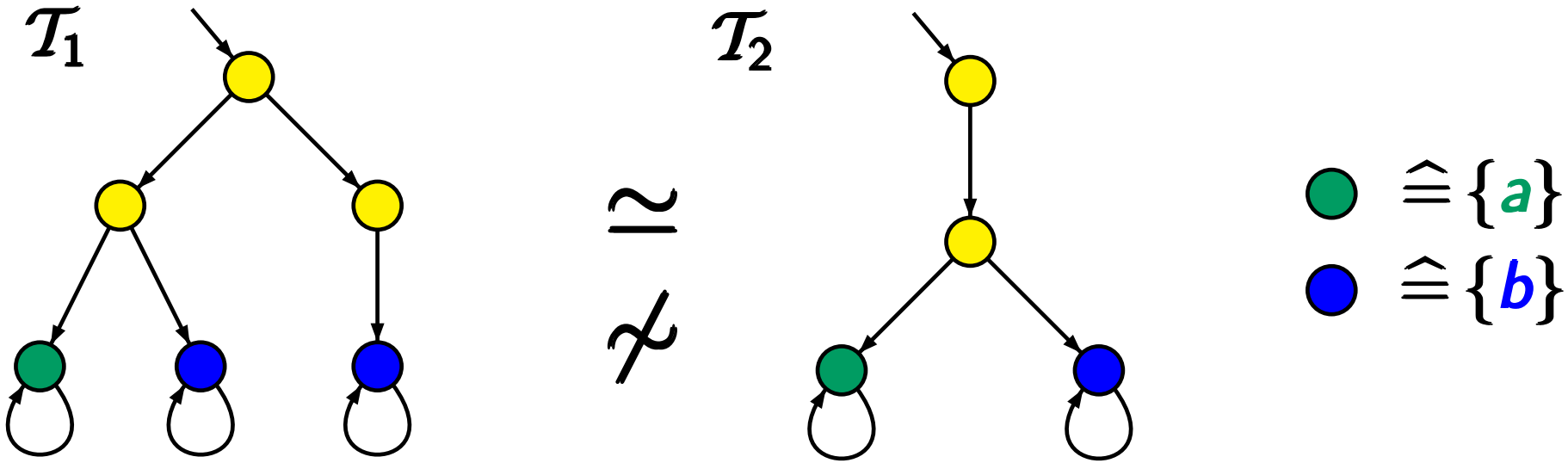


Simulation equivalence



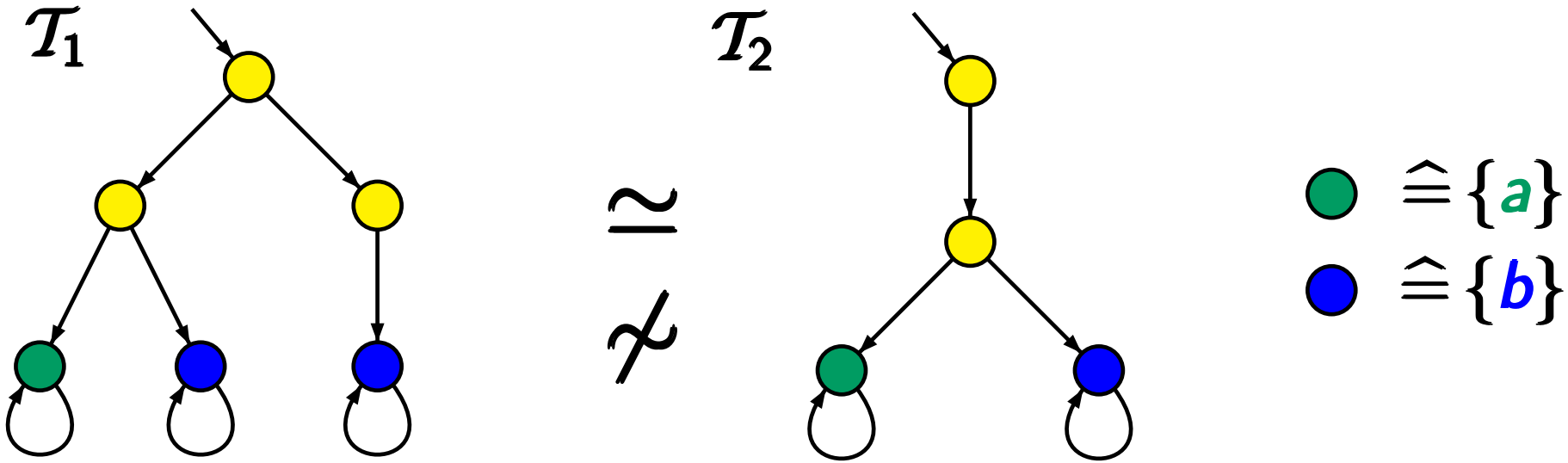
$\mathcal{T}_1$, $\mathcal{T}_2$ cannot be distinguished by the temporal logics $\forall$CTL, $\forall$CTL*, $\exists$CTL, or $\exists$CTL*,

Simulation equivalence



$\mathcal{T}_1$, $\mathcal{T}_2$ cannot be distinguished by the temporal logics $\forall$CTL, $\forall$CTL*, $\exists$CTL, or $\exists$CTL*,

Simulation equivalence



$\mathcal{T}_1$, $\mathcal{T}_2$ cannot be distinguished by the temporal logics
$\forall$**CTL**, $\forall$**CTL***, $\exists$**CTL**, or $\exists$**CTL***,

but by **CTL**:

$$\mathcal{T}_1 \not\models \forall\bigcirc(\exists\bigcirc a \wedge \exists\bigcirc b)$$

$$\mathcal{T}_2 \models \forall\bigcirc(\exists\bigcirc a \wedge \exists\bigcirc b)$$

Correct or wrong?

In finite TS without terminal states:

If s_1, s_2 satisfy the same $\exists$CTL formulas
then s_1, s_2 satisfy the same LTL formulas

Correct or wrong?

In finite TS without terminal states:

> If s_1, s_2 satisfy the same $\exists$CTL formulas
> then s_1, s_2 satisfy the same LTL formulas

correct

Correct or wrong?

In finite TS without terminal states:

> If s_1, s_2 satisfy the same $\exists$CTL formulas
> then s_1, s_2 satisfy the same LTL formulas

correct

$\exists$CTL equivalence

$=$ simulation equivalence

$=$ $\forall$CTL* equivalence

Correct or wrong?

In finite TS without terminal states:

> If s_1, s_2 satisfy the same $\exists$CTL formulas
> then s_1, s_2 satisfy the same LTL formulas

correct

$\exists$CTL equivalence

$=$ simulation equivalence

$=$ $\forall$CTL* equivalence

and LTL is a sublogic of $\forall$CTL*

Correct or wrong?

In finite TS without terminal states:

> If s_1, s_2 satisfy the same $\exists$CTL formulas
> then s_1, s_2 satisfy the same LTL formulas

correct

> If s_1, s_2 satisfy the same LTL formulas
> then s_1, s_2 satisfy the same $\forall$CTL formulas

Correct or wrong?

In finite TS without terminal states:

> If s_1, s_2 satisfy the same $\exists$CTL formulas
> then s_1, s_2 satisfy the same LTL formulas

correct

> If s_1, s_2 satisfy the same LTL formulas
> then s_1, s_2 satisfy the same $\forall$CTL formulas

wrong

Correct or wrong?

In finite TS without terminal states:

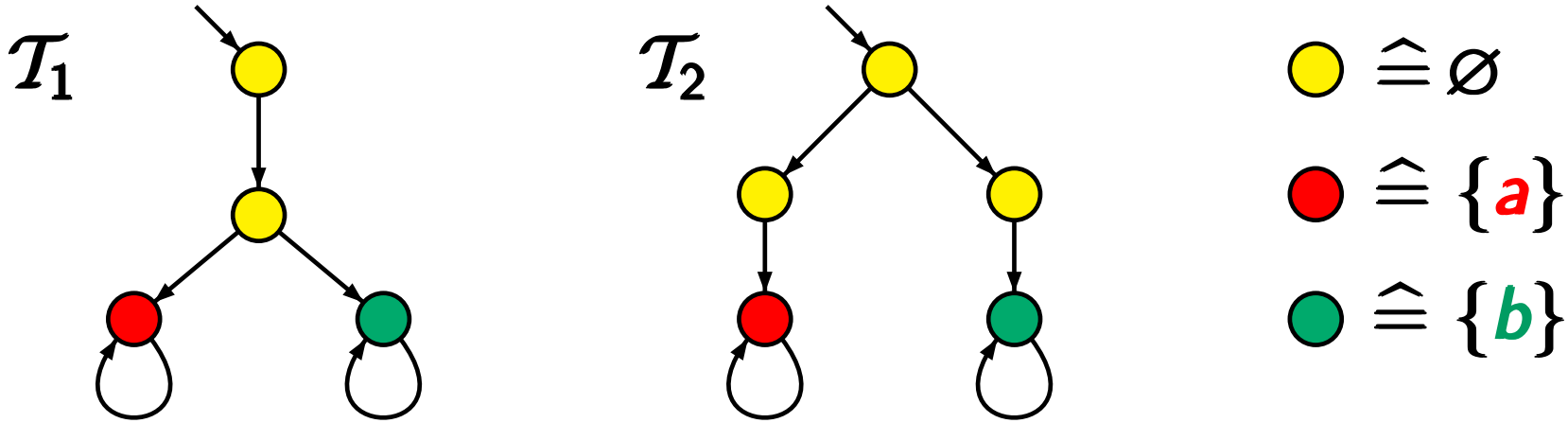
> If s_1, s_2 satisfy the same $\exists$CTL formulas
> then s_1, s_2 satisfy the same LTL formulas

correct

> If s_1, s_2 satisfy the same LTL formulas
> then s_1, s_2 satisfy the same $\forall$CTL formulas

wrong, as trace equivalence does not imply simulation equivalence

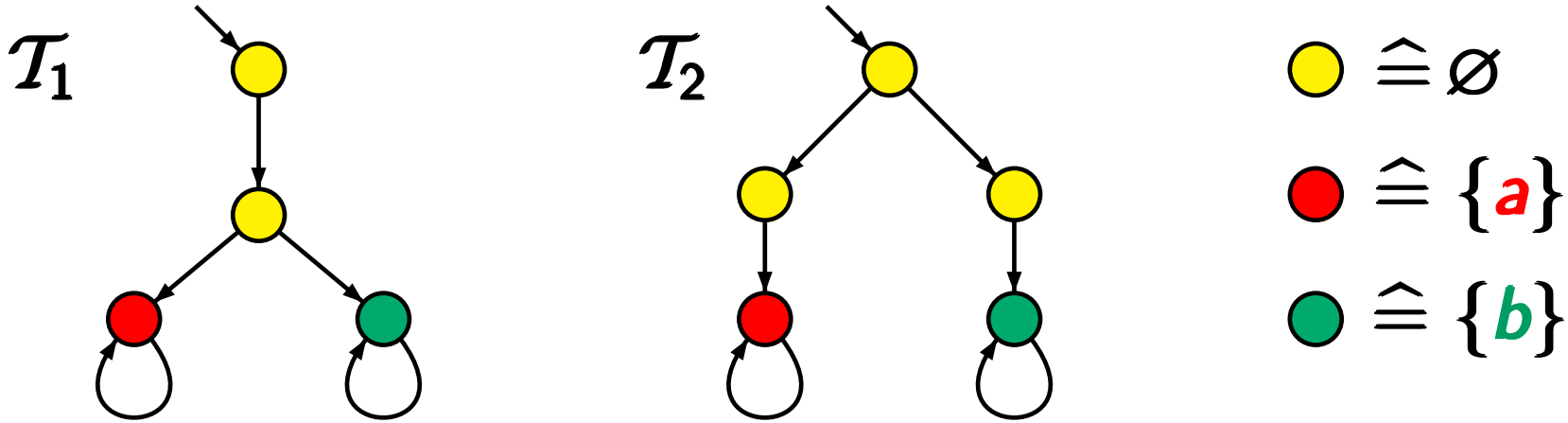
Does there exist ...?



Does there exist a $\exists$CTL formula Φ s.t.

$\mathcal{T}_1 \models \Phi$ and $\mathcal{T}_2 \not\models \Phi$?

Does there exist ...?

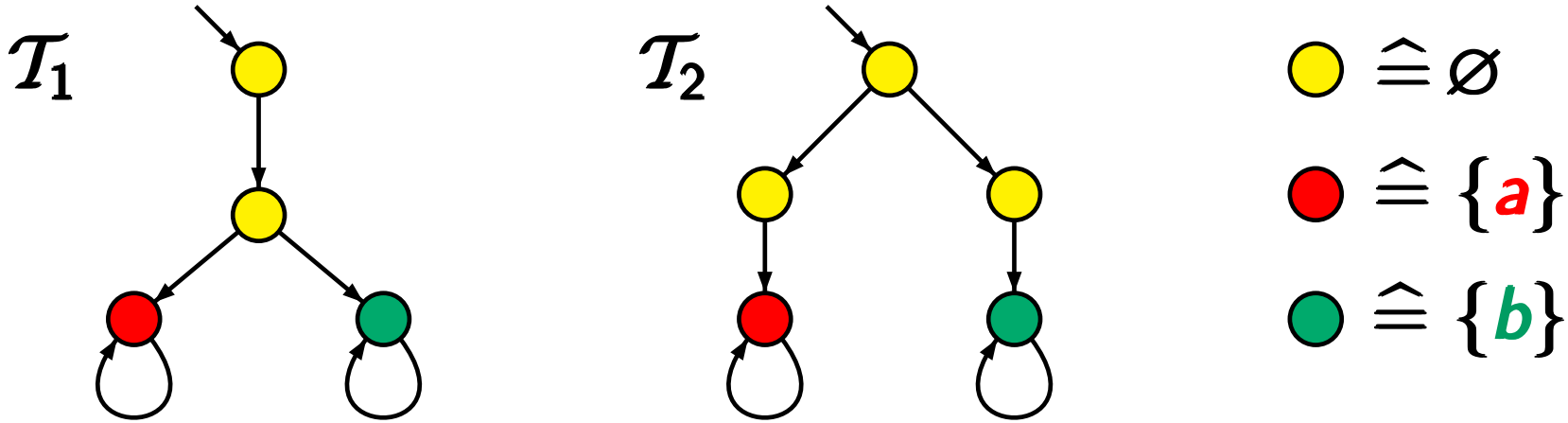


Does there exist a $\exists$CTL formula Φ s.t.

$\mathcal{T}_1 \models \Phi$ and $\mathcal{T}_2 \not\models \Phi$?

yes

Does there exist ...?

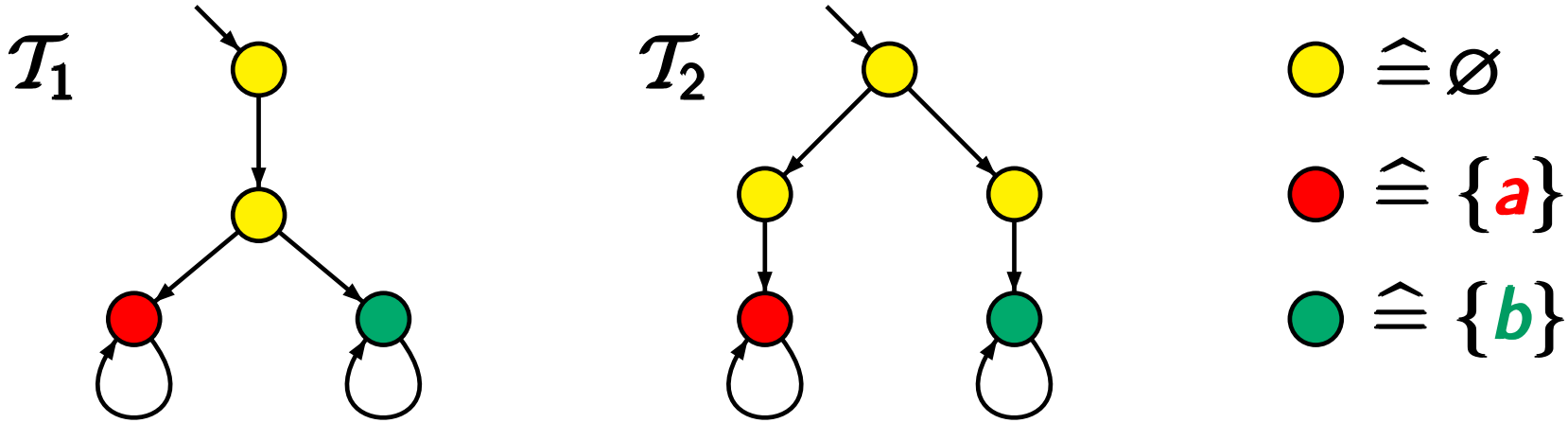


Does there exist a $\exists$CTL formula Φ s.t.

$\mathcal{T}_1 \models \Phi$ and $\mathcal{T}_2 \not\models \Phi$?

yes, as $\mathcal{T}_1 \not\trianglelefteq \mathcal{T}_2$

Does there exist ...?

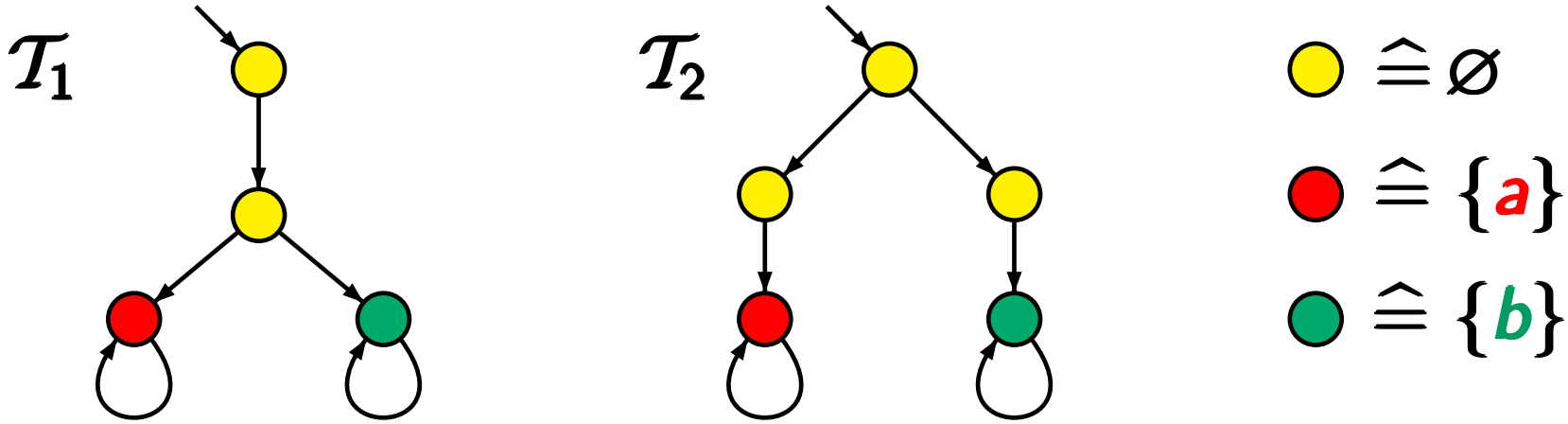


Does there exist a $\exists$**CTL** formula Φ s.t.

$\mathcal{T}_1 \models \Phi$ and $\mathcal{T}_2 \not\models \Phi$?

yes, as $\mathcal{T}_1 \not\trianglelefteq \mathcal{T}_2$, e.g., $\Phi = \exists\bigcirc(\exists\bigcirc a \wedge \exists\bigcirc b)$

Does there exist ...?



Does there exist a $\exists$**CTL** formula Φ s.t.

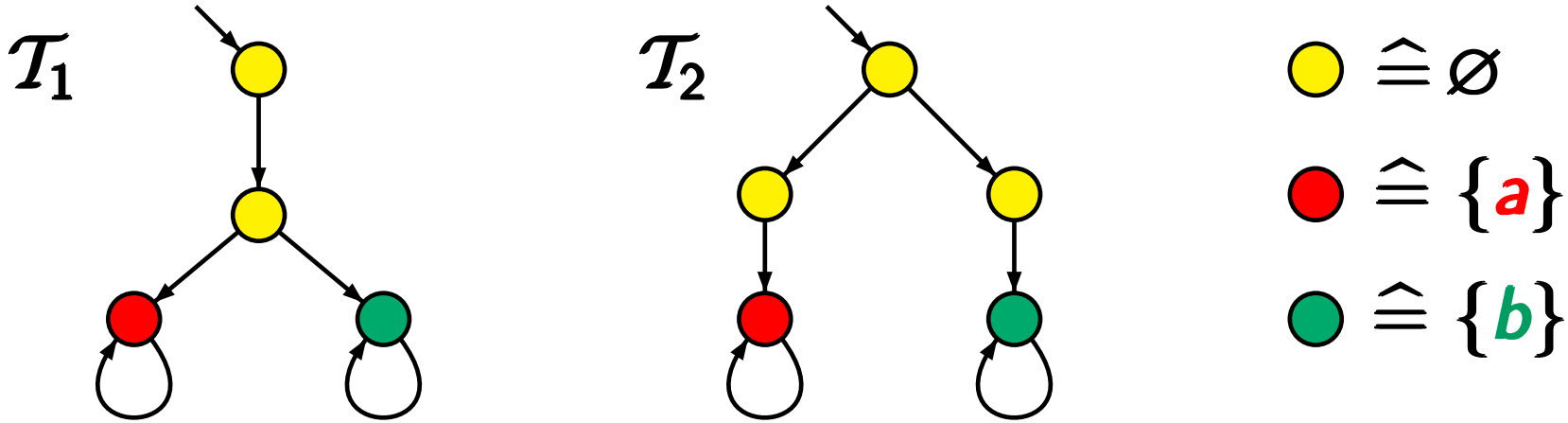
$\mathcal{T}_1 \models \Phi$ and $\mathcal{T}_2 \not\models \Phi$?

yes, as $\mathcal{T}_1 \not\preceq \mathcal{T}_2$, e.g., $\Phi = \exists\bigcirc(\exists\bigcirc a \wedge \exists\bigcirc b)$

Does there exist a $\forall$**CTL** formula Φ s.t.

$\mathcal{T}_1 \models \Phi$ and $\mathcal{T}_2 \not\models \Phi$?

Does there exist ...?



Does there exist a $\exists$**CTL** formula Φ s.t.

$\mathcal{T}_1 \models \Phi$ and $\mathcal{T}_2 \not\models \Phi$?

