LP properties for MUTEX

$AP = \{\mathbf{wait}_1, \mathbf{crit}_1, \mathbf{wait}_2, \mathbf{crit}_2\}$

safety:

$$MUTEX = \begin{array}{l} \text{set of all infinite words } A_0\, A_1\, A_2 \ldots \\ \text{over } 2^{AP} \text{ such that for all } i \in \mathbb{N}: \\ \quad \mathbf{crit}_1 \notin A_i \;\text{ or }\; \mathbf{crit}_2 \notin A_i \end{array}$$

liveness (starvation freedom):

$$LIVE = \begin{array}{l} \text{set of all infinite words } A_0\, A_1\, A_2 \ldots \text{ s.t.} \\ \quad \overset{\infty}{\exists}\, i \in \mathbb{N}.\mathbf{wait}_1 \in A_i \implies \overset{\infty}{\exists}\, i \in \mathbb{N}.\mathbf{crit}_1 \in A_i \\ \wedge\; \overset{\infty}{\exists}\, i \in \mathbb{N}.\mathbf{wait}_2 \in A_i \implies \overset{\infty}{\exists}\, i \in \mathbb{N}.\mathbf{crit}_2 \in A_i \end{array}$$

Satisfaction relation for LT properties

An LT property over AP is a language E of infinite words over the alphabet $\Sigma = 2^{AP}$, i.e., $E \subseteq (2^{AP})^{\omega}$.

Satisfaction relation $\models$ for TS and states:

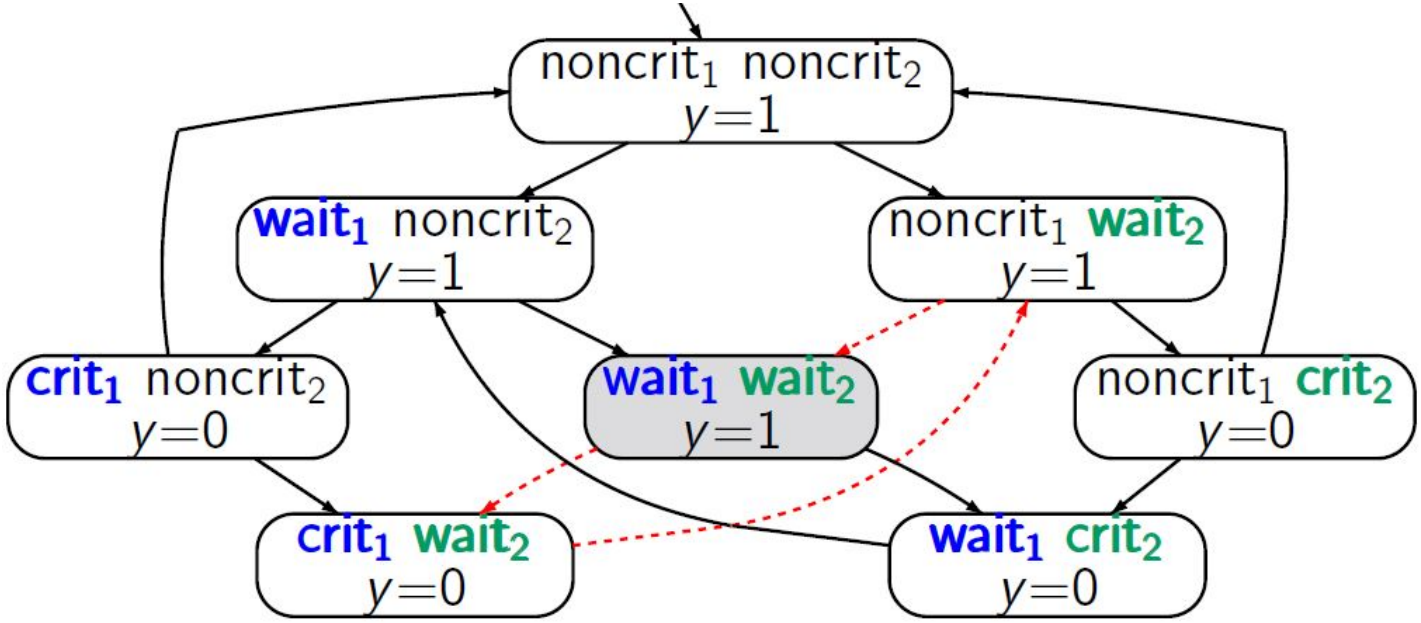
If $\mathcal{T}$ is a TS (without terminal states) over AP and E an LT property over AP then

$$\mathcal{T} \models E \quad \text{iff} \quad \mathit{Traces}(\mathcal{T}) \subseteq E$$