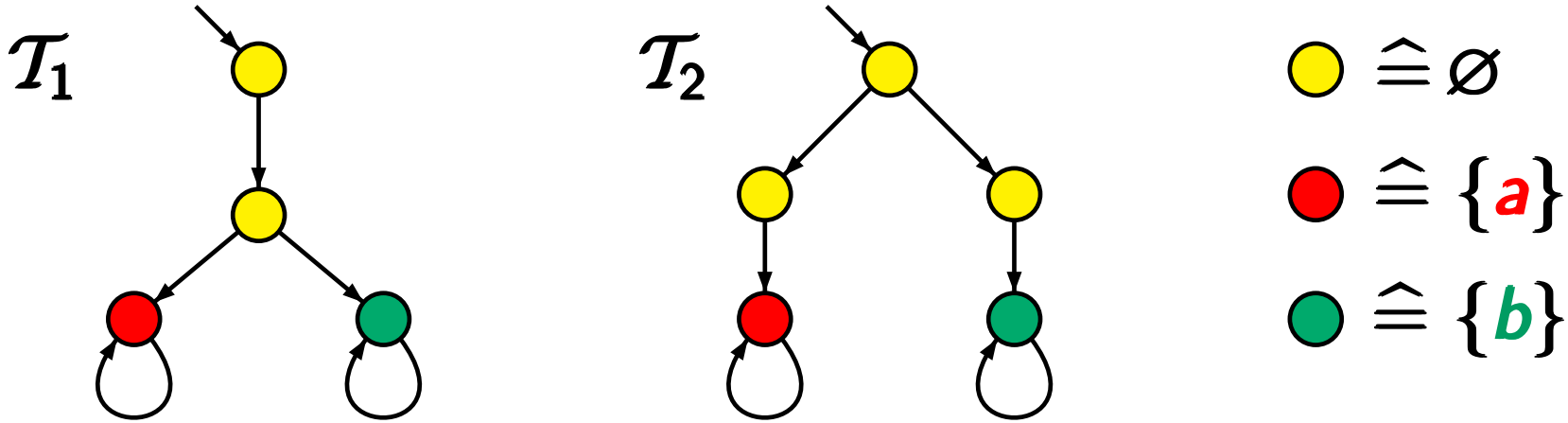
yes, as $\mathcal{T}_1 \not\trianglelefteq \mathcal{T}_2$, e.g., $\Phi = \exists\bigcirc(\exists\bigcirc a \wedge \exists\bigcirc b)$

Does there exist a $\forall$**CTL** formula Φ s.t.

$\mathcal{T}_1 \models \Phi$ and $\mathcal{T}_2 \not\models \Phi$?

no

Does there exist ...?



Does there exist a $\exists$**CTL** formula Φ s.t.

$\mathcal{T}_1 \models \Phi$ and $\mathcal{T}_2 \not\models \Phi$?

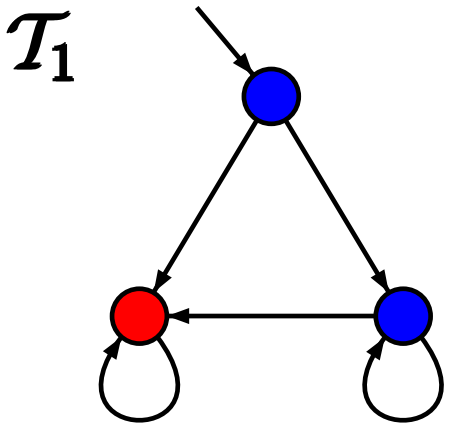
yes, as $\mathcal{T}_1 \not\preceq \mathcal{T}_2$, e.g., $\Phi = \exists\bigcirc(\exists\bigcirc a \wedge \exists\bigcirc b)$

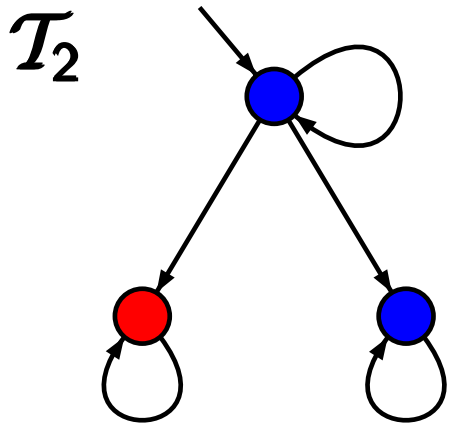
Does there exist a $\forall$**CTL** formula Φ s.t.

$\mathcal{T}_1 \models \Phi$ and $\mathcal{T}_2 \not\models \Phi$?

no, as $\mathcal{T}_2 \preceq \mathcal{T}_1$

Does there exist ...?

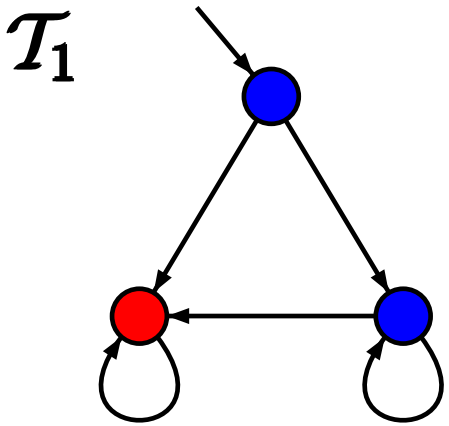


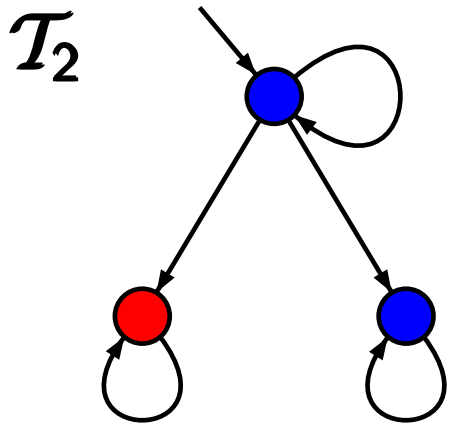


Does there exist a $\exists$**CTL** formula Φ s.t.

$\mathcal{T}_1 \models \Phi$ and $\mathcal{T}_2 \not\models \Phi$?

Does there exist ...?



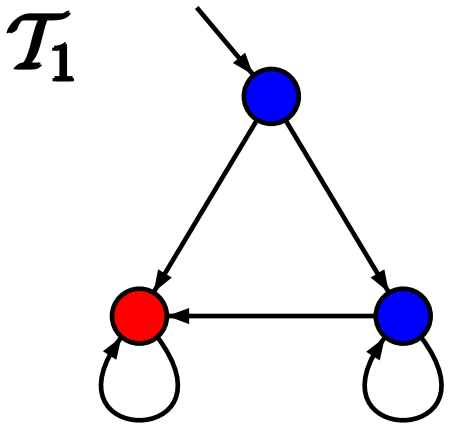


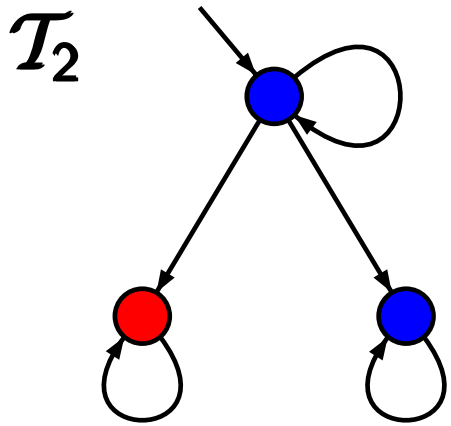
Does there exist a $\exists$**CTL** formula Φ s.t.

$\mathcal{T}_1 \models \Phi$ and $\mathcal{T}_2 \not\models \Phi$?

no

Does there exist ...?



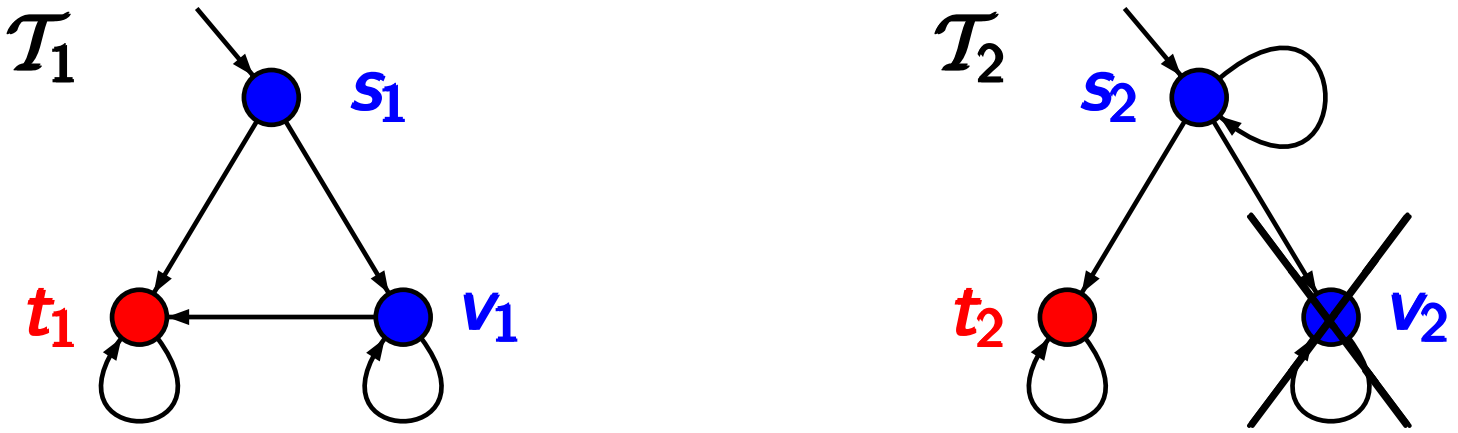


Does there exist a $\exists$**CTL** formula Φ s.t.

$\mathcal{T}_1 \models \Phi$ and $\mathcal{T}_2 \not\models \Phi$?

no, since $\mathcal{T}_1 \simeq \mathcal{T}_2$

Does there exist ...?



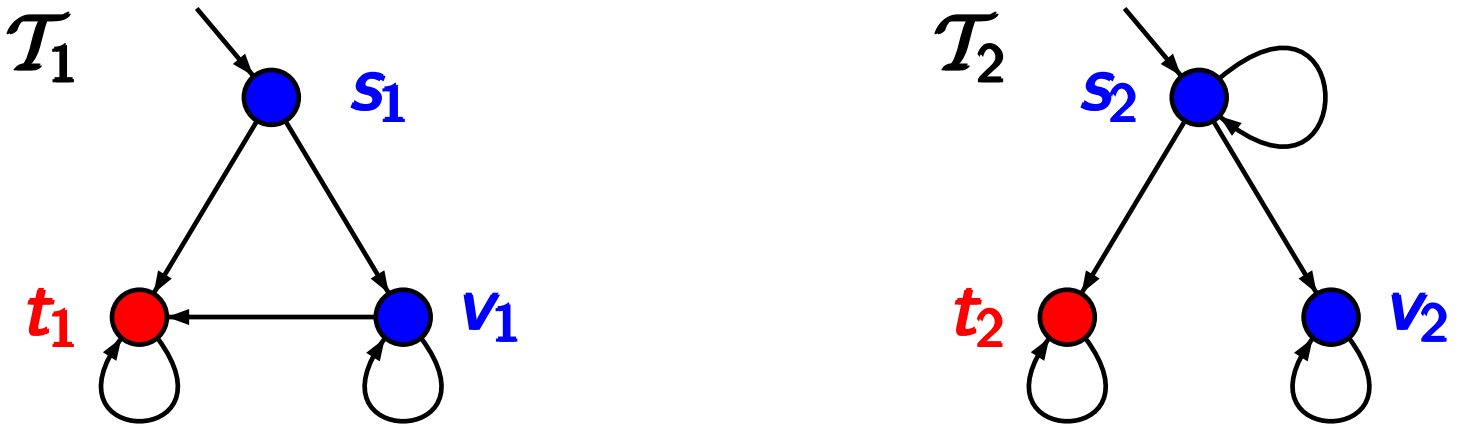
Does there exist a $\exists$**CTL** formula Φ s.t.

$\mathcal{T}_1 \models \Phi$ and $\mathcal{T}_2 \not\models \Phi$?

no, since $\mathcal{T}_1 \simeq \mathcal{T}_2$

simulation for $(\mathcal{T}_1, \mathcal{T}_2)$: $\{(s_1, s_2), (v_1, s_2), (t_1, t_2)\}$

Does there exist ...?



Does there exist a $\exists$**CTL** formula Φ s.t.

$\mathcal{T}_1 \models \Phi$ and $\mathcal{T}_2 \not\models \Phi$?

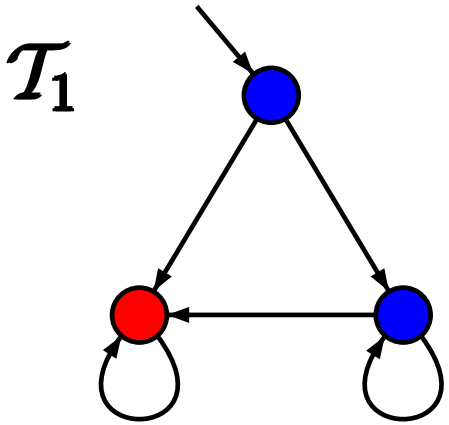
no, since $\mathcal{T}_1 \simeq \mathcal{T}_2$

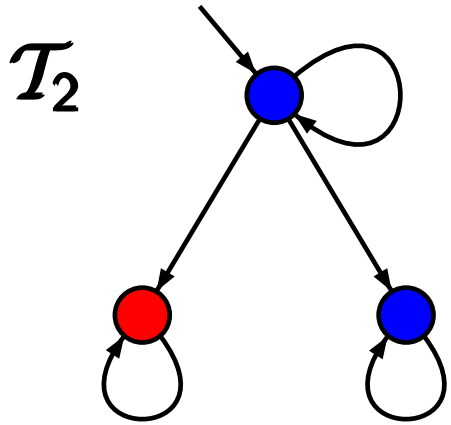
simulation for $(\mathcal{T}_1, \mathcal{T}_2)$: $\{(s_1, s_2),\ (v_1, s_2),\ (t_1, t_2)\}$

simulation for $(\mathcal{T}_2, \mathcal{T}_1)$:

$\{(s_2, s_1),\ (s_2, v_1),\ (v_2, v_1),\ (t_1, t_2)\}$

Does there exist ...?

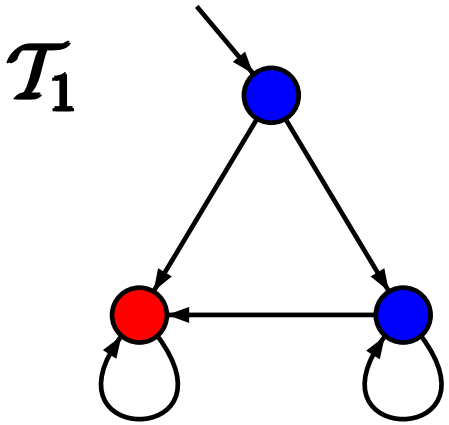


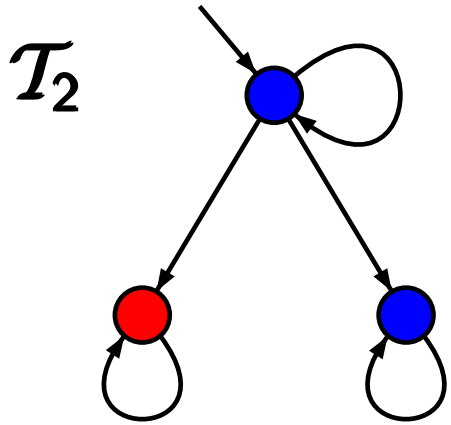


Does there exist a **CTL** formula Φ s.t.

$\mathcal{T}_1 \not\models \Phi$ and $\mathcal{T}_2 \models \Phi$?

Does there exist ...?



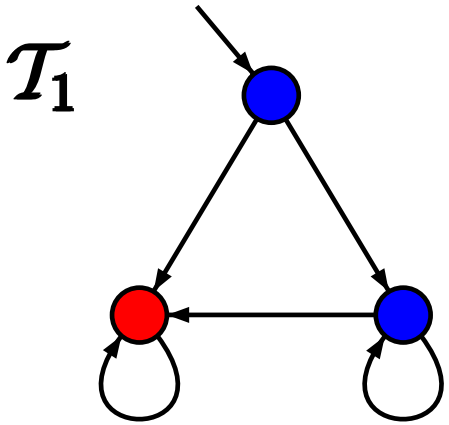


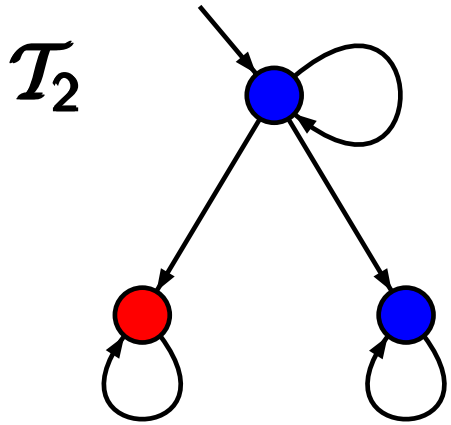
Does there exist a **CTL** formula Φ s.t.

$\mathcal{T}_1 \not\models \Phi$ and $\mathcal{T}_2 \models \Phi$?

yes

Does there exist ...?



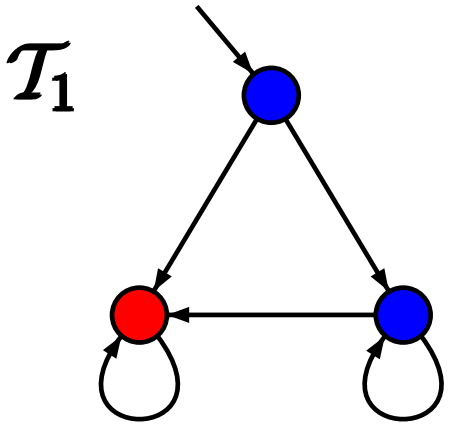


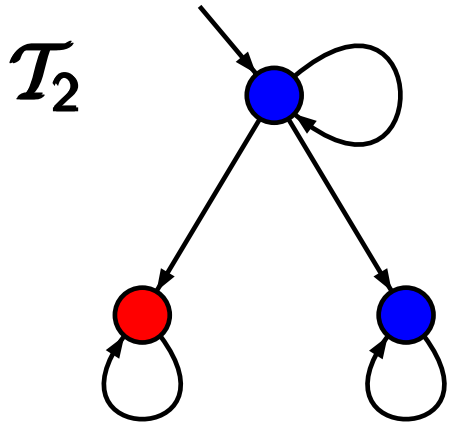
Does there exist a **CTL** formula Φ s.t.

$\mathcal{T}_1 \not\models \Phi$ and $\mathcal{T}_2 \models \Phi$?

yes, as $\mathcal{T}_1 \not\sim \mathcal{T}_2$

Does there exist ...?



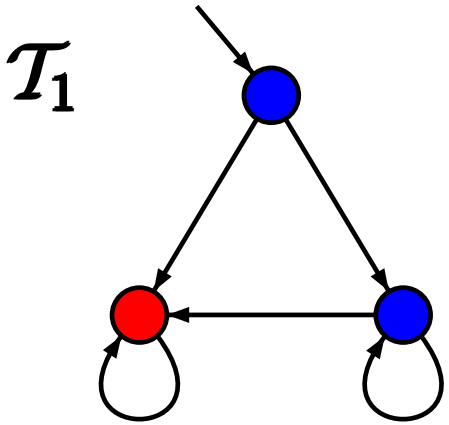


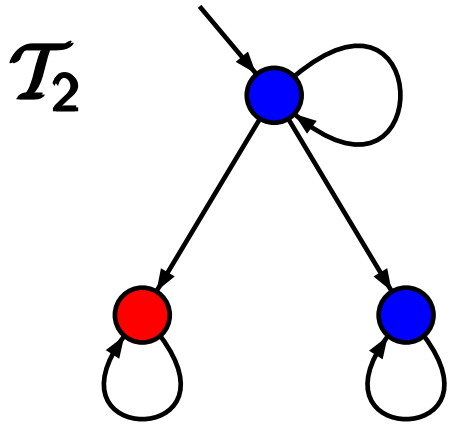
Does there exist a **CTL** formula Φ s.t.

$\mathcal{T}_1 \not\models \Phi$ and $\mathcal{T}_2 \models \Phi$?

yes, as $\mathcal{T}_1 \not\sim \mathcal{T}_2$, e.g., $\Phi = \exists\bigcirc\forall\Box \textit{blue}$

Does there exist ...?





Does there exist a **CTL** formula Φ s.t.

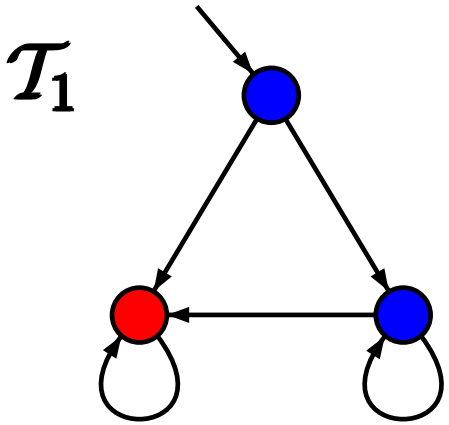
$\mathcal{T}_1 \not\models \Phi$ and $\mathcal{T}_2 \models \Phi$?

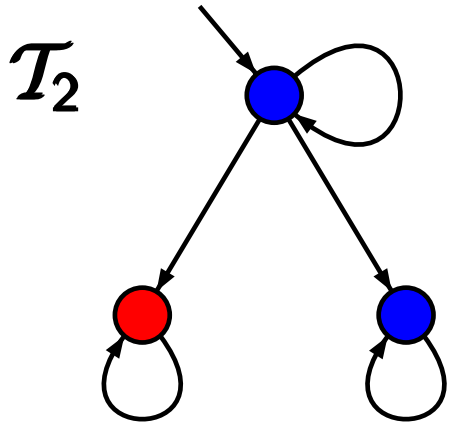
yes, as $\mathcal{T}_1 \not\sim \mathcal{T}_2$, e.g., $\Phi = \exists\bigcirc\forall\Box \textit{blue}$

Does there exist a **LTL** formula φ s.t.

$\mathcal{T}_1 \not\models \varphi$ and $\mathcal{T}_2 \models \varphi$?

Does there exist ...?





Does there exist a **CTL** formula Φ s.t.

$\mathcal{T}_1 \not\models \Phi$ and $\mathcal{T}_2 \models \Phi$?

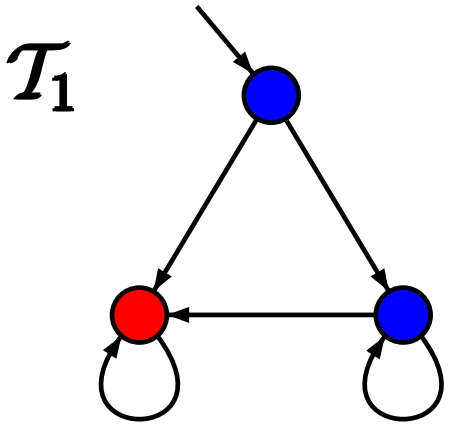
yes, as $\mathcal{T}_1 \not\sim \mathcal{T}_2$, e.g., $\Phi = \exists\bigcirc\forall\Box \textit{blue}$

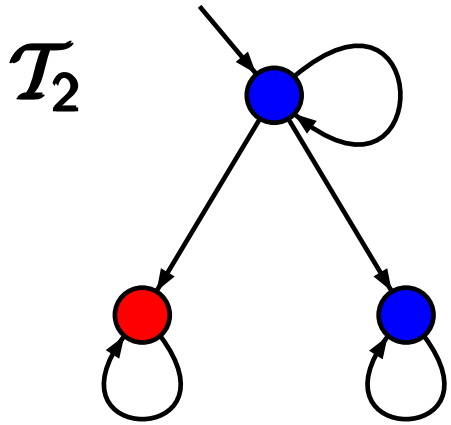
Does there exist a **LTL** formula φ s.t.

$\mathcal{T}_1 \not\models \varphi$ and $\mathcal{T}_2 \models \varphi$?

no

Does there exist ...?





Does there exist a **CTL** formula Φ s.t.

$\mathcal{T}_1 \not\models \Phi$ and $\mathcal{T}_2 \models \Phi$?

yes, as $\mathcal{T}_1 \not\sim \mathcal{T}_2$, e.g., $\Phi = \exists\bigcirc\forall\Box \mathit{blue}$

Does there exist a **LTL** formula φ s.t.

$\mathcal{T}_1 \not\models \varphi$ and $\mathcal{T}_2 \models \varphi$?

no, as $\mathcal{T}_1$, $\mathcal{T}_2$ are simulation equivalent

Simulation quotient

Simulation quotient

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

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

transition system that arises from $\mathcal{T}$ by collapsing all simulation equivalent states

Simulation quotient

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

$$\mathcal{T}/\simeq \stackrel{\text{def}}{=} (S/\simeq, Act', \rightarrow_{\simeq}, S'_0, AP', L')$$

Simulation quotient

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

$$\mathcal{T}/\simeq \stackrel{\text{def}}{=} (S/\simeq, Act', \rightarrow_{\simeq}, S'_0, AP', L')$$

- state space $S/\simeq$ $\longleftarrow$ set of all simulation equivalence classes

Simulation quotient

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

$$\mathcal{T}/\simeq \stackrel{\text{def}}{=} (S/\simeq, Act', \rightarrow_{\simeq}, S'_0, AP', L')$$

- state space $S/\simeq$ $\longleftarrow$ set of all simulation equivalence classes
- initial states: $S'_0 = \{[s] : s \in S_0\}$

$$[s] = \{s' \in S : s \simeq_{\mathcal{T}} s'\}$$

Simulation quotient

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

$$\mathcal{T}/\simeq \stackrel{\text{def}}{=} (S/\simeq, Act', \rightarrow_{\simeq}, S_0', AP', L')$$

- state space $S/\simeq$ $\longleftarrow$ set of all simulation equivalence classes
- initial states: $S_0' = \{[s] : s \in S_0\}$
- labeling: $AP' = AP$ and $L'([s]) = L(s)$

$$[s] = \{s' \in S : s \simeq_{\mathcal{T}} s'\}$$

Simulation quotient

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

$$\mathcal{T}/\simeq \stackrel{\text{def}}{=} (S/\simeq, Act', \rightarrow_{\simeq}, S_0', AP', L')$$

- state space $S/\simeq$ $\longleftarrow$ set of all simulation equivalence classes
- initial states: $S_0' = \{[s] : s \in S_0\}$
- labeling: $AP' = AP$ and $L'([s]) = L(s)$
- transition relation: $$\frac{s \longrightarrow s'}{[s] \longrightarrow_{\simeq} [s']}$$

Simulation quotient

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

$$\mathcal{T}/\simeq \stackrel{\text{def}}{=} (S/\simeq, Act', \rightarrow_{\simeq}, S_0', AP', L')$$

- state space $S/\simeq$ $\longleftarrow$ set of all simulation equivalence classes
- initial states: $S_0' = \{[s] : s \in S_0\}$
- labeling: $AP' = AP$ and $L'([s]) = L(s)$
- transition relation: $$\frac{s \longrightarrow s'}{[s] \longrightarrow_{\simeq} [s']}$$

  action labels: irrelevant

Similarity of $\mathcal{T}$ and $\mathcal{T}/\simeq$

Similarity of $\mathcal{T}$ and $\mathcal{T}/\simeq$

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

$$\mathcal{T}/\simeq \;=\; (S/\simeq, Act', \rightarrow_{\simeq}, S'_0, AP, L')$$

where the transitions are given by $\dfrac{s \longrightarrow s'}{[s] \longrightarrow_{\simeq} [s']}$

$\mathcal{T}$ and $\mathcal{T}/\simeq$ are simulation equivalent, i.e.,
$\mathcal{T} \preceq \mathcal{T}/\simeq$ and $\mathcal{T}/\simeq \preceq \mathcal{T}$

Similarity of $\mathcal{T}$ and $\mathcal{T}/\simeq$

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

$$\mathcal{T}/\simeq \;=\; (S/\simeq, Act', \rightarrow_{\simeq}, S'_0, AP, L')$$

where the transitions are given by $\dfrac{s \longrightarrow s'}{[s] \longrightarrow_{\simeq} [s']}$

$\mathcal{T}$ and $\mathcal{T}/\simeq$ are simulation equivalent, i.e.,
$\mathcal{T} \preceq \mathcal{T}/\simeq$ and $\mathcal{T}/\simeq \preceq \mathcal{T}$

Proof. provide simulations for $(\mathcal{T}, \mathcal{T}/\simeq)$ and $(\mathcal{T}/\simeq, \mathcal{T})$

Similarity of $\mathcal{T}$ and $\mathcal{T}/\simeq$

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

$$\mathcal{T}/\simeq \;=\; (S/\simeq, Act', \rightarrow_{\simeq}, S'_0, AP, L')$$

where the transitions are given by $\dfrac{s \longrightarrow s'}{[s] \longrightarrow_{\simeq} [s']}$

$\mathcal{T}$ and $\mathcal{T}/\simeq$ are simulation equivalent, i.e.,
$\mathcal{T} \preceq \mathcal{T}/\simeq$ and $\mathcal{T}/\simeq \preceq \mathcal{T}$

Proof. provide simulations for $(\mathcal{T}, \mathcal{T}/\simeq)$ and $(\mathcal{T}/\simeq, \mathcal{T})$

simulation for $(\mathcal{T}, \mathcal{T}/\simeq)$: $\{(s, [s]) : s \in S\}$

Similarity of $\mathcal{T}$ and $\mathcal{T}/\simeq$

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

$$\mathcal{T}/\simeq \;=\; (S/\simeq, Act', \rightarrow_{\simeq}, S'_0, AP, L')$$

where the transitions are given by $\dfrac{s \longrightarrow s'}{[s] \longrightarrow_{\simeq} [s']}$

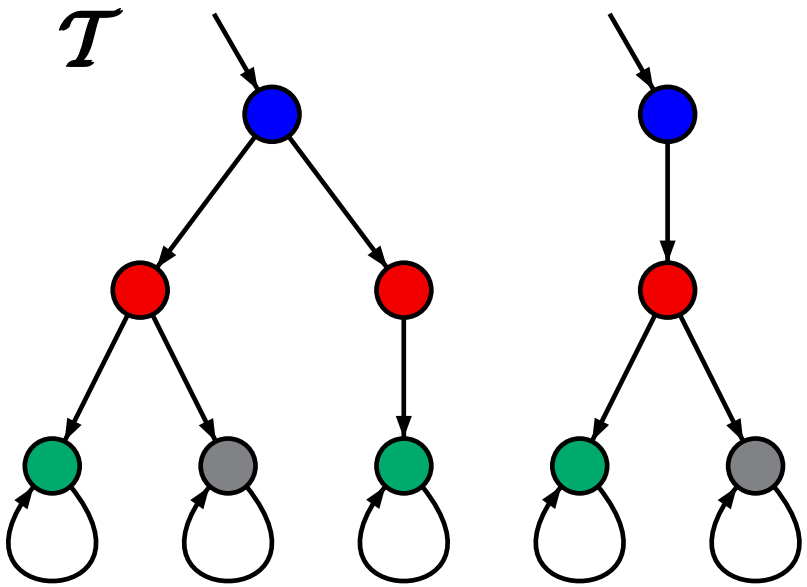
$\mathcal{T}$ and $\mathcal{T}/\simeq$ are simulation equivalent, i.e.,
$\mathcal{T} \preceq \mathcal{T}/\simeq$ and $\mathcal{T}/\simeq \preceq \mathcal{T}$

Proof. provide simulations for $(\mathcal{T}, \mathcal{T}/\simeq)$ and $(\mathcal{T}/\simeq, \mathcal{T})$

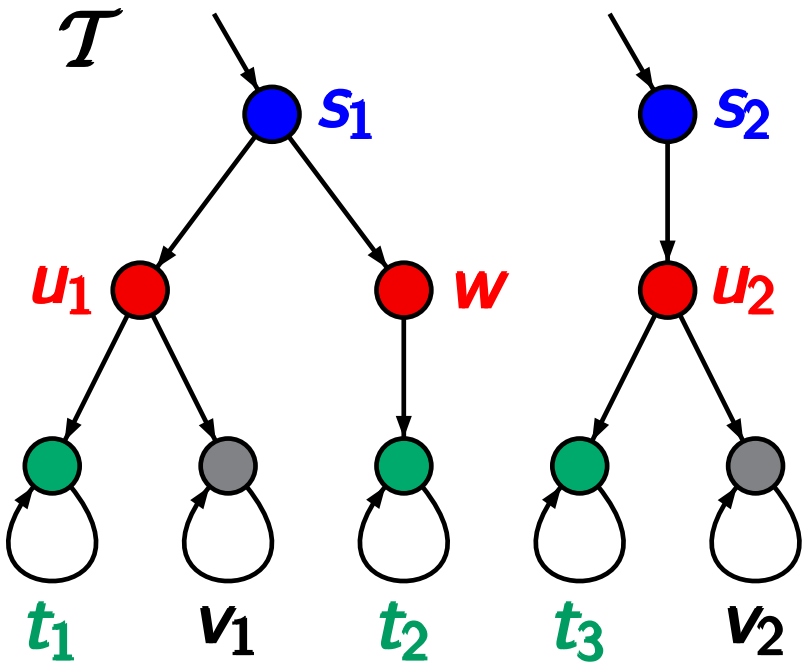
simulation for $(\mathcal{T}, \mathcal{T}/\simeq)$: $\{(s, [s]) : s \in S\}$

simulation for $(\mathcal{T}/\simeq, \mathcal{T})$: ?

Example: simulation quotient



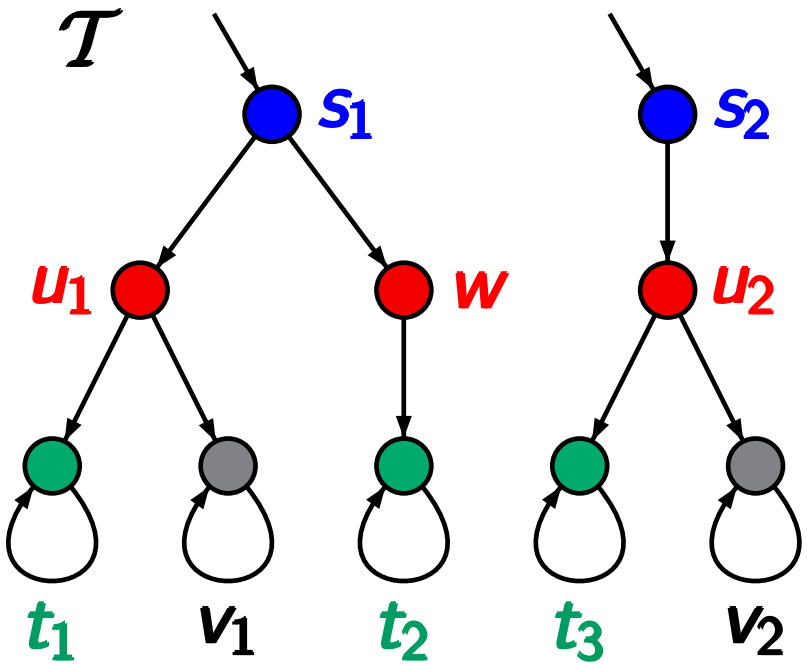
Example: simulation quotient



t_1, t_2, t_3 are simulation equivalent

v_1, v_2 are simulation equivalent

Example: simulation quotient

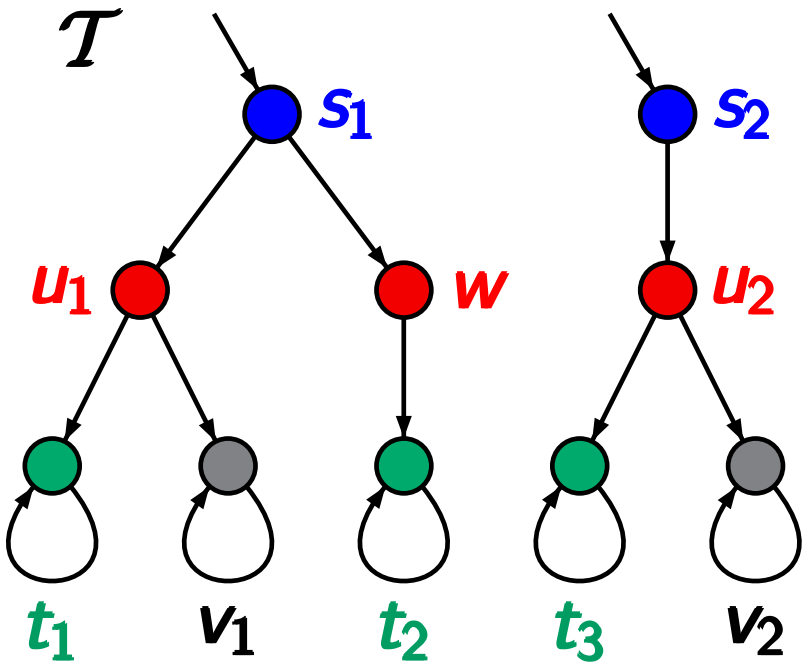


t_1, t_2, t_3 are simulation equivalent

v_1, v_2 are simulation equivalent

$u_1 \simeq u_2$,

Example: simulation quotient

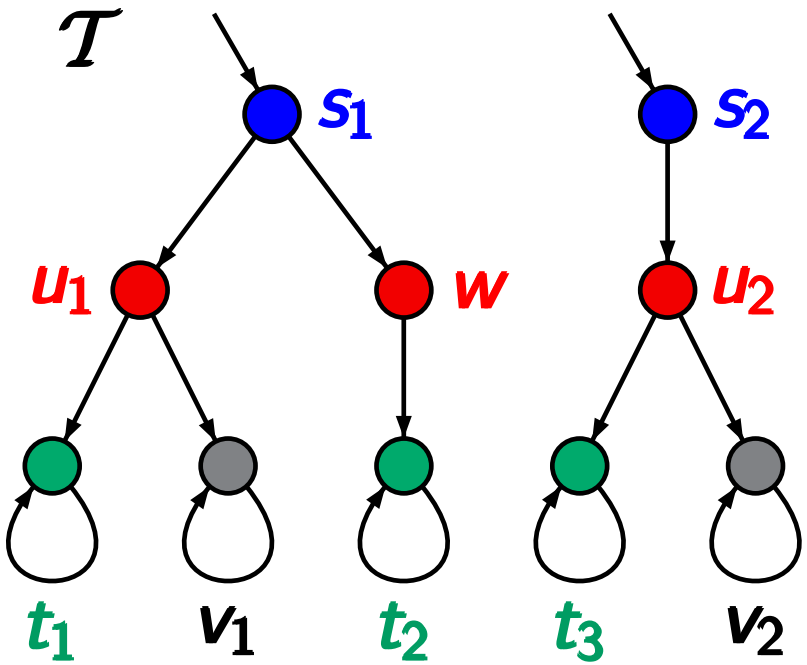


t_1, t_2, t_3 are simulation equivalent

v_1, v_2 are simulation equivalent

$u_1 \simeq u_2$, $\quad w \preceq u_1, u_2$, but $w \not\simeq u_1, u_2$

Example: simulation quotient



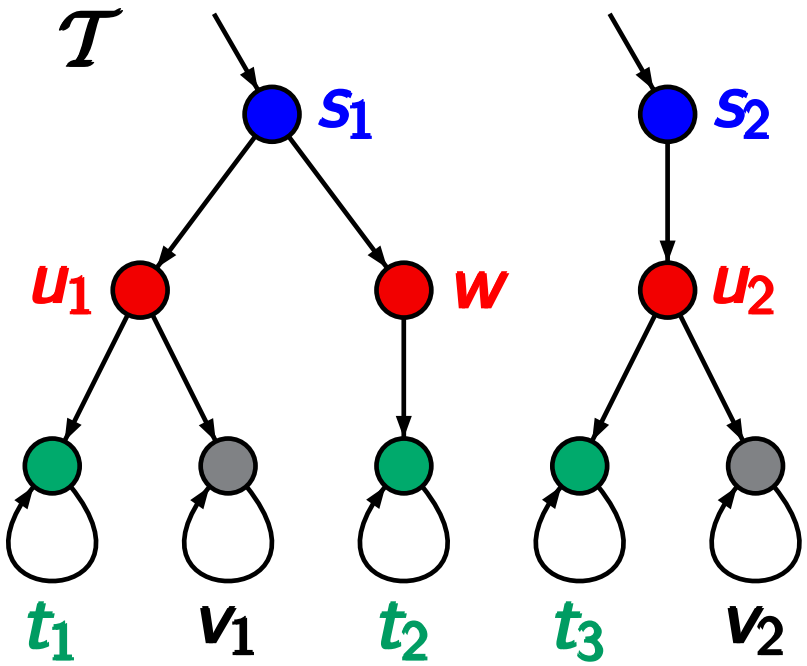
t_1, t_2, t_3 are simulation equivalent

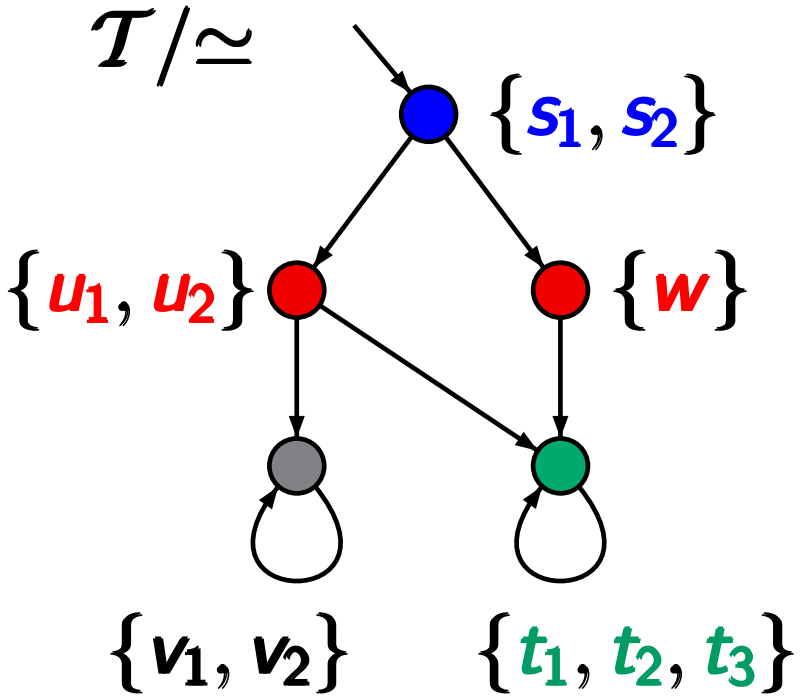
v_1, v_2 are simulation equivalent

$u_1 \simeq u_2$, $\quad w \preceq u_1, u_2$, but $w \not\simeq u_1, u_2$

$s_1 \simeq s_2$

Example: simulation quotient





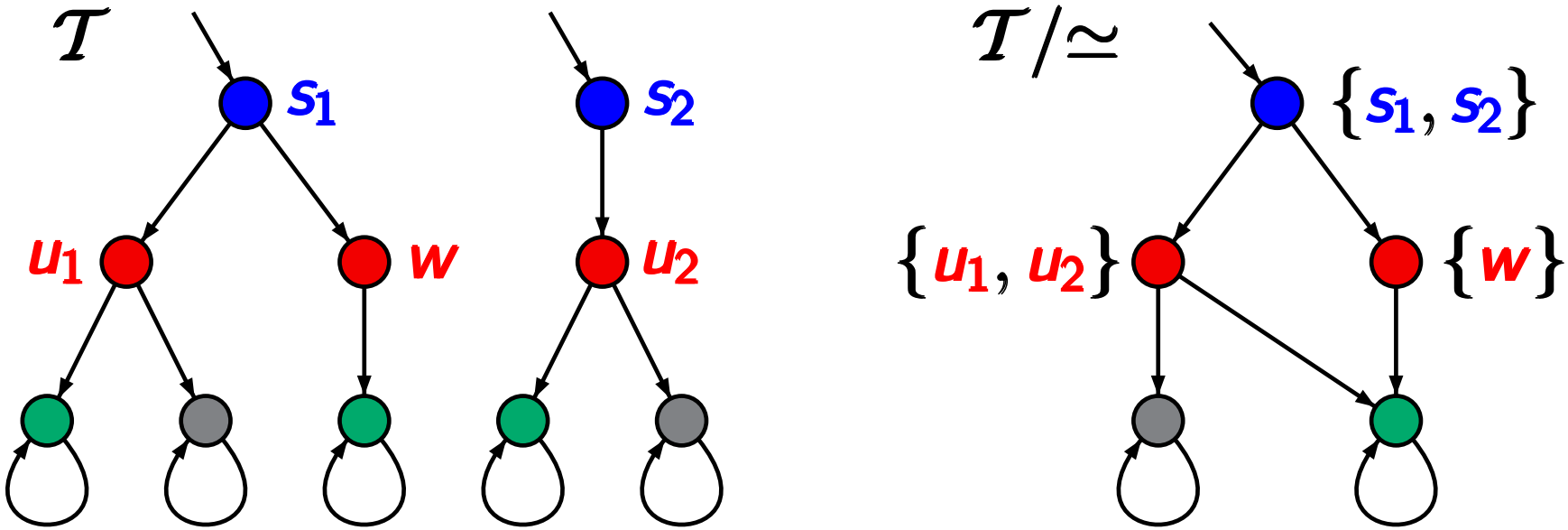
t_1, t_2, t_3 are simulation equivalent

v_1, v_2 are simulation equivalent

$u_1 \simeq u_2$, $\quad w \preceq u_1, u_2$, but $w \not\simeq u_1, u_2$

$s_1 \simeq s_2$

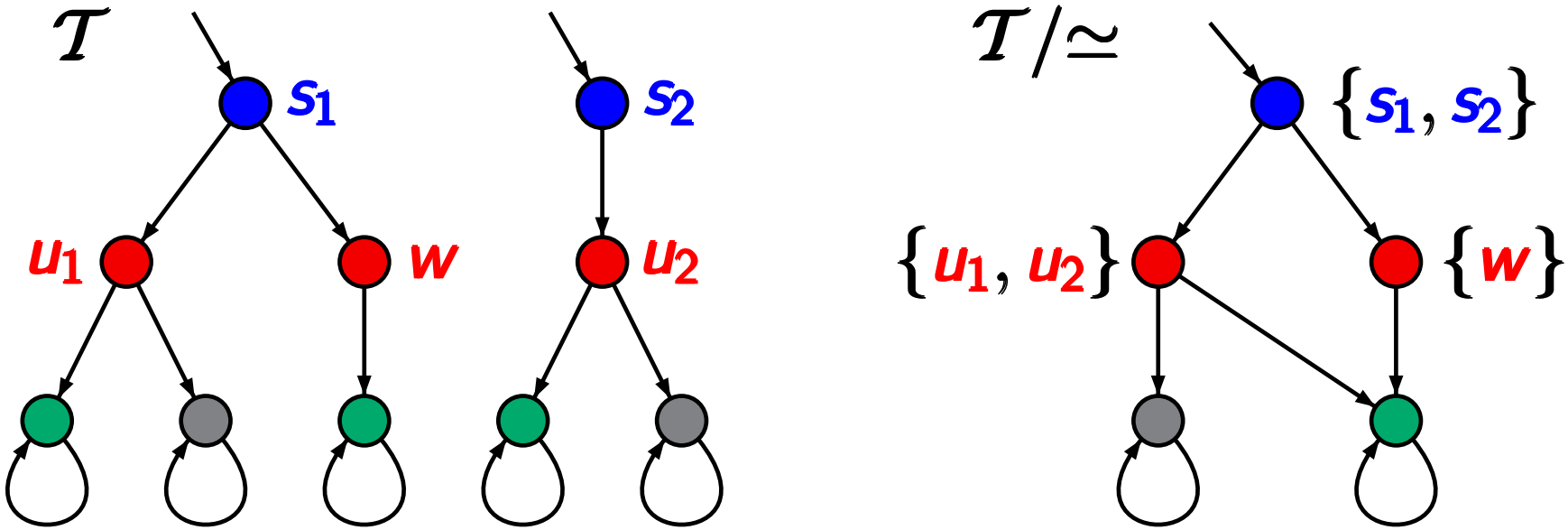
Example: simulation quotient



simulation for $(\mathcal{T}, \mathcal{T}/\simeq)$:

$$\{(s, [s]) : s \text{ is a state in } \mathcal{T}\}$$

Example: simulation quotient



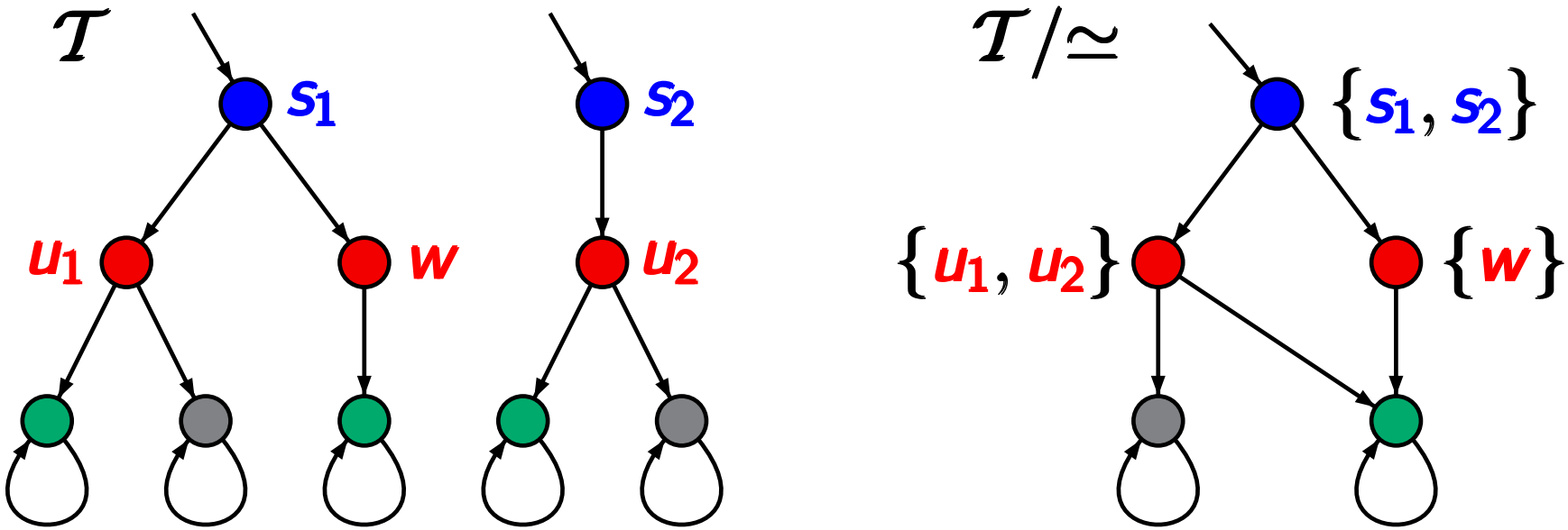
simulation for $(\mathcal{T}, \mathcal{T}/\simeq)$:

$$\{(s, [s]) : s \text{ is a state in } \mathcal{T}\}$$

but $\{([s], s) : s \text{ is a state in } \mathcal{T}\}$

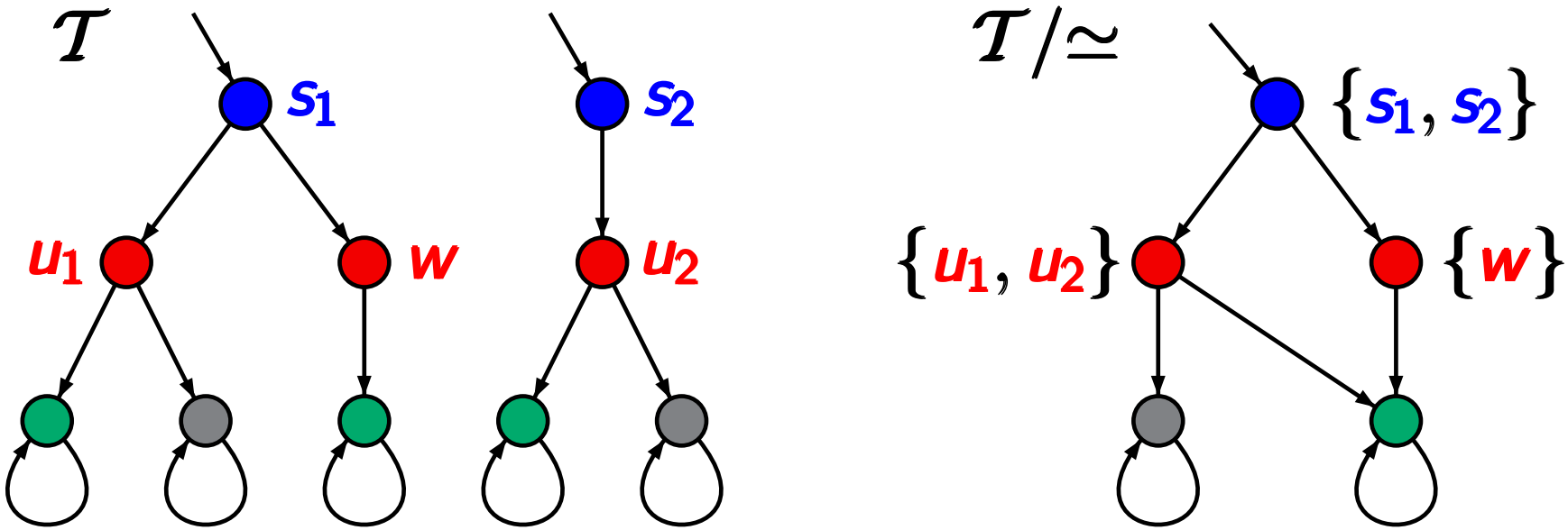
is not a simulation for $(\mathcal{T}/\simeq, \mathcal{T})$

Example: simulation quotient



show that $\mathcal{R} = \{([s], s) : s \text{ is a state in } \mathcal{T}\}$
is not a simulation for $(\mathcal{T}/\simeq, \mathcal{T})$

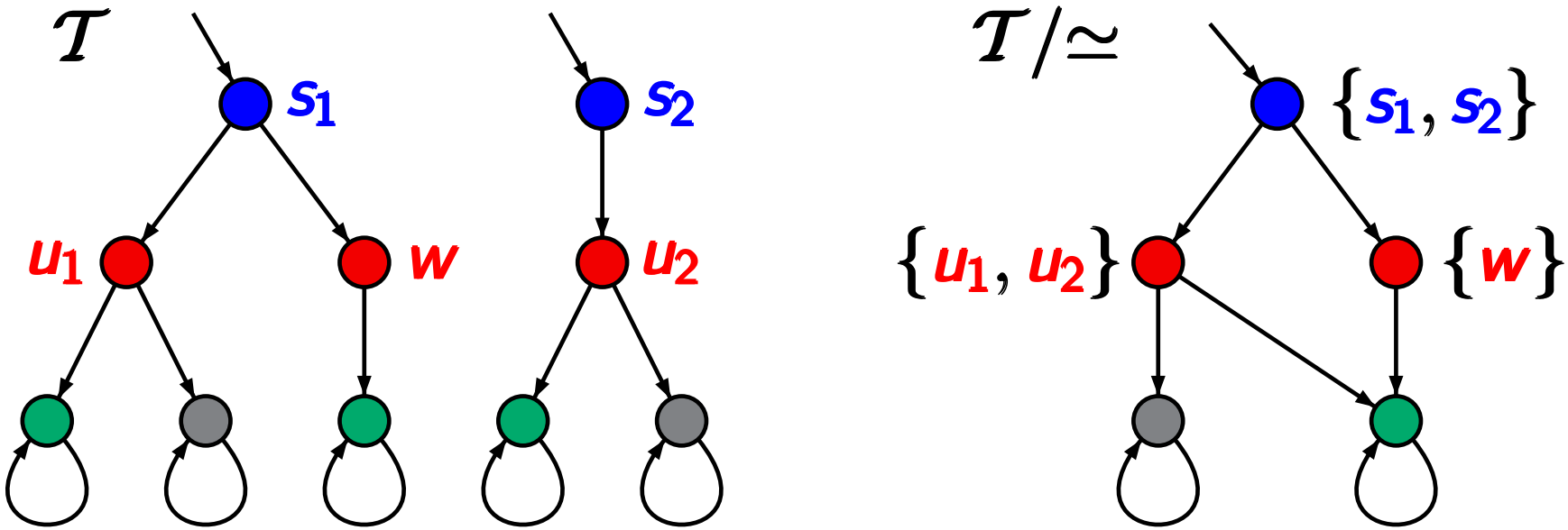
Example: simulation quotient



show that $\mathcal{R} = \{([s], s) : s \text{ is a state in } \mathcal{T}\}$
is not a simulation for $(\mathcal{T}/\simeq, \mathcal{T})$

regard $(\{s_1, s_2\}, s_2) \in \mathcal{R}$

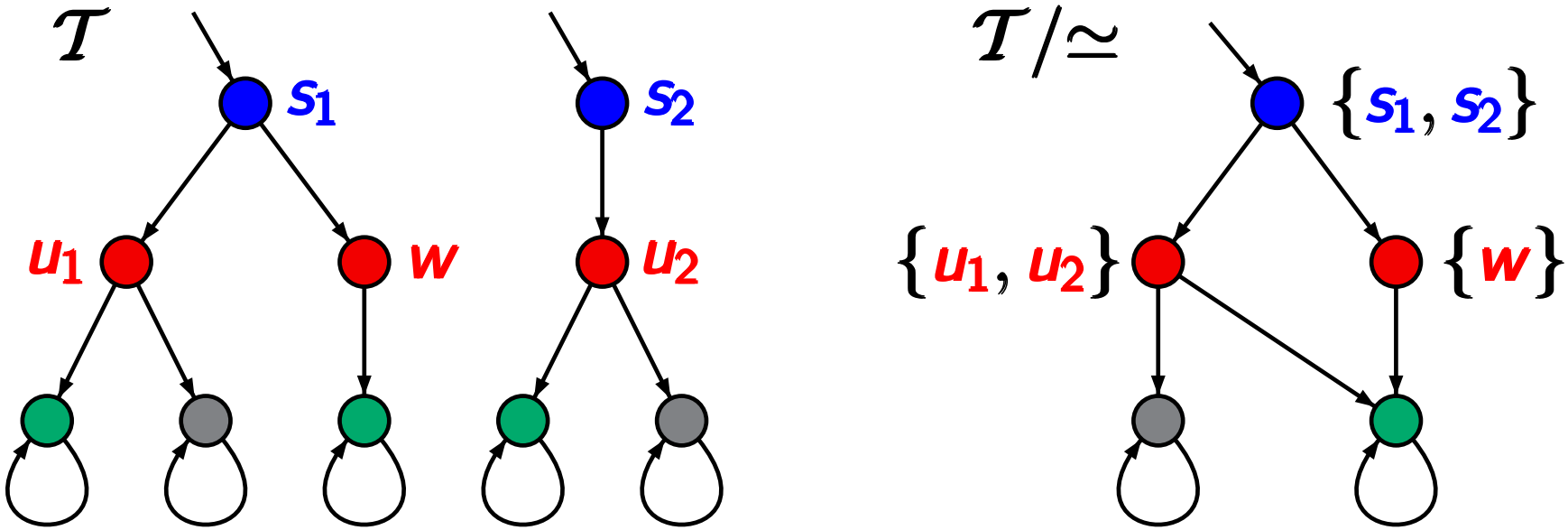
Example: simulation quotient



show that $\mathcal{R} = \{([s], s) : s \text{ is a state in } \mathcal{T}\}$
is not a simulation for $(\mathcal{T}/\simeq, \mathcal{T})$

regard $(\{s_1, s_2\}, s_2) \in \mathcal{R}$ and $\{s_1, s_2\} \rightarrow_{\simeq} \{w\}$

Example: simulation quotient



show that $\mathcal{R} = \{([s], s) : s \text{ is a state in } \mathcal{T}\}$
is not a simulation for $(\mathcal{T}/\simeq, \mathcal{T})$

regard $(\{s_1, s_2\}, s_2) \in \mathcal{R}$ and $\{s_1, s_2\} \rightarrow_{\simeq} \{w\}$

there is no transition $s_2 \rightarrow w'$ in $\mathcal{T}$ s.t. $(\{w\}, w') \in \mathcal{R}$

Similarity of $\mathcal{T}$ and $\mathcal{T}/\simeq$

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

$$\mathcal{T}/\simeq \;=\; (S/\simeq, Act', \rightarrow_{\simeq}, S_0', AP, L')$$

where the transitions are given by $\dfrac{s \longrightarrow s'}{[s] \longrightarrow_{\simeq} [s']}$

$\mathcal{T}$ and $\mathcal{T}/\simeq$ are simulation equivalent, i.e.,
$\mathcal{T} \preceq \mathcal{T}/\simeq$ and $\mathcal{T}/\simeq \preceq \mathcal{T}$

Proof. provide simulations for $(\mathcal{T}, \mathcal{T}/\simeq)$ and $(\mathcal{T}/\simeq, \mathcal{T})$

simulation for $(\mathcal{T}, \mathcal{T}/\simeq)$: $\{(s, [s]) : s \in S\}$

simulation for $(\mathcal{T}/\simeq, \mathcal{T})$: ?

Similarity of $\mathcal{T}$ and $\mathcal{T}/\simeq$

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

$$\mathcal{T}/\simeq \;=\; (S/\simeq, Act', \rightarrow_{\simeq}, S_0', AP, L')$$

where the transitions are given by $\dfrac{s \longrightarrow s'}{[s] \longrightarrow_{\simeq} [s']}$

$\mathcal{T}$ and $\mathcal{T}/\simeq$ are simulation equivalent, i.e.,
$\mathcal{T} \preceq \mathcal{T}/\simeq$ and $\mathcal{T}/\simeq \preceq \mathcal{T}$

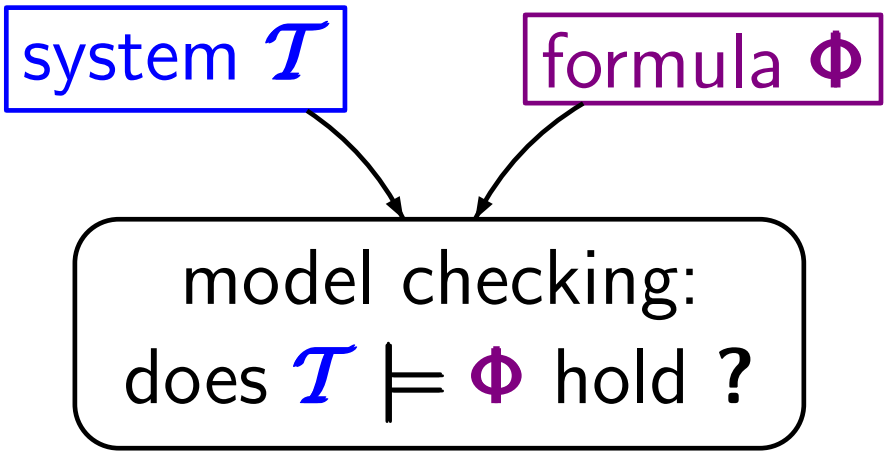
Proof. provide simulations for $(\mathcal{T}, \mathcal{T}/\simeq)$ and $(\mathcal{T}/\simeq, \mathcal{T})$

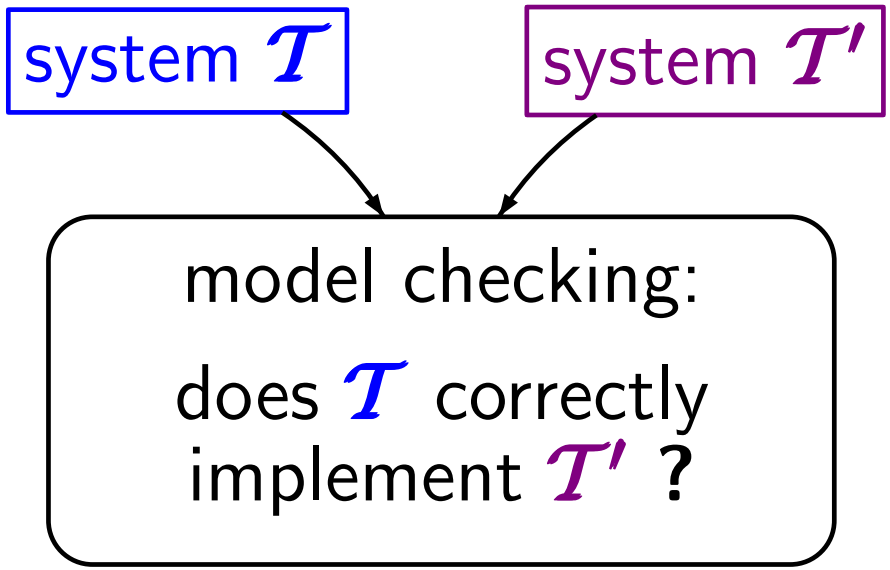
simulation for $(\mathcal{T}, \mathcal{T}/\simeq)$: $\{(s, [s]) : s \in S\}$

simulation for $(\mathcal{T}/\simeq, \mathcal{T})$: $\{([s], t) : s \preceq_{\mathcal{T}} t\}$

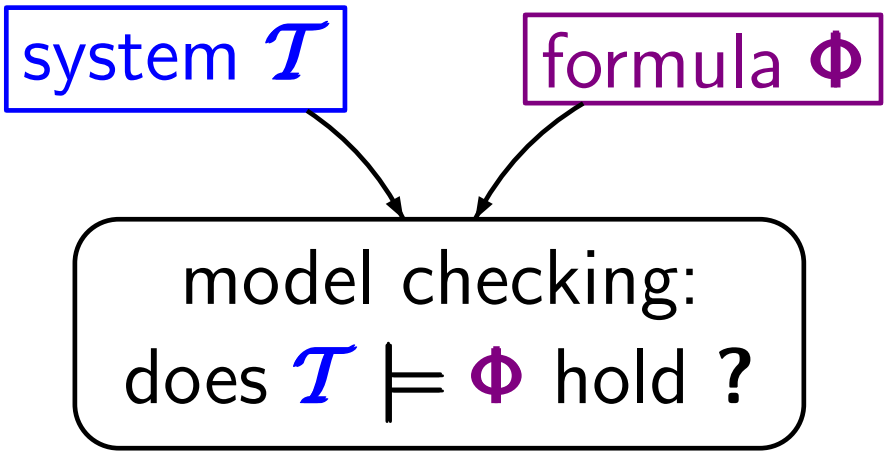
Heterogeneous/homogeneous model checking GRM5.5-30

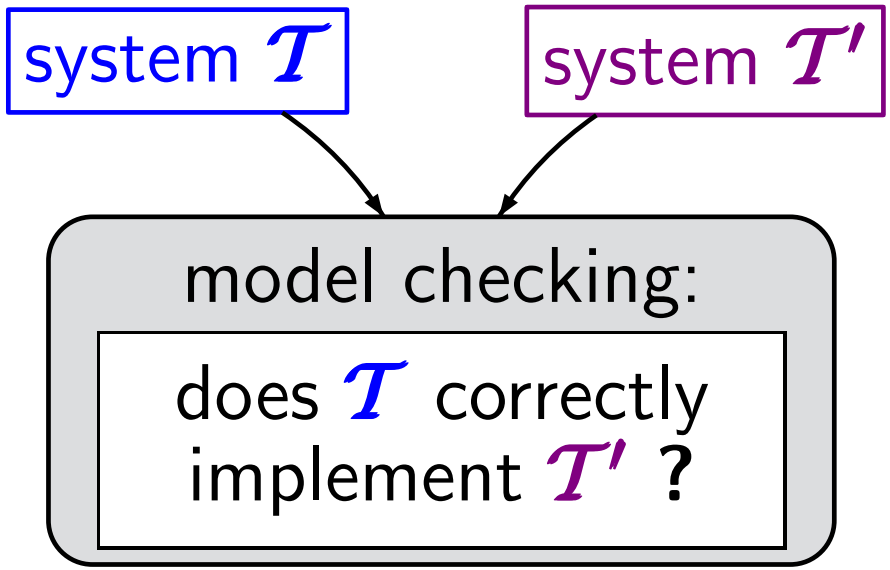
Heterogeneous/homogeneous model checking





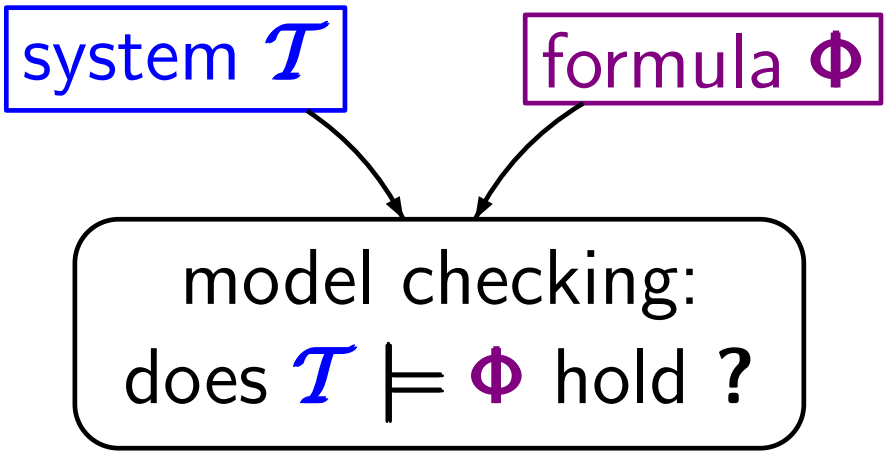
Heterogeneous/homogeneous model checking