If s is a state in $\mathcal{T}$ then

$$s \models E \quad \text{iff} \quad \mathit{Traces}(s) \subseteq E$$

MUTEX with semaphore



$$\mathcal{T}_{Sem} \models \mathit{MUTEX}, \quad \mathcal{T}_{Sem} \not\models \mathit{LIVE}$$

$$\varnothing \, \{\mathrm{wait}_1\} \, \big(\{\mathrm{wait}_1, \mathrm{wait}_2\} \, \{\mathrm{crit}_1, \mathrm{wait}_2\} \{\mathrm{wait}_2\} \big)^{\omega} \notin \mathit{LIVE}$$

LP properties and trace inclusion

An LT property over AP is a language E of infinite words over the alphabet $\Sigma = 2^{AP}$, i.e., $E \subseteq (2^{AP})^{\omega}$.

If $\mathcal{T}$ is a TS over AP then $\mathcal{T} \models E$ iff $Traces(\mathcal{T}) \subseteq E$.

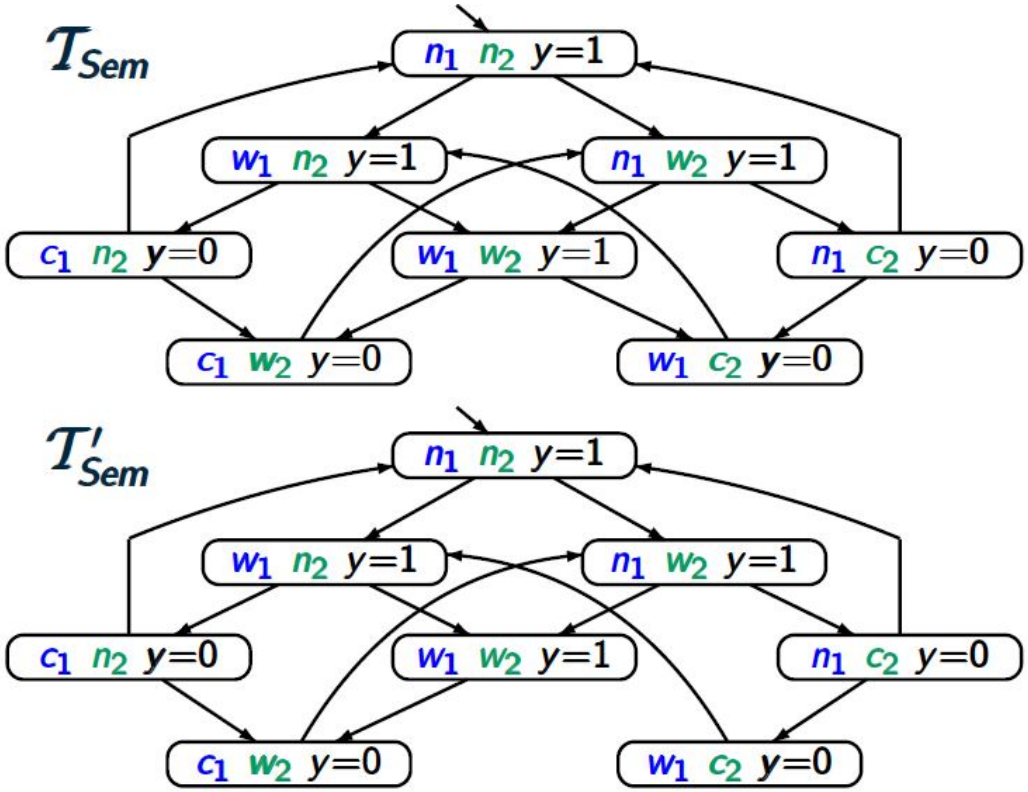
Consequence of these definitions:

If $\mathcal{T}_1$ and $\mathcal{T}_2$ are TS over AP then for all LT properties E over AP:

$$Traces(\mathcal{T}_1) \subseteq Traces(\mathcal{T}_2) \wedge \mathcal{T}_2 \models E \Longrightarrow \mathcal{T}_1 \models E$$

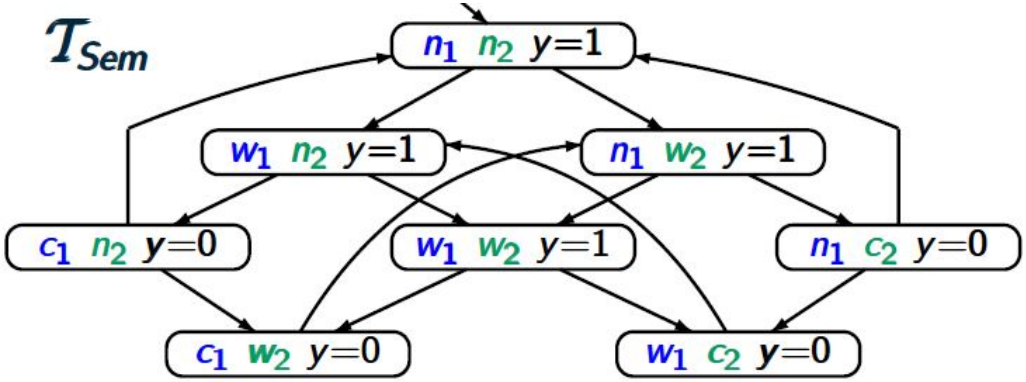
note: $Traces(\mathcal{T}_1) \subseteq Traces(\mathcal{T}_2) \subseteq E$