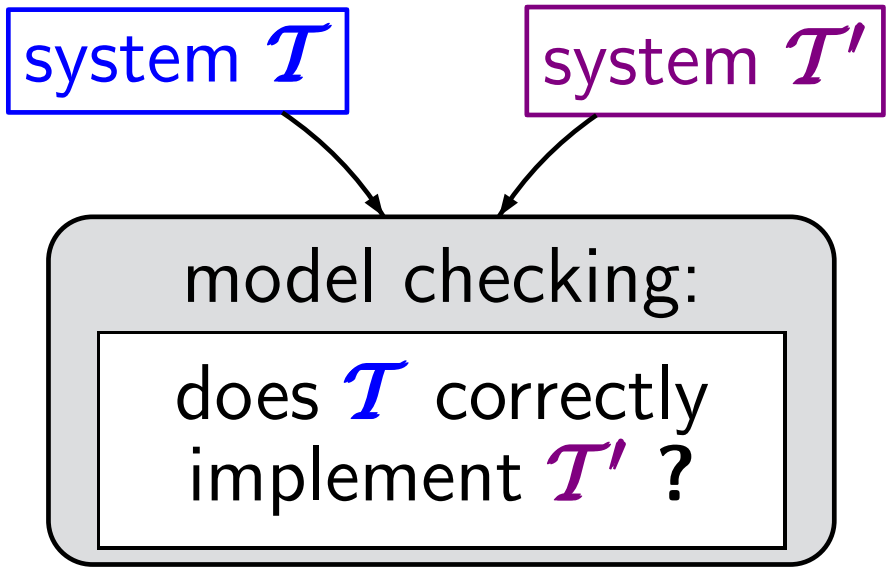




trace inclusion checking
trace equivalence checking

Heterogeneous/homogeneous model checking

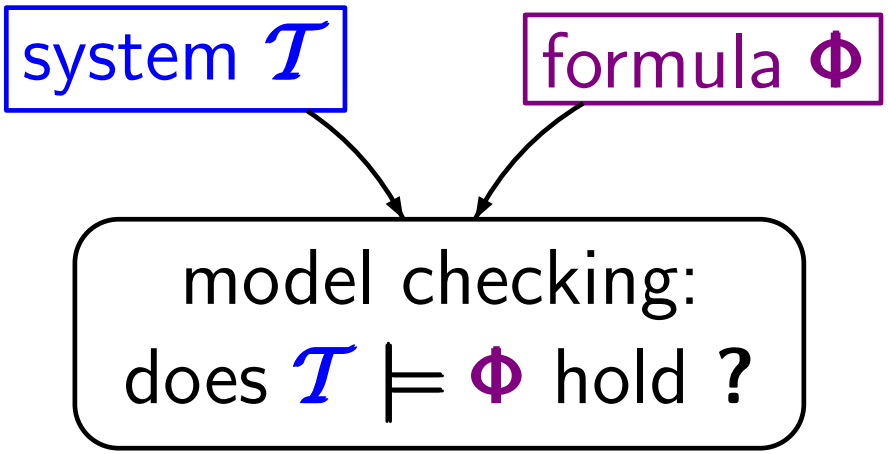


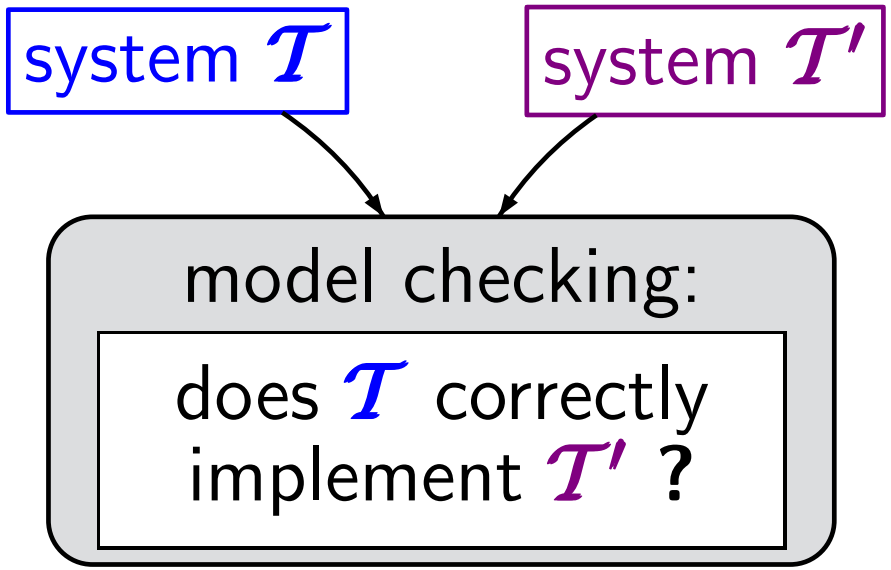


trace inclusion checking
trace equivalence checking

$\longleftarrow$ ***PSPACE***-complete

Heterogeneous/homogeneous model checking





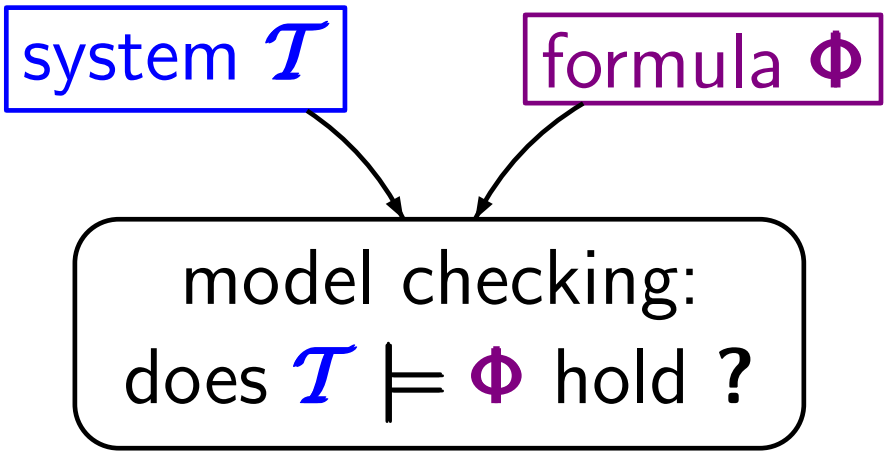
trace inclusion checking
trace equivalence checking

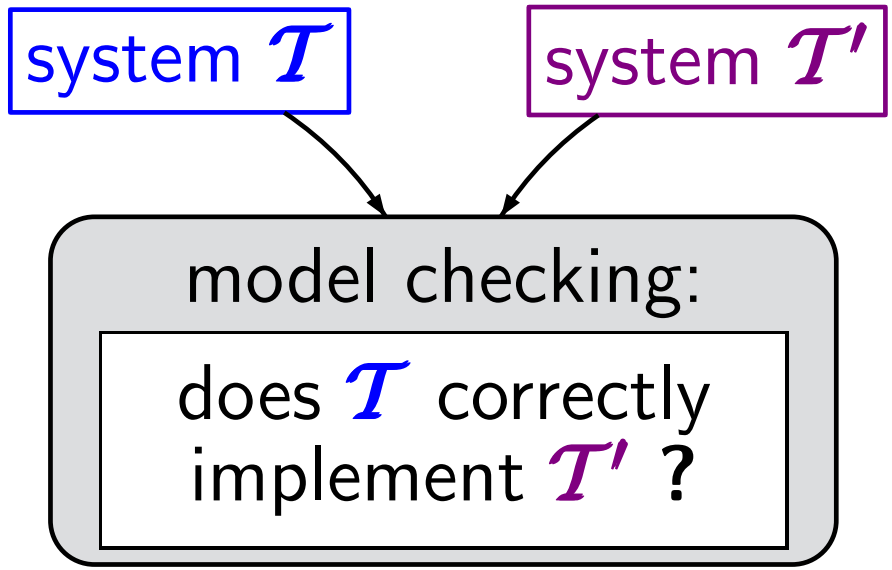
⟵ ***PSPACE***-complete

bisimulation equivalence checking

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

Heterogeneous/homogeneous model checking



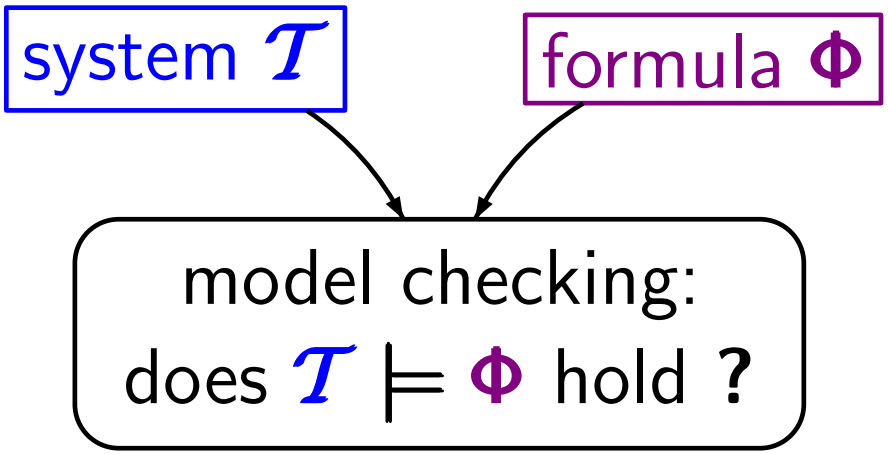


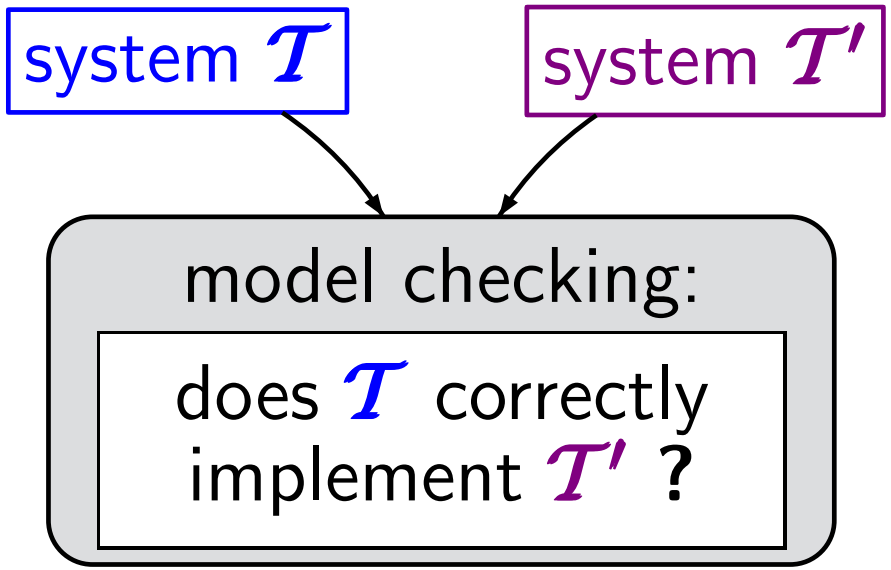
trace inclusion checking
trace equivalence checking
$\longleftarrow$ ***PSPACE***-complete

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

n = #states
m = #transitions

Heterogeneous/homogeneous model checking





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

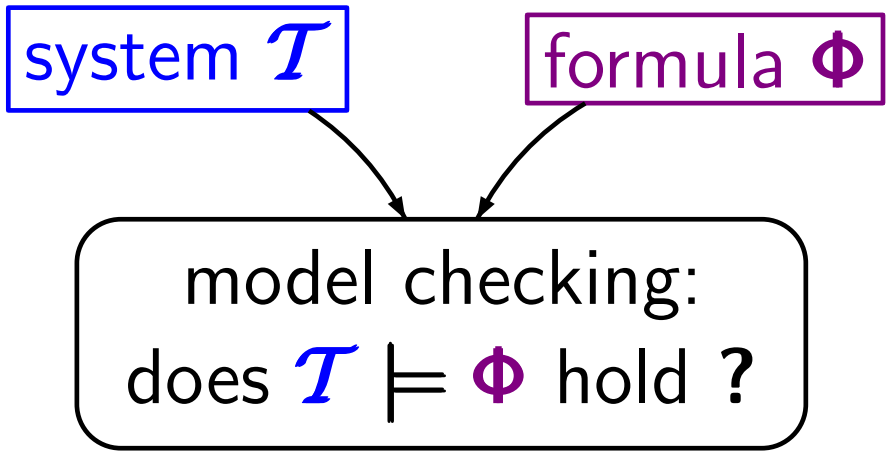
bisimulation equivalence checking
$\longleftarrow$ $\mathcal{O}(m \cdot \log n)$

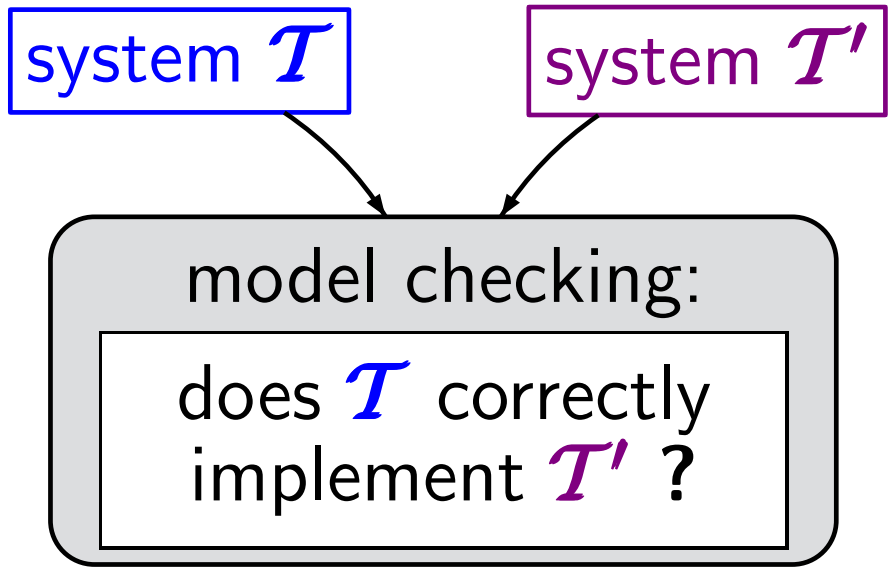
"does $\mathcal{T} \sim \mathcal{T}'$ hold ?"

refinement checking via simulation

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

Heterogeneous/homogeneous model checking





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

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

refinement checking via simulation
"does $\mathcal{T} \preceq \mathcal{T}'$ hold ?"
$\longleftarrow$ $\mathcal{O}(m \cdot n)$

Refinement checking via simulation

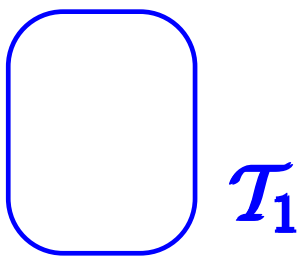
given: **2** finite transition system $\mathcal{T}_1$ and $\mathcal{T}_2$ over the same set of propositions AP

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

Refinement checking via simulation

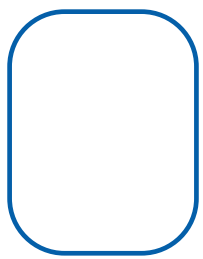
given: 2 finite transition system $\mathcal{T}_1$ and $\mathcal{T}_2$ over the same set of propositions AP

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



$\mathcal{T}_1$

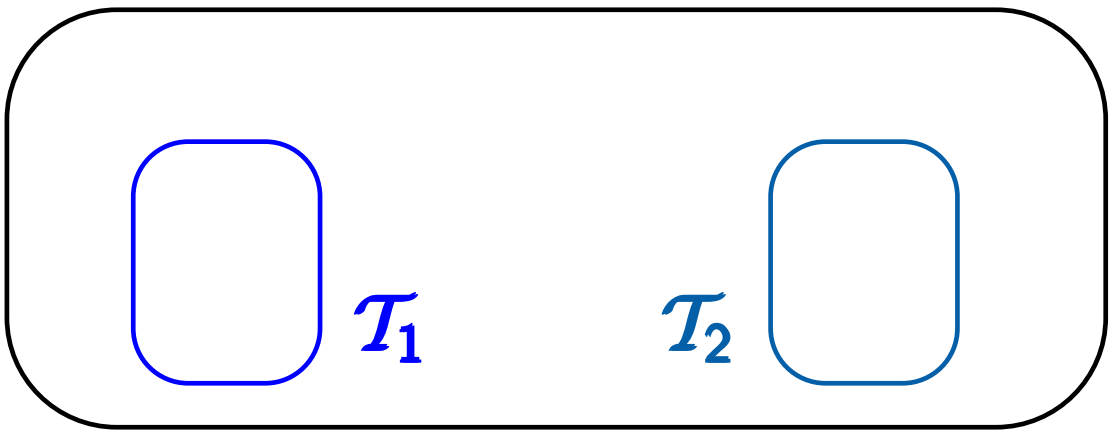
$\mathcal{T}_2$



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 ?

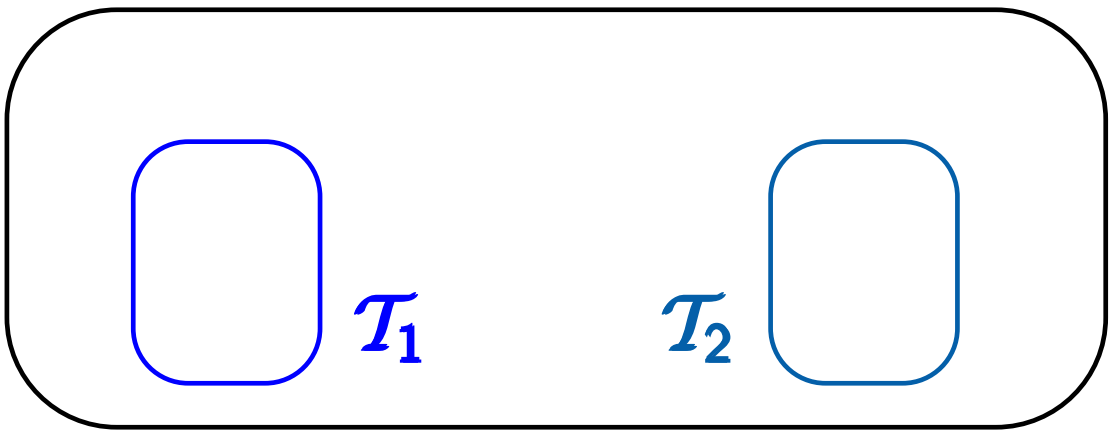


composite system $\mathcal{T} = \mathcal{T}_1 \uplus \mathcal{T}_2$

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 ?



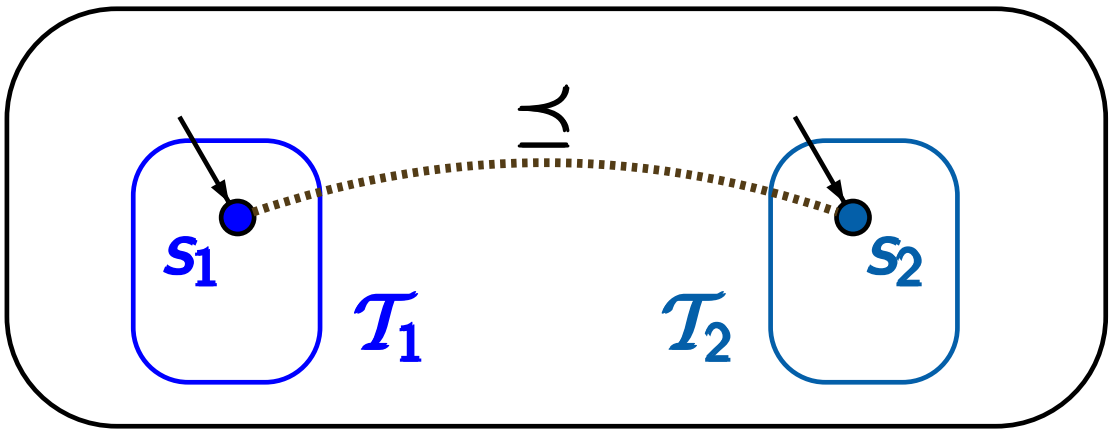
composite system $\mathcal{T} = \mathcal{T}_1 \uplus \mathcal{T}_2$

- 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 $\mathcal{T} = (S, Act, \rightarrow, S_0, AP, L)$
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 $\preceq_T$

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

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 $\mathcal{R}$

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. $s_1 \to s_1'$, but there is no transition $s_2 \to s_2'$ 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 $\mathcal{R} = \preceq_{\mathcal{T}}$

Computing the simulation preorder $\preceq_T$

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

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

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

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

OD

return $\mathcal{R}$

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

Computing the simulation preorder $\preceq_T$

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

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

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

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

OD

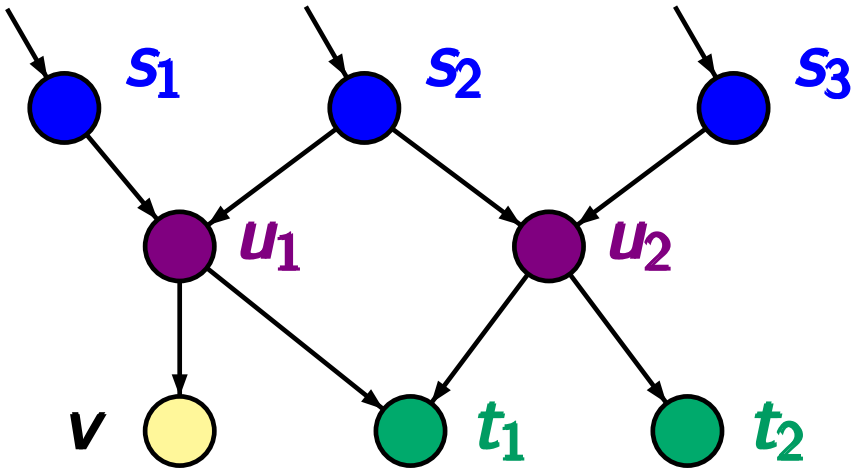
return $\mathcal{R}$

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

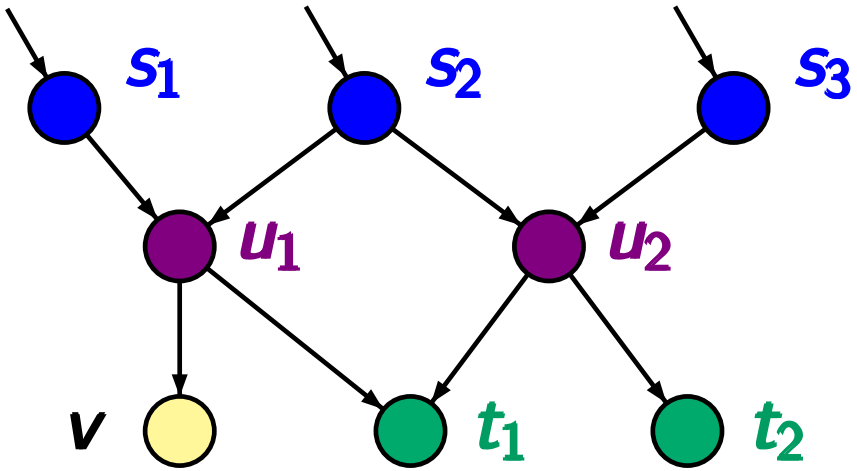
representation of $\mathcal{R}$ by simulator sets

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

Example: computation of $\preceq$



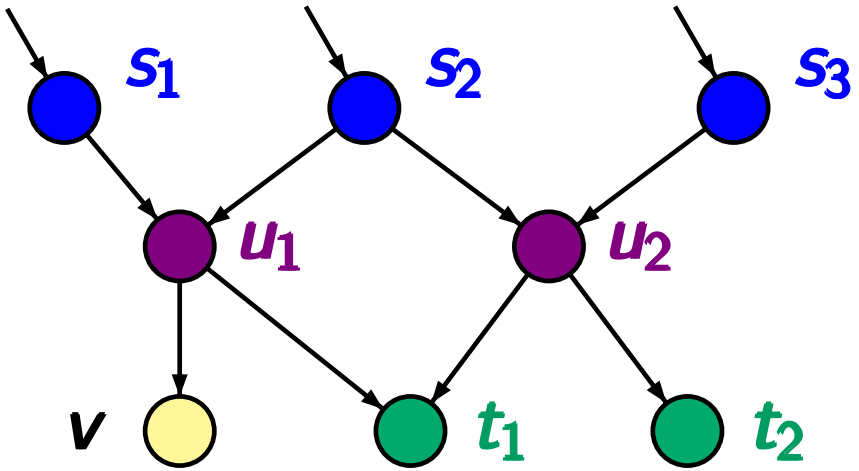
Example: computation of $\preceq$



initially:

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

Example: computation of $\preceq$

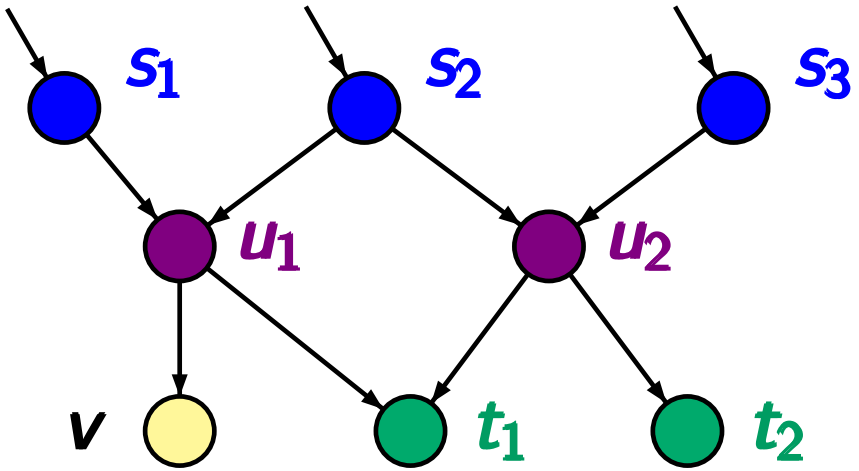


initially:

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

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

Example: computation of $\preceq$

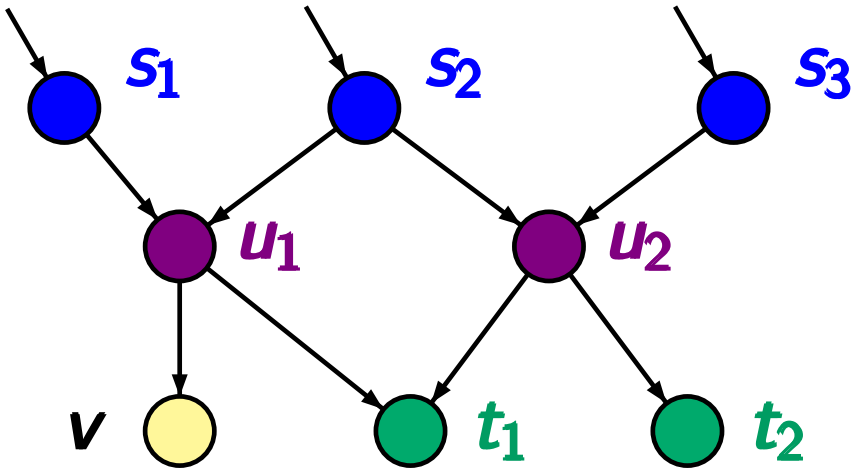


initially:

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

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

Example: computation of $\preceq$



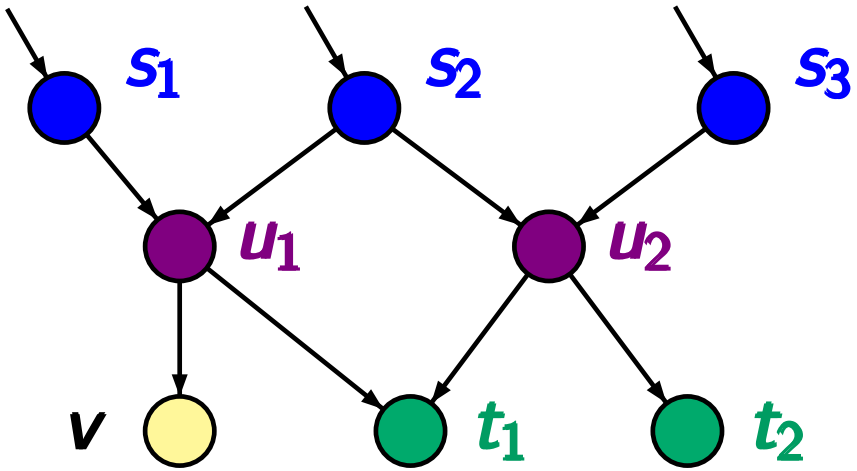
initially:

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

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

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

Example: computation of $\preceq$



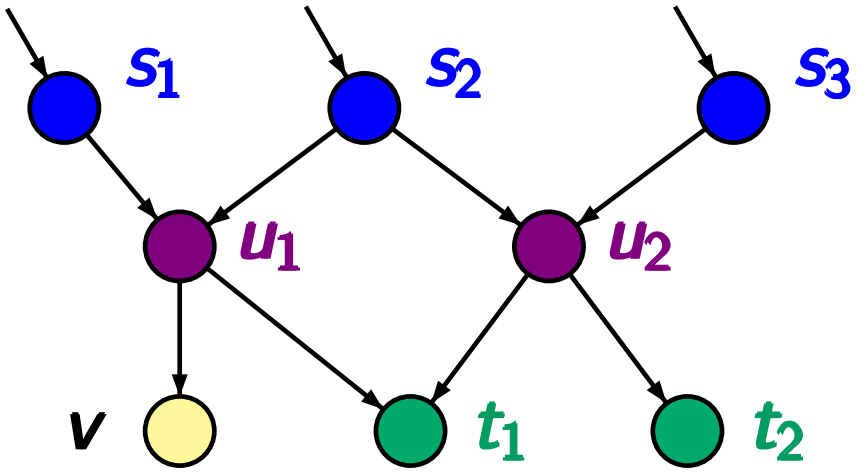
initially:

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

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

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

Example: computation of $\preceq$



initially:

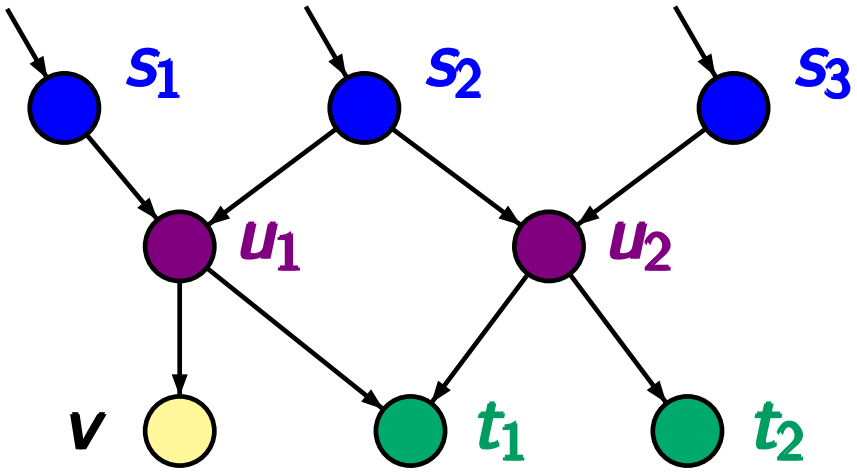
$$Sim(s_i) = \{s_1, s_2, s_3\}$$
$$Sim(u_1) = Sim(u_2) = \{u_1, u_2\}$$
$$Sim(v) = \{v\}$$
$$Sim(t_1) = Sim(t_2) = \{t_1, t_2\}$$

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

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

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

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

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

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

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

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. $Post(s_2) \cap Sim(s_1') = \emptyset$ 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 $\preceq_T$

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. $Post(s_2) \cap Sim(s_1') = \emptyset$ DO

  choose such states s_1, s_2

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

OD

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

$s_1 \longrightarrow s_1'$

s_2

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

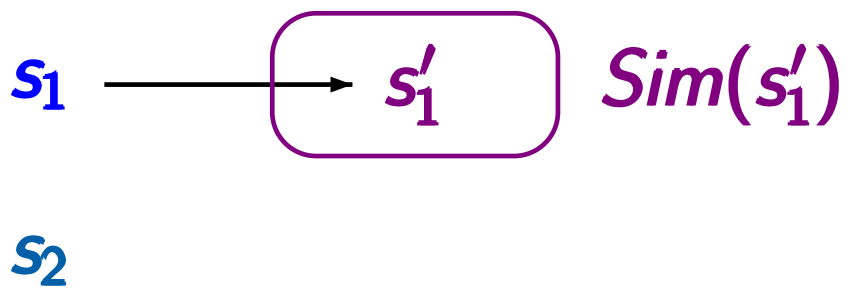
  s.t. $Post(s_2) \cap Sim(s_1') = \emptyset$ DO

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

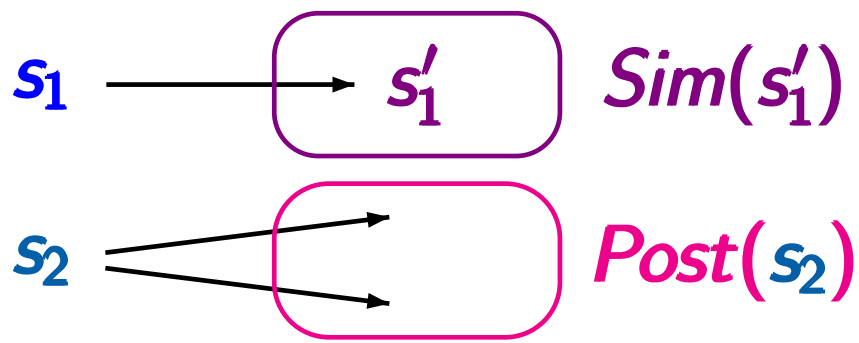
  s.t. $Post(s_2) \cap Sim(s_1') = \emptyset$ DO

  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 $\preceq_T$

FOR ALL $s_1 \in S$ DO

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

OD

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

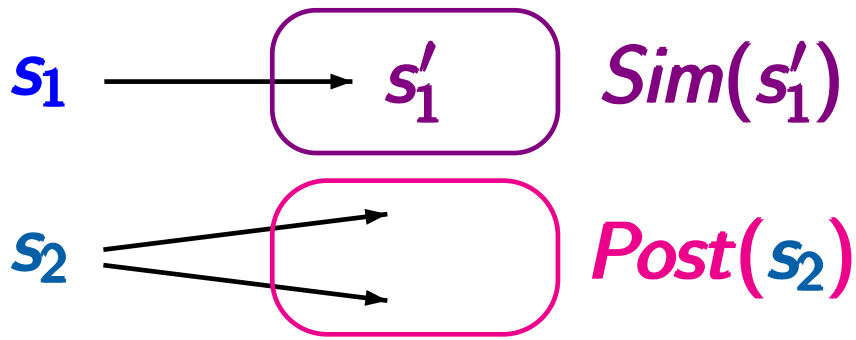
  s.t. $Post(s_2) \cap Sim(s_1') = \varnothing$ DO

  choose such states s_1, s_2

  $Sim(s_1) := Sim(s_1) \setminus \{s_2\}$ $\longleftarrow$ $s_1 \not\preceq_T s_2$

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

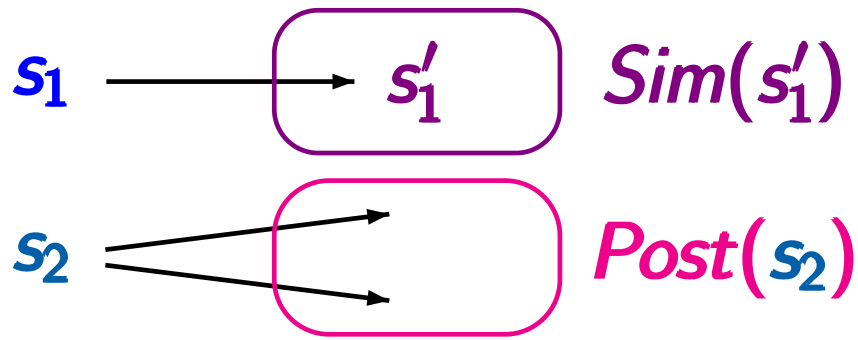
  choose such states s_1, s_2

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

OD

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

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

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

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

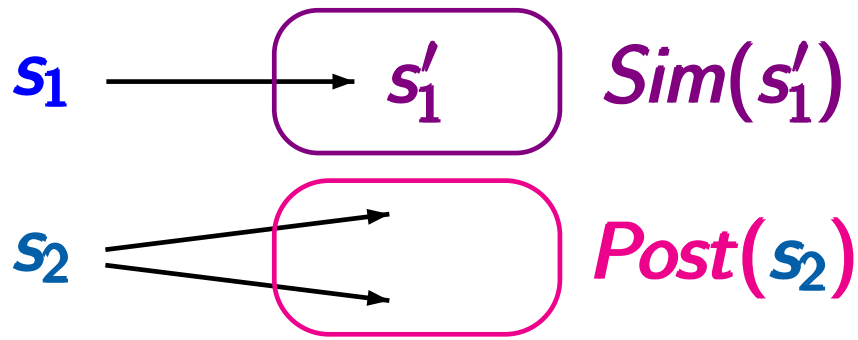
  choose such states s_1, s_2

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

OD

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

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



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

$\mathcal{O}(m \cdot |S|^2)$-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 $s_1 \in S$ 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

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 $s_1 \in S$ 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

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

  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

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 $s_1 \in S$ 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

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

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

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

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

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 $V \neq \emptyset$ 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 ⟵ #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. $s_2 \in \mathit{Sim}(s_1)$ and $s_2' \in \mathit{Sim}(s_1')$.

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. $s_2 \in \mathit{Sim}(s_1)$ and $s_2' \in \mathit{Sim}(s_1')$.
- suppose that s_2' will be removed from $\mathit{Sim}(s_1')$.

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. $s_2 \in \mathit{Sim}(s_1)$ and $s_2' \in \mathit{Sim}(s_1')$.
- suppose that s_2' will be removed from $\mathit{Sim}(s_1')$.

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

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

... 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. $s_2 \in \mathit{Sim}(s_1)$ and $s_2' \in \mathit{Sim}(s_1')$.
- suppose that s_2' will be removed from $\mathit{Sim}(s_1')$.

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

idea: collect all such states s_2 in $\mathit{Remove}(s_1')$

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

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

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

$Sim_{old}(s_1')$

Idea of the HHK-algorithm

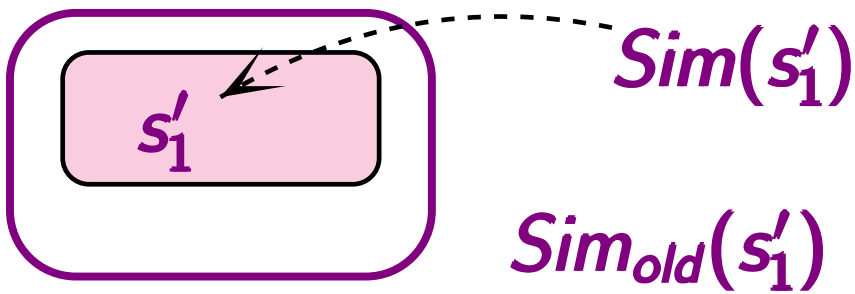
If s_2' is removed from $Sim(s_1')$ then regard all direct predecessors s_1 of s_1' and remove all states in

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

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

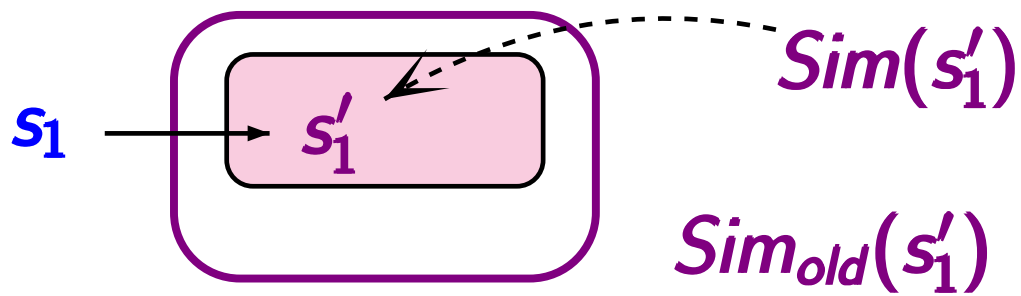
If s_2' is removed from $Sim(s_1')$ then regard all direct predecessors s_1 of s_1' and remove all states in

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

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

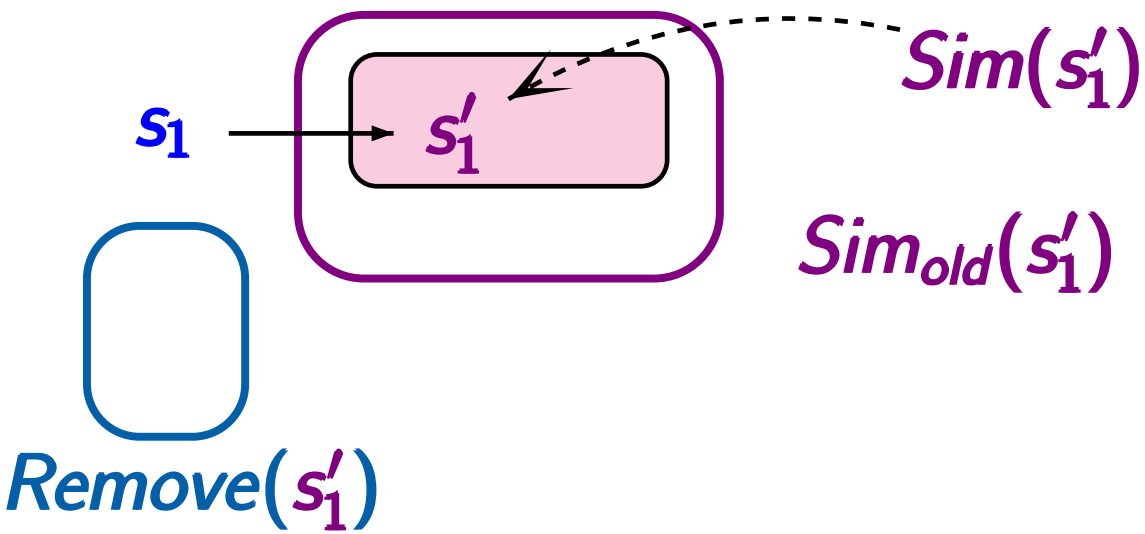
If s_2' is removed from $Sim(s_1')$ then regard all direct predecessors s_1 of s_1' and remove all states in

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

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

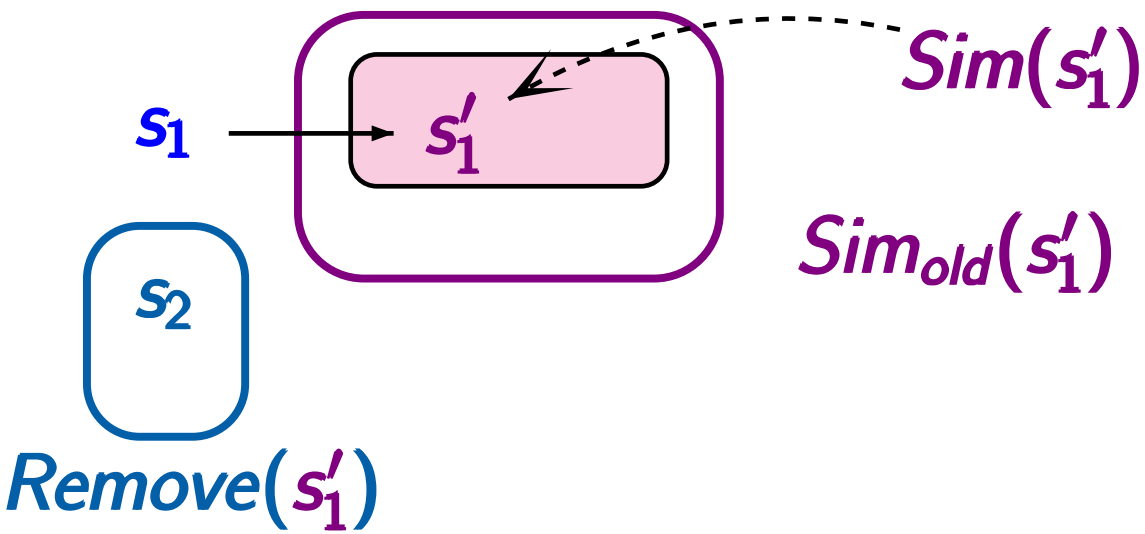
If s_2' is removed from $Sim(s_1')$ then regard all direct predecessors s_1 of s_1' and remove all states in

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

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

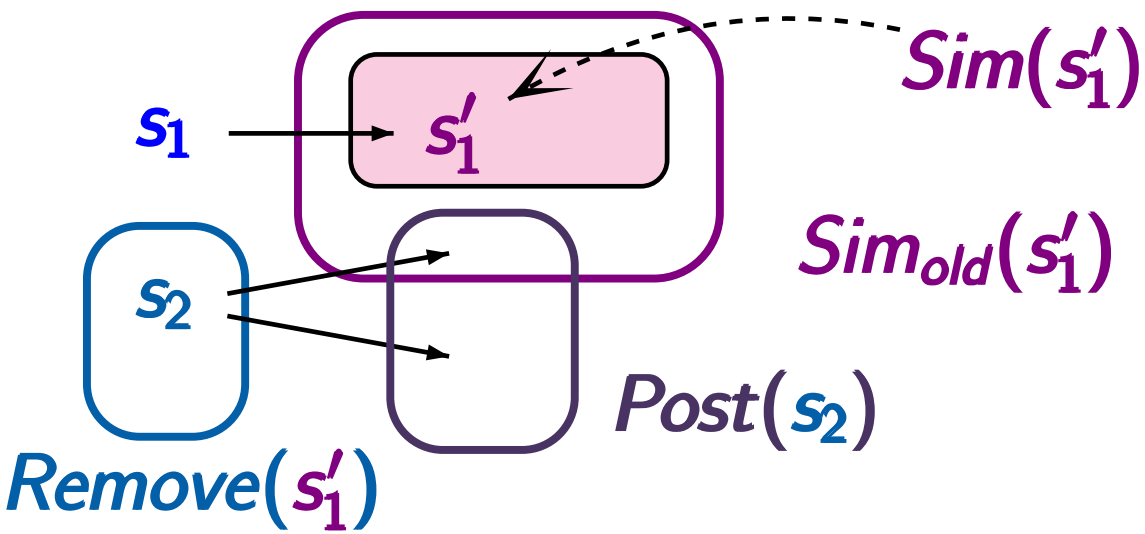
If s_2' is removed from $Sim(s_1')$ then regard all direct predecessors s_1 of s_1' and remove all states in

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

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

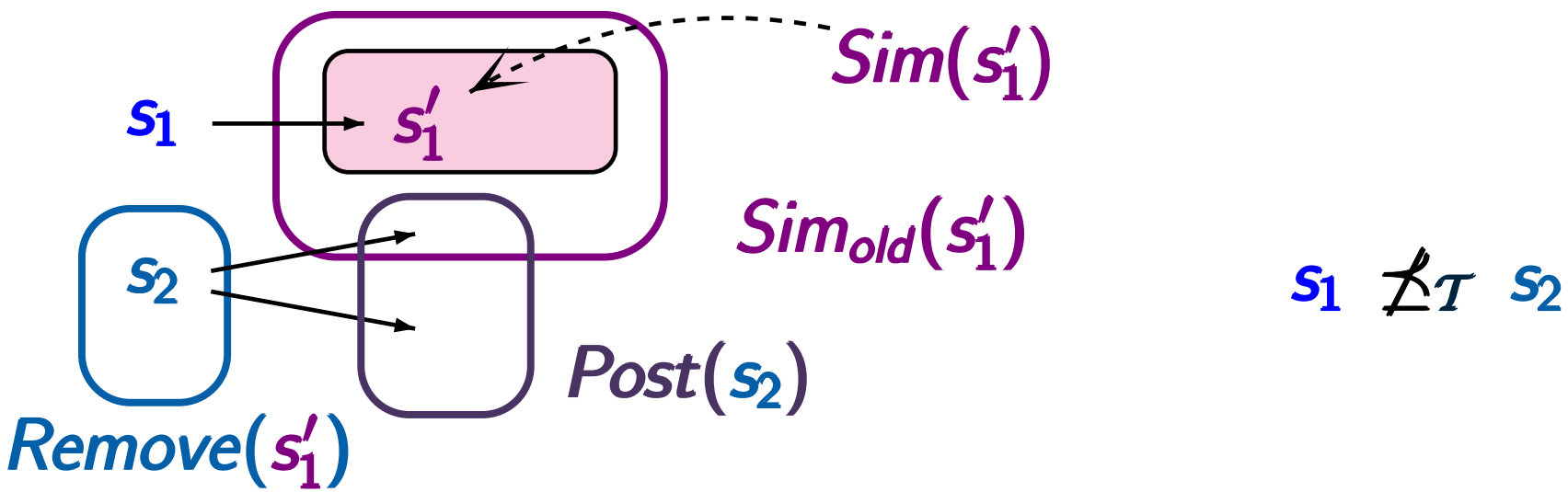
If s_2' is removed from $Sim(s_1')$ then regard all direct predecessors s_1 of s_1' and remove all states in