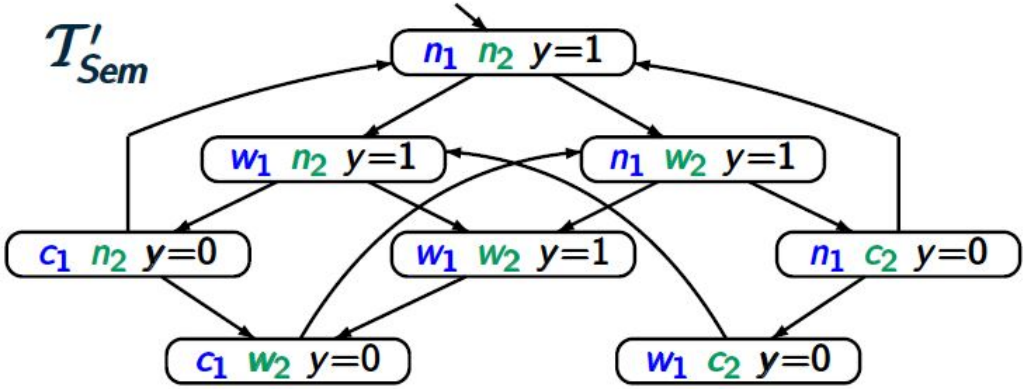
MUTEX with semaphore



$$Paths(\mathcal{T}'_{Sem}) \subseteq Paths(\mathcal{T}_{Sem})$$

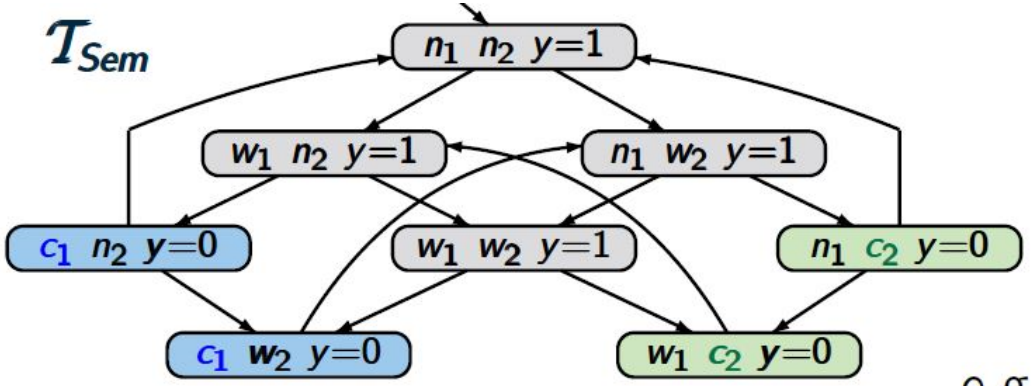
MUTEX with semaphore



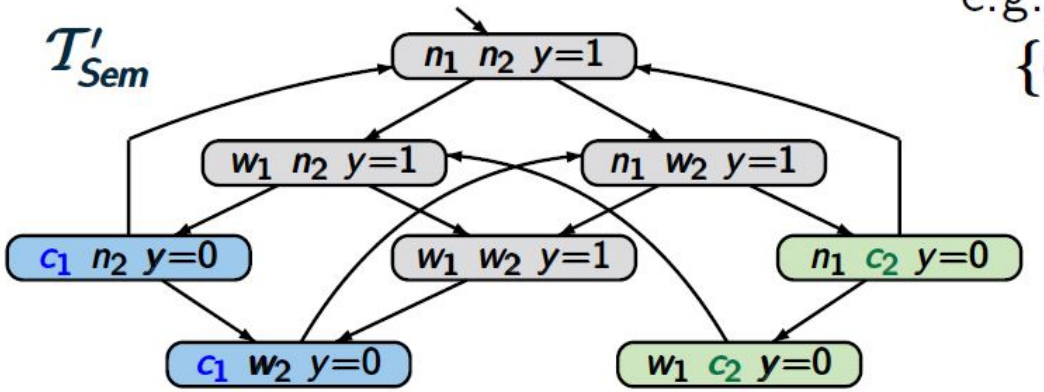


$Traces(\mathcal{T}'_{Sem}) \subseteq Traces(\mathcal{T}_{Sem})$ for any AP

MUTEX with semaphore



e.g., for $AP = \{\mathrm{crit}_1, \mathrm{crit}_2\}$



$Traces(\mathcal{T}_{Sem}) \models E$ implies $Traces(\mathcal{T}'_{Sem}) \models E$ for any E

Trace equivalence

Transition systems $\mathcal{T}_1$ and $\mathcal{T}_2$ over the same set AP of atomic propositions are called trace equivalent iff

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

i.e., trace equivalence requires trace inclusion in both directions

Trace equivalent TS satisfy the **same LT properties**

LP properties and trace relations

Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be TS over AP.

The following statements are equivalent:

(1) $Traces(\mathcal{T}_1) \subseteq Traces(\mathcal{T}_2)$

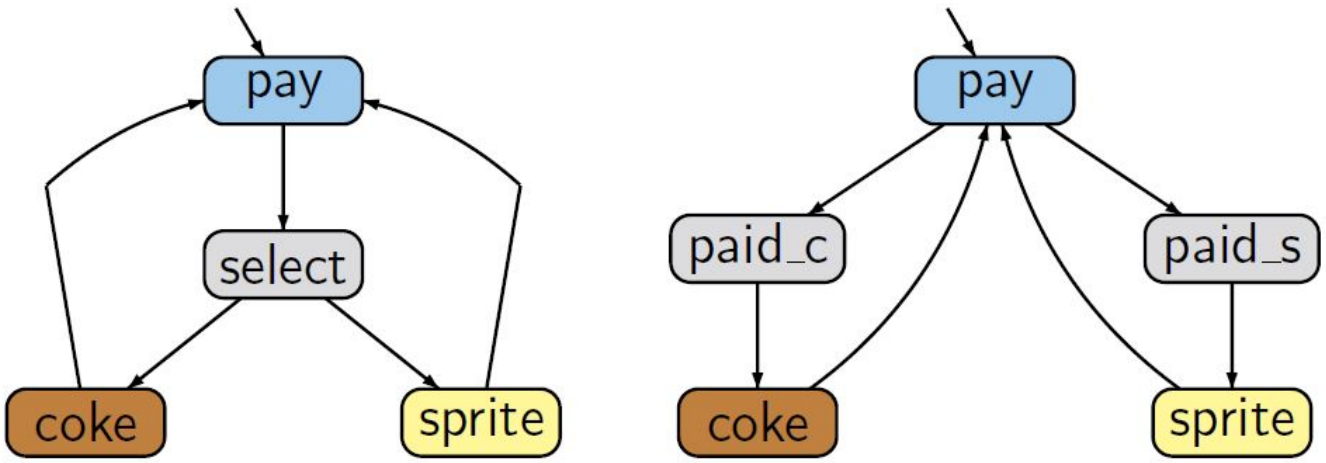
(2) for all LT-properties E: $\mathcal{T}_2 \models E \Longrightarrow \mathcal{T}_1 \models E$