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

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

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

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



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

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

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

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

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

OD

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

choose such a state s_1';

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

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

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

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

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 ;

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

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

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

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

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

HHK-algorithm (first version)

if s_2' is terminal then so is s_1'

FOR ALL states s_1' DO

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

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

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

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

HHK-algorithm (first version)

FOR ALL states s_1' DO

$Sim_{old}(s_1') :=$ "undefined"

$Sim(s_1') := \{ s_2' \in s : L(s_1') = L(s_2') \text{ and } \ldots \}$

OD

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

choose such a state s_1'

IF $Sim_{old}(s_1') =$ "undefined"

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

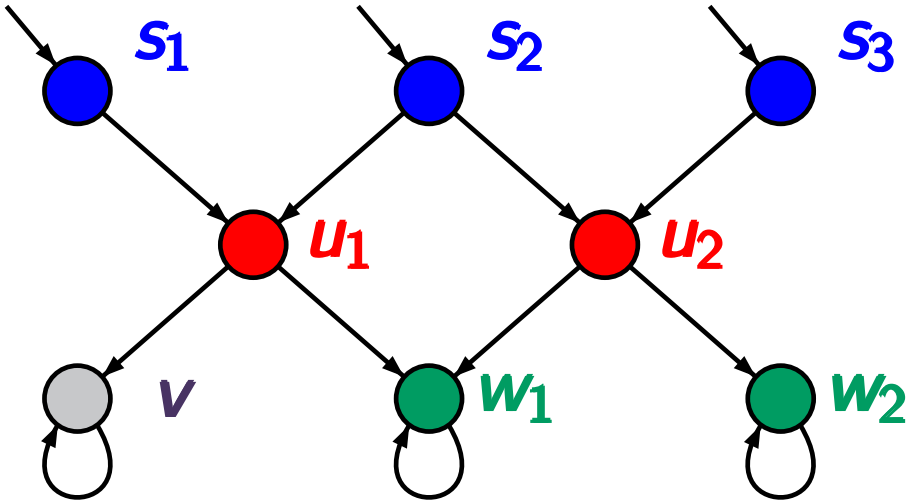
ELSE $Remove(s_1') := Pre(Sim_{old}(s_1')) \setminus Pre(Sim(s_1'))$

FI

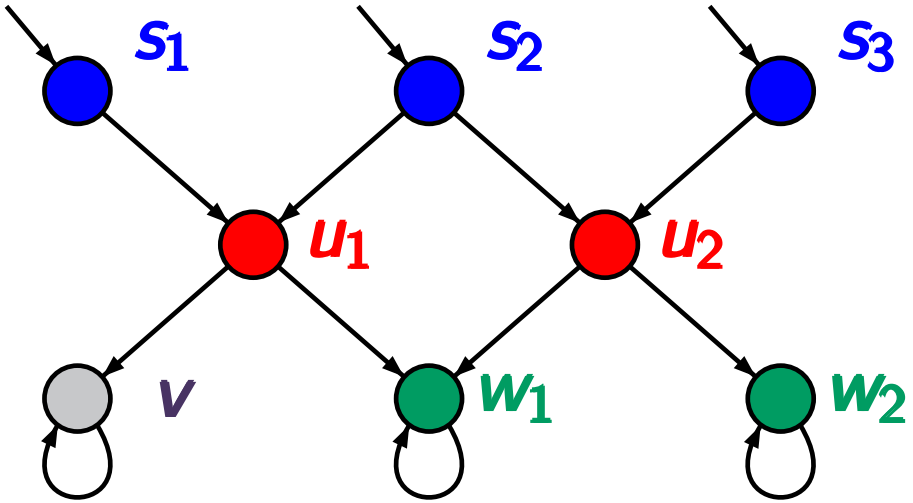
FOR ALL $s_1 \in Pre(s_1')$ DO

...

Example: HHK-algorithm



Example: HHK-algorithm



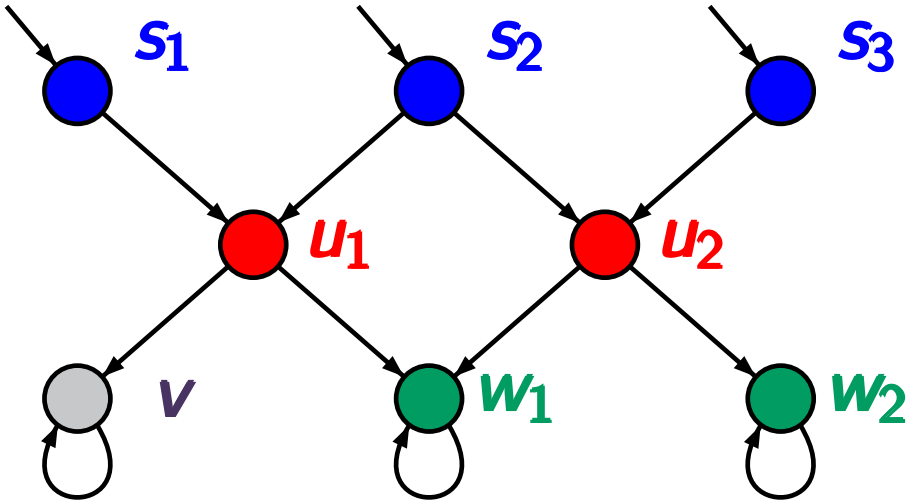
initially:

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

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

$\vdots$

Example: HHK-algorithm



initially:

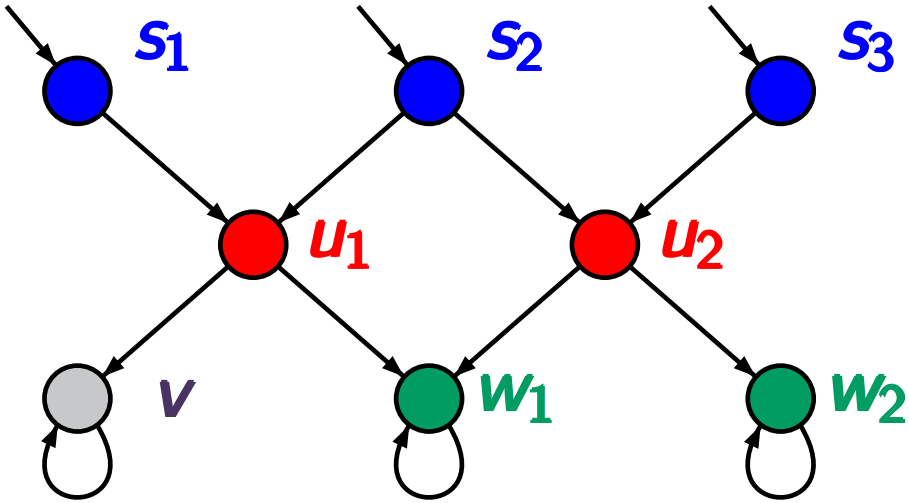
$Sim_{old}(t) = \bot$ for all states t

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

$\vdots$

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

Example: HHK-algorithm



initially:

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

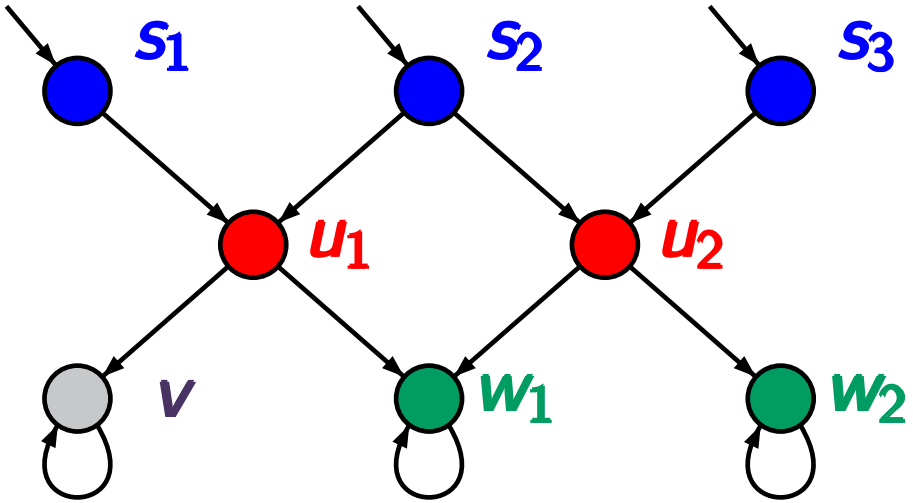
$Sim(s_1) = \{s_1, s_2, s_3\}$

$\vdots$

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:

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

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

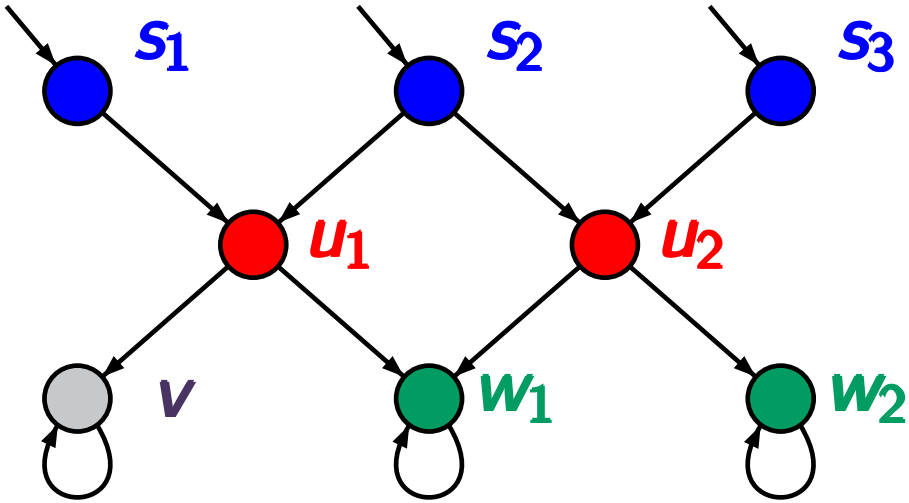
$\vdots$

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

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

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

Example: HHK-algorithm



initially:

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

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

$\vdots$

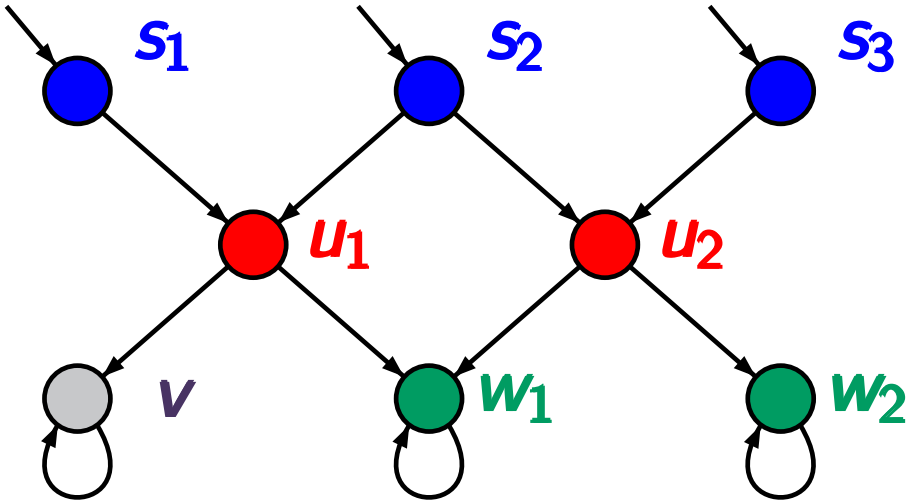
choose state $s_1' = v$ with $Sim_{old}(v) \neq Sim(v)$:

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

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

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

Example: HHK-algorithm



initially:

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

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

$\vdots$

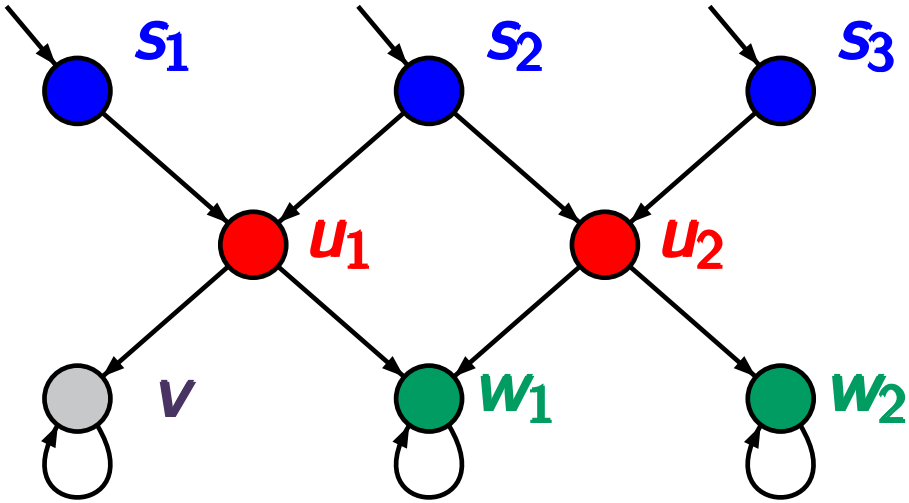
choose state $s_1' = v$ with $Sim_{old}(v) \neq Sim(v)$:

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

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

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

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 $Sim_{old}(v) \neq Sim(v)$:

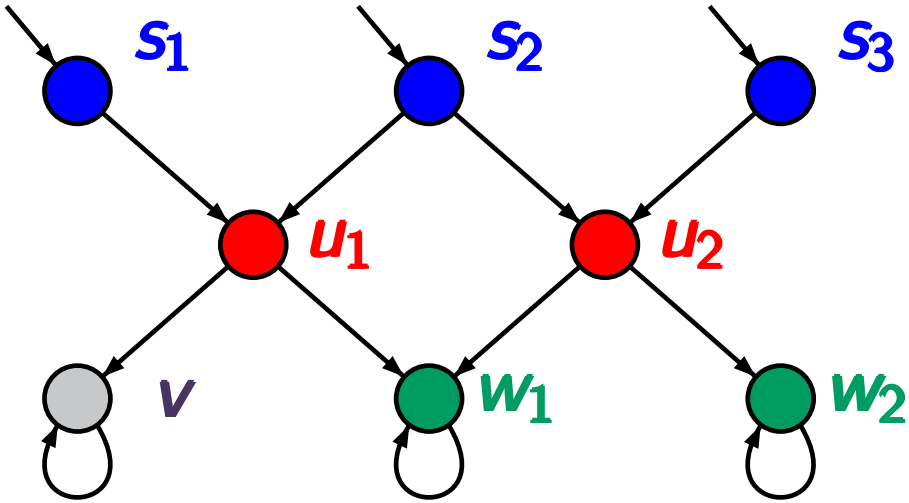
$Remove(v) = S \setminus Pre(Sim(v)) = S \setminus \{u_1\}$

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

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

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 $Sim_{old}(v) \neq Sim(v)$:

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

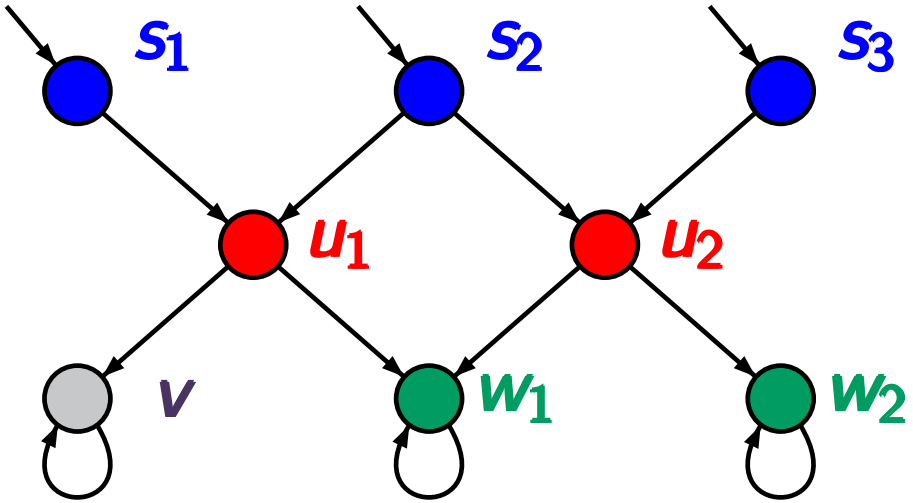
$Sim(u_1) := Sim(u_1) \setminus Remove(v) = \{u_1\}$

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

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:

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

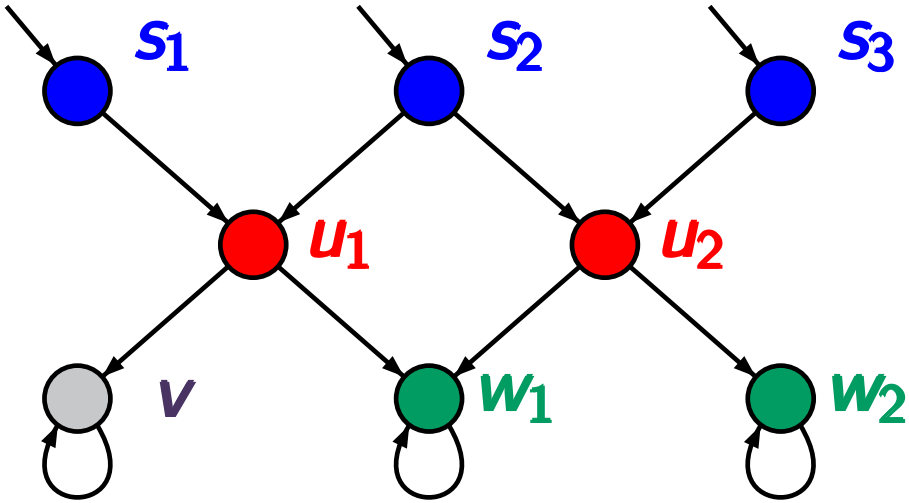
$Sim(s_1) = \{s_1, s_2, s_3\}$

$\vdots$

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

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

Example: HHK-algorithm



initially:

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

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

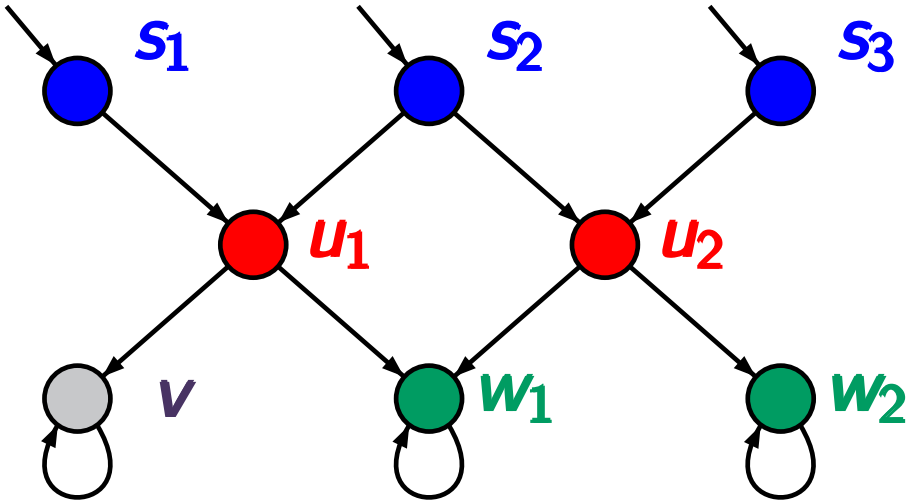
$\vdots$

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

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

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

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

$\vdots$

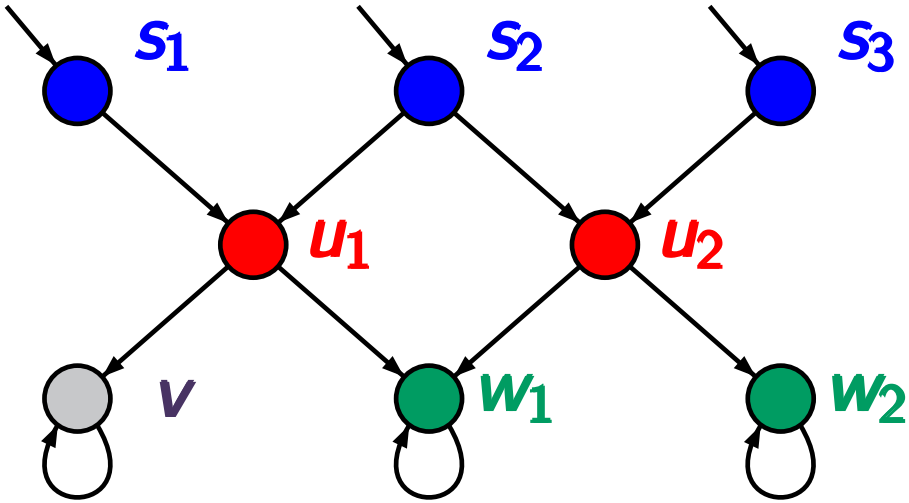
$s_1' = v$: $Sim(u_1) = \{u_1\}$, $Sim_{old}(v) = Sim(v) = \{v\}$

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

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:

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

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

$\vdots$

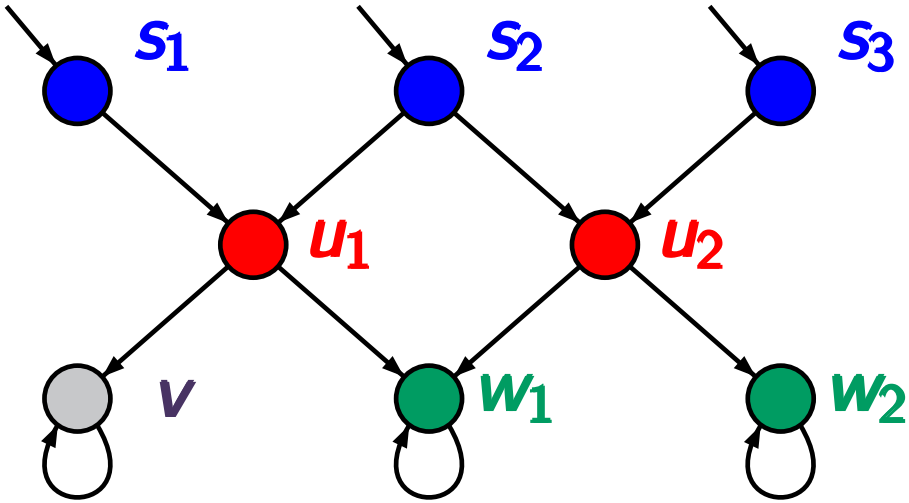
$s_1' = v$: $Sim(u_1) = \{u_1\}$, $Sim_{old}(v) = Sim(v) = \{v\}$

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

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

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

Example: HHK-algorithm



initially:

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

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

$\vdots$

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

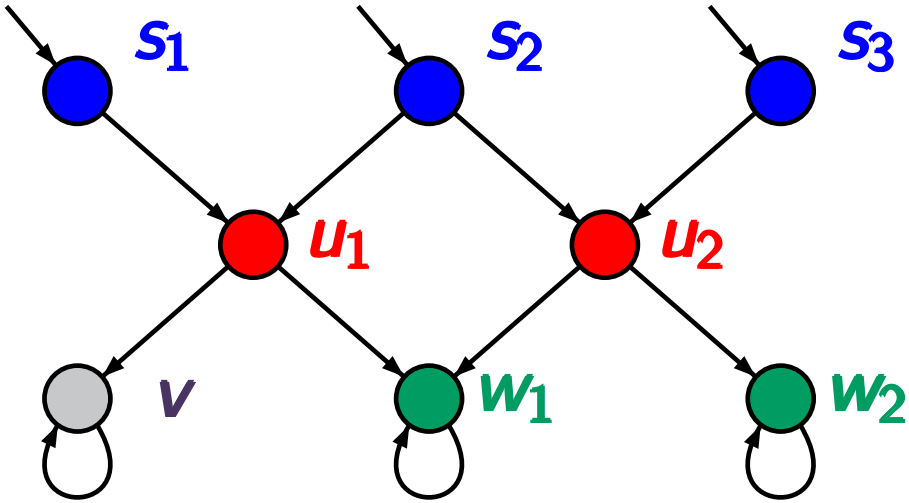
$s_1' = s_i$: $Sim_{old}(s_i) = Sim(s_i) = \{s_1, s_2, s_3\}$

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

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

$\vdots$

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

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

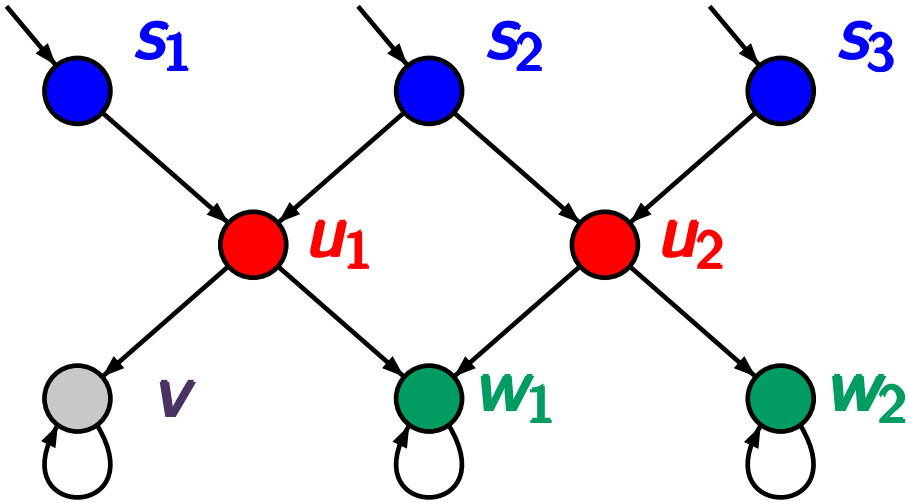
choose next state $s_1' = u_1$ with $Sim_{old}(u_1) \neq Sim(u_1)$:

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

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

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

Example: HHK-algorithm



initially:

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

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

$\vdots$

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

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

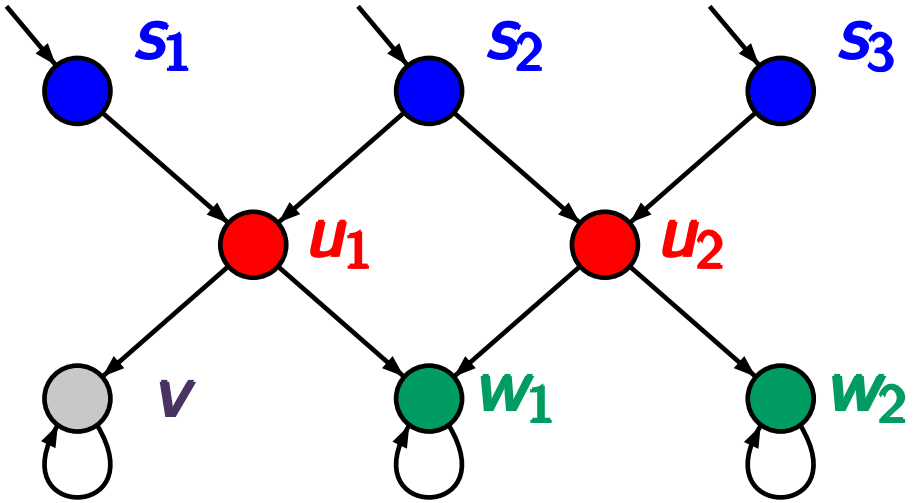
choose next state $s_1' = u_1$ with $Sim_{old}(u_1) \neq Sim(u_1)$:

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

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

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

Example: HHK-algorithm



initially:

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

$\vdots$

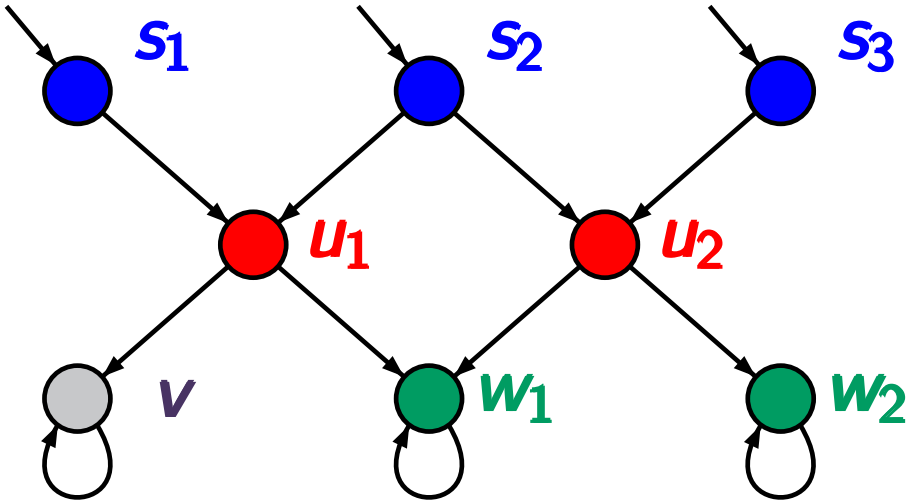
$Sim(u_2) = \{u_1, u_2\}$

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 $s_1' = u_2$:

Example: HHK-algorithm



initially:

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

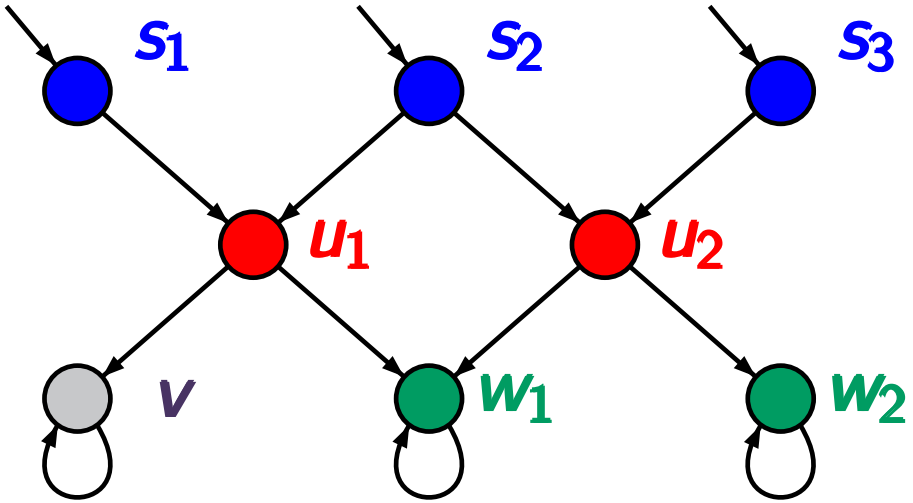
v: $Sim(u_1) = \{u_1\}$, $Sim_{old}(v) = Sim(v) = \{v\}$

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:

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

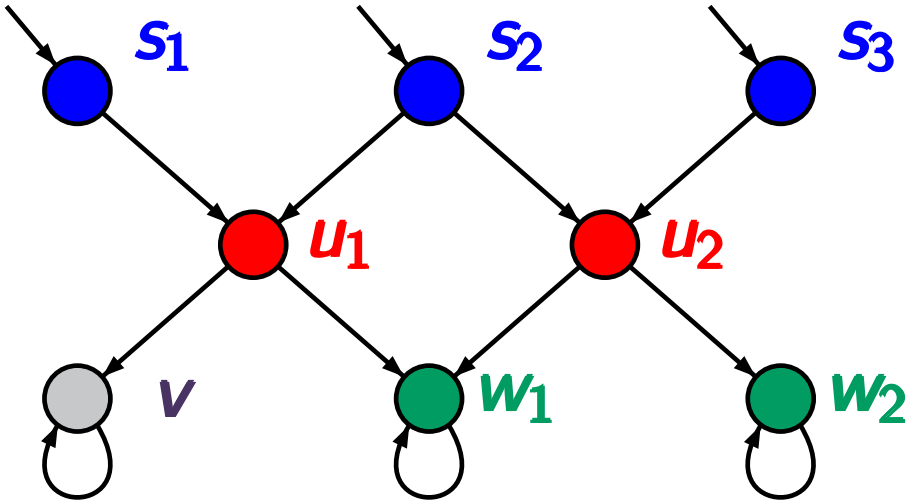
v: $Sim(u_1) = \{u_1\}$, $Sim_{old}(v) = Sim(v) = \{v\}$

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:

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

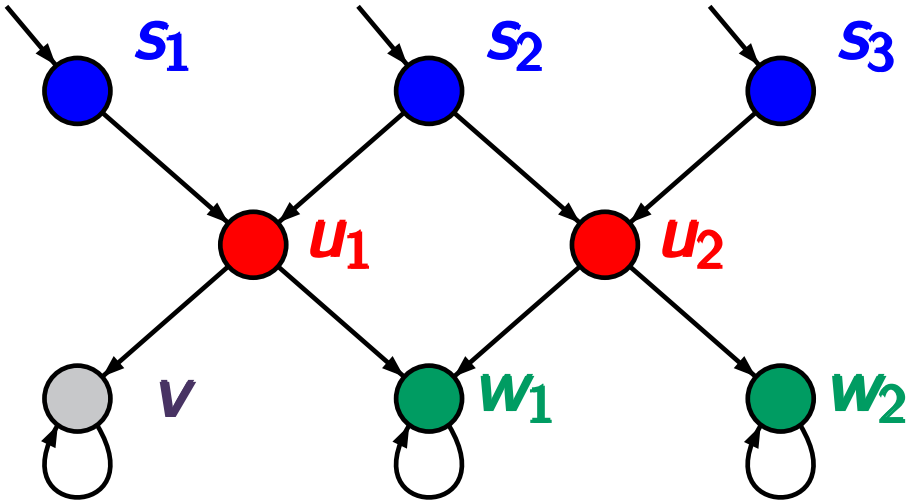
v: $Sim(u_1) = \{u_1\}$, $Sim_{old}(v) = Sim(v) = \{v\}$

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

choose state $s_1' = u_2$:

$$Remove(u_2) = S \setminus Pre(Sim(u_2)) = S \setminus \{s_1, s_2, s_3\}$$
$$Sim(s_i) := Sim(s_2) \setminus Remove(u_2) = \{s_1, s_2\},\ i = 1, 2$$

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 $\mathcal{O}(m \cdot |S|)$-algorithm for computing $\preceq_T$
- 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) $Sim(s_1') \supseteq \{s_2' \in S : s_1' \preceq_T s_2'\}$

Loop invariant of the HHK-algorithm

(1) $Sim(s_1') \supseteq \{s_2' \in S : s_1' \preceq_\tau s_2'\}$

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

Loop invariant of the HHK-algorithm

(1) $Sim(s_1') \supseteq \{s_2' \in S : s_1' \preceq_T s_2'\}$

(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) $Sim(s_1') \supseteq \{s_2' \in S : s_1' \preceq_T s_2'\}$

(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 $s_1 \not\preceq_T s_2$

Loop invariant of the HHK-algorithm

(1) $Sim(s_1') \supseteq \{s_2' \in S : s_1' \preceq_T s_2'\}$

(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 $s_1 \not\preceq_T s_2$

(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) $Sim(s_1') \supseteq \{s_2' \in S : s_1' \preceq_\tau s_2'\}$

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

Loop invariant of the HHK-algorithm

(1) $Sim(s_1') \supseteq \{s_2' \in S : s_1' \preceq_\tau s_2'\}$

(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):

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

Loop invariant of the HHK-algorithm

(1) $Sim(s_1') \supseteq \{s_2' \in S : s_1' \preceq_T s_2'\}$

(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

Loop invariant of the HHK-algorithm

(1) $Sim(s_1') \supseteq \{s_2' \in S : s_1' \preceq_T s_2'\}$

(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

since $\preceq_T$ is the coarsest simulation, (1) yields:

$$s_2' \in Sim(s_1') \quad \text{iff} \quad s_1' \preceq_T s_2'$$

HHK-algorithm (second version)

HHK-algorithm (second version)

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

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

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

OD

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

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'

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

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 $Remove(s_1') \neq \emptyset$ 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

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

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

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

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 $Remove(s_1') \neq \emptyset$ 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

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

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

OD

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

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

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

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

OD OD

HHK-algorithm (second version)

FOR ALL states s_1' DO

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

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

OD

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

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

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

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

OD OD

$Remove(s_1') := \emptyset$

DO

HHK-algorithm (second version)

$\vdots$

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

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

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

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

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

FI

$\ldots$

HHK-algorithm (second version)

$\vdots$

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

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

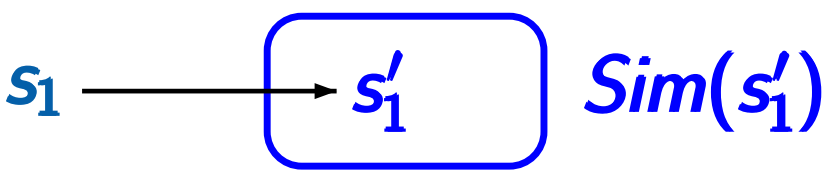
$Sim(s_1) := Sim(s_1) \setminus \{s_2\}$

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

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

FI

...



HHK-algorithm (second version)

$\vdots$

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

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

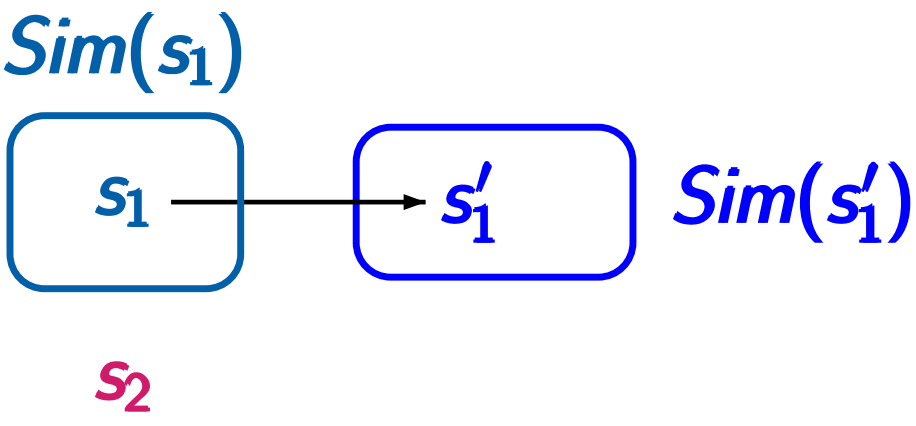
$Sim(s_1) := Sim(s_1) \setminus \{s_2\}$

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

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

FI

$\cdots$



HHK-algorithm (second version)

$\vdots$

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

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

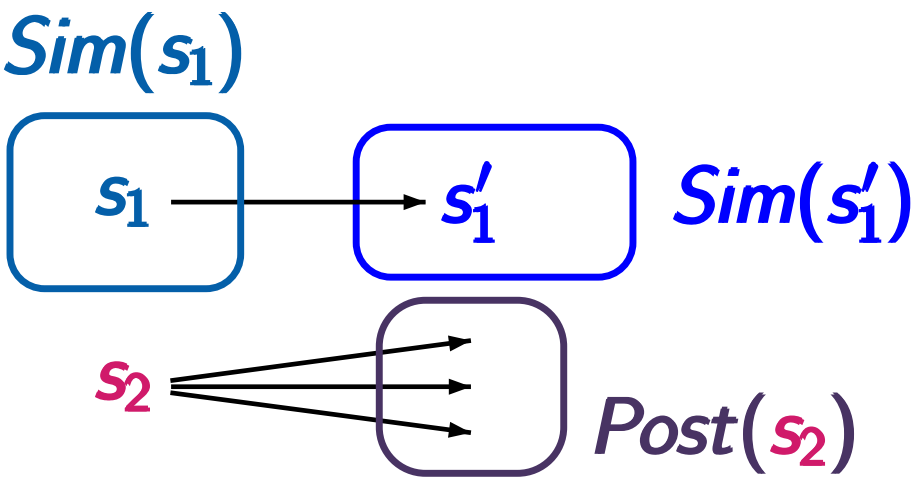
$Sim(s_1) := Sim(s_1) \setminus \{s_2\}$

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

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

FI

$\cdots$



HHK-algorithm (second version)

$\vdots$

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

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

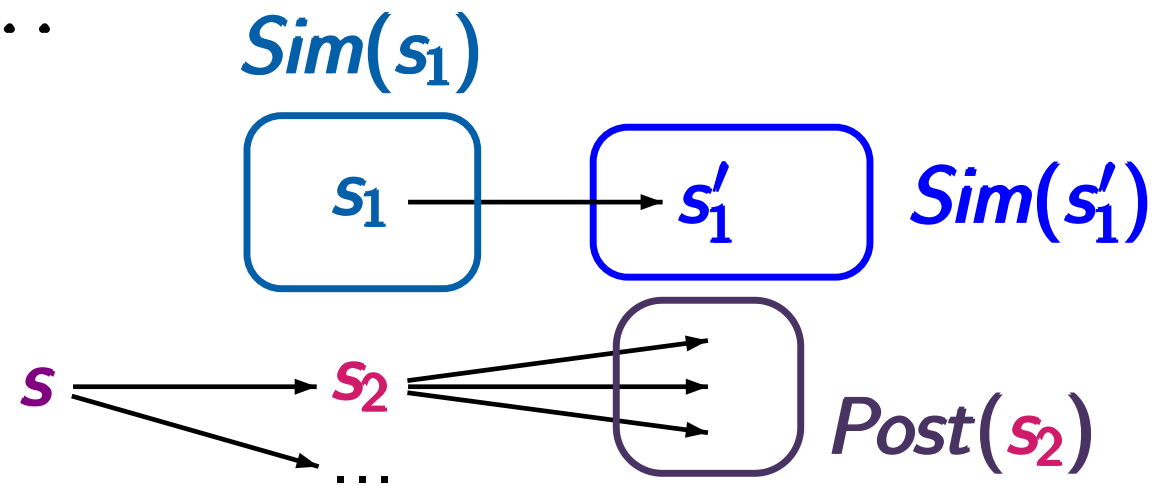
$Sim(s_1) := Sim(s_1) \setminus \{s_2\}$

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

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

FI

$\ldots$



HHK-algorithm (second version)

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

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

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

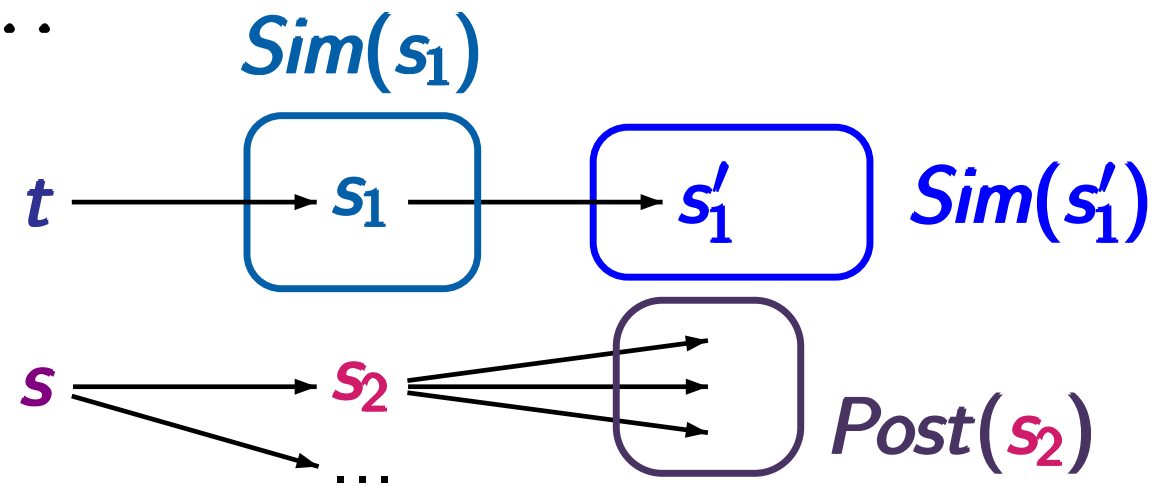
IF $s_2 \in Sim(s_1)$ THEN

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

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

FI

...



HHK-algorithm (second version)

$\vdots$

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

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

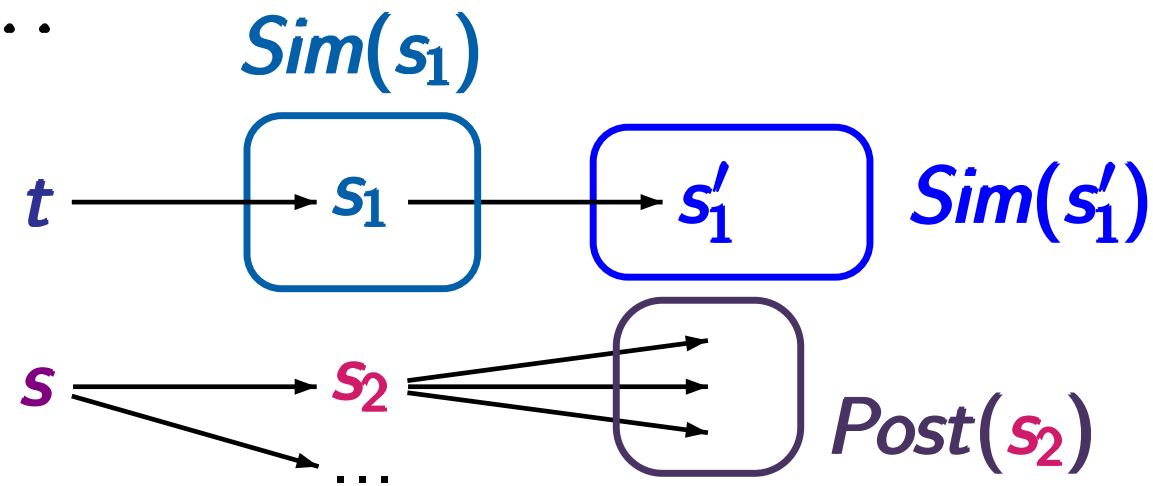
$Sim(s_1) := Sim(s_1) \setminus \{s_2\}$

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

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

FI

$\ldots$



$t \not\preceq_T s$

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

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

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

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

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

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

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

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

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

```
  ⋮
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. $s \longrightarrow s_2$ 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 s_1' with $Remove(s_1') \neq \emptyset$ 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

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

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

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

FI

OD

OD

$Remove(s_1') := \emptyset$

DO

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

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

OD

OD

$Remove(s_1') := \varnothing$

DO

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

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

$Remove(s_1') := \varnothing$

DO

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

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

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

$Remove(s_1') := \varnothing$

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

The End