The following statements are equivalent:

(1) $Traces(\mathcal{T}_1) = Traces(\mathcal{T}_2)$

(2) for all LT-properties E: $\mathcal{T}_1 \models E$ iff $\mathcal{T}_2 \models E$

Trace equivalent beverage machines



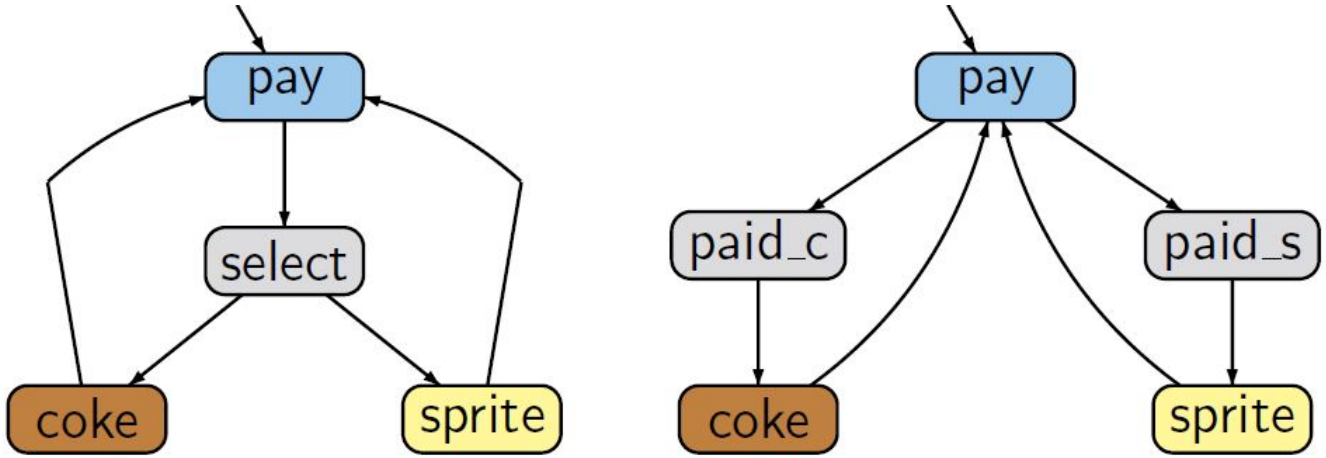
set of atomic propositions $AP = \{pay, coke, sprite\}$

$Traces(\mathcal{T}_1) = Traces(\mathcal{T}_2) =$ set of all infinite words

$\{pay\}\,\varnothing\,\{drink_1\}\,\{pay\}\,\varnothing\,\{drink_2\}\,\ldots$

where $drink_1, drink_2, \ldots \in \{coke, sprite\}$

Trace equivalent beverage machines



set of atomic propositions $AP = \{pay, coke, sprite\}$

$Traces(\mathcal{T}_1) = Traces(\mathcal{T}_2) =$ set of all infinite words

$\{pay\}\,\varnothing\,\{drink_1\}\,\{pay\}\,\varnothing\,\{drink_2\}\ \ldots$

$\mathcal{T}_1$ and $\mathcal{T}_2$ satisfy the same LT-properties over AP

Linear Time Properties

- state-based and linear time view
- definition of linear time properties
- invariants and safety

Classification of LT properties

safety properties *"nothing bad will happen"*

examples:

- mutual exclusion
- deadlock freedom

} special case: **invariants** *"no bad state will be reached"*

- "every red phase is preceded by a yellow phase"

liveness properties *"something good will happen"*

examples:

- "each waiting process will eventually enter its critical section"
- "each philosopher will eat infinitely often"

Propositional logic

$$\Phi \ ::= \ \mathit{true} \ \Big| \ a \ \Big| \ \Phi_1 \wedge \Phi_2 \ \Big| \ \neg\Phi \ \Big| \ \Phi_1 \vee \Phi_2 \ \Big| \ \Phi_1 \rightarrow \Phi_2 \ \Big| \ \ldots$$

$\uparrow$

atomic proposition, i.e., $a \in AP$

semantics: Let $A \subseteq AP$

$A \models \mathit{true}$		
$A \models a$	iff	$a \in A$
$A \models \Phi_1 \wedge \Phi_2$	iff	$A \models \Phi_1$ and $A \models \Phi_2$
$A \models \neg\Phi$	iff	$A \not\models \Phi$

e.g., $\{a, b\} \not\models (a \rightarrow \neg b) \vee c \qquad \{a, b\} \models a \vee c$

Propositional logic

$$\Phi \ ::= \ \mathit{true} \ \Big| \ a \ \Big| \ \Phi_1 \wedge \Phi_2 \ \Big| \ \neg\Phi \ \Big| \ \Phi_1 \vee \Phi_2 \ \Big| \ \Phi_1 \rightarrow \Phi_2 \ \Big| \ \ldots$$

$\uparrow$

atomic proposition, i.e., $a \in AP$

semantics: Let $A \subseteq AP$

$A \models \mathit{true}$		
$A \models a$	iff	$a \in A$
$A \models \Phi_1 \wedge \Phi_2$	iff	$A \models \Phi_1$ and $A \models \Phi_2$
$A \models \neg\Phi$	iff	$A \not\models \Phi$

for state s of a TS over AP: $s \models \Phi$ iff $L(s) \models \Phi$

Invariant

Let E be an LT property over AP.

E is called an invariant if there exists a propositional formula Φ over AP such that

$$E = \{ A_0 A_1 A_2 \ldots \in (2^{AP})^\omega : \forall i \geq 0.\ A_i \models \Phi \}$$

Φ is called the invariant condition of E.

Examples of invariants

mutual exclusion (safety):

$$\mathit{MUTEX} = \begin{array}{c}\text{set of all infinite words } A_0\, A_1\, A_2 \ldots \text{ s.t.} \\ \forall i \in \mathbb{N}.\ \ \mathbf{crit}_1 \notin A_i \ \text{ or } \ \mathbf{crit}_2 \notin A_i\end{array}$$

invariant condition: $\Phi = \neg\mathbf{crit}_1 \vee \neg\mathbf{crit}_2$

here: $AP = \{\mathbf{crit}_1, \mathbf{crit}_2, \ldots\}$

Examples for invariants

mutual exclusion (safety):

$$MUTEX = \begin{array}{l} \text{set of all infinite words } A_0\, A_1\, A_2 \ldots \text{ s.t.} \\ \forall i \in \mathbb{N}.\ \ \mathbf{crit_1} \notin A_i \ \text{ or } \ \mathbf{crit_2} \notin A_i \end{array}$$

invariant condition: $\Phi = \neg\mathbf{crit_1} \vee \neg\mathbf{crit_2}$

deadlock freedom for 5 dining philosophers:

$$DF = \begin{array}{l} \text{set of all infinite words } A_0\, A_1\, A_2 \ldots \text{ s.t.} \\ \forall i \in \mathbb{N}\ \exists \mathbf{j} \in \{0,1,2,3,4\}.\ \mathbf{wait}_j \notin A_i \end{array}$$

invariant condition:

$$\Phi = \neg\mathbf{wait_0} \vee \neg\mathbf{wait_1} \vee \neg\mathbf{wait_2} \vee \neg\mathbf{wait_3} \vee \neg\mathbf{wait_4}$$

here: $AP = \{\mathbf{wait}_j : 0 \leq j \leq 4\} \cup \{\ldots\}$

Satisfaction of invariants

Let E be an LT property over AP. E is called an invariant if there exists a propositional formula Φ s.t.

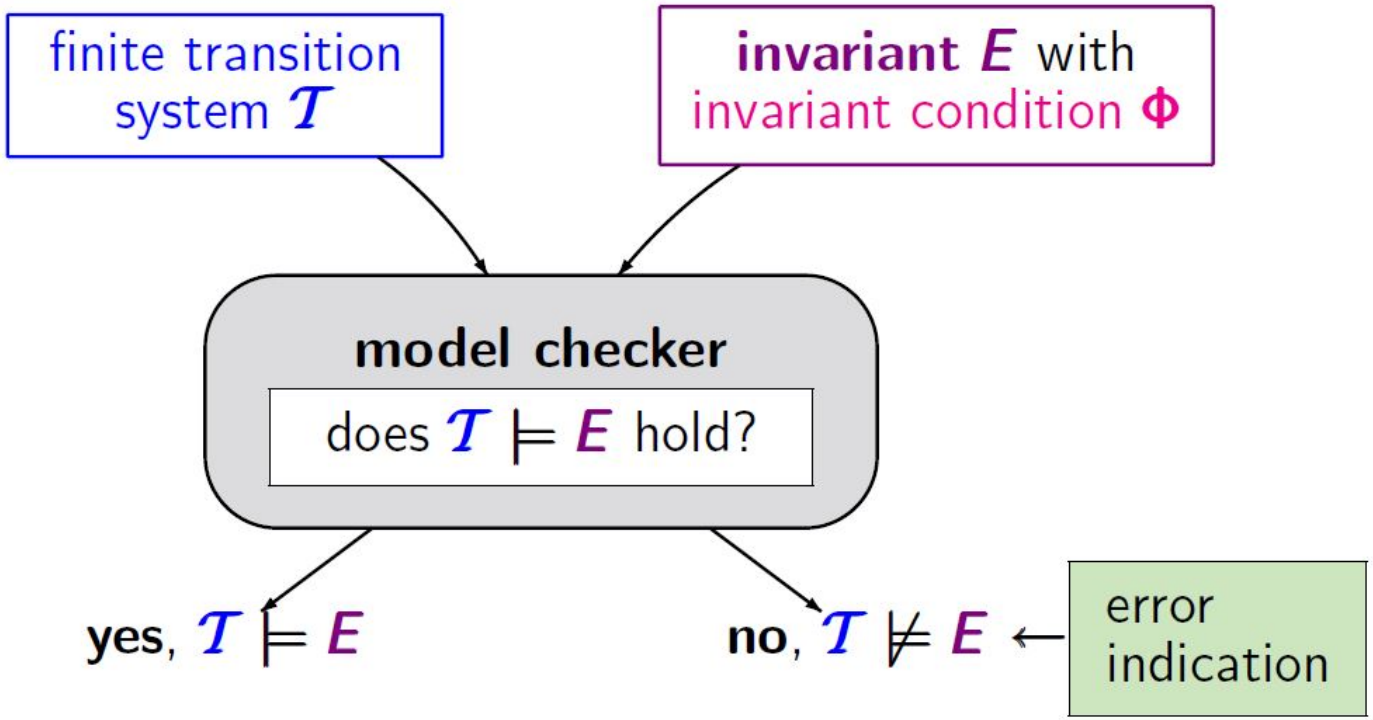
$$E = \{ A_0 A_1 A_2 \ldots \in (2^{AP})^\omega : \forall i > 0. A_i \models \Phi \}$$

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

$$\begin{aligned} \mathcal{T} \models E \quad & \text{iff} \quad trace(\pi) \in E \text{ for all } \pi \in Paths(\mathcal{T}) \\ & \text{iff} \quad s \models \Phi \text{ for all states } s \text{ on a path of } \mathcal{T} \\ & \text{iff} \quad s \models \Phi \text{ for all states } s \in Reach(\mathcal{T}) \end{aligned}$$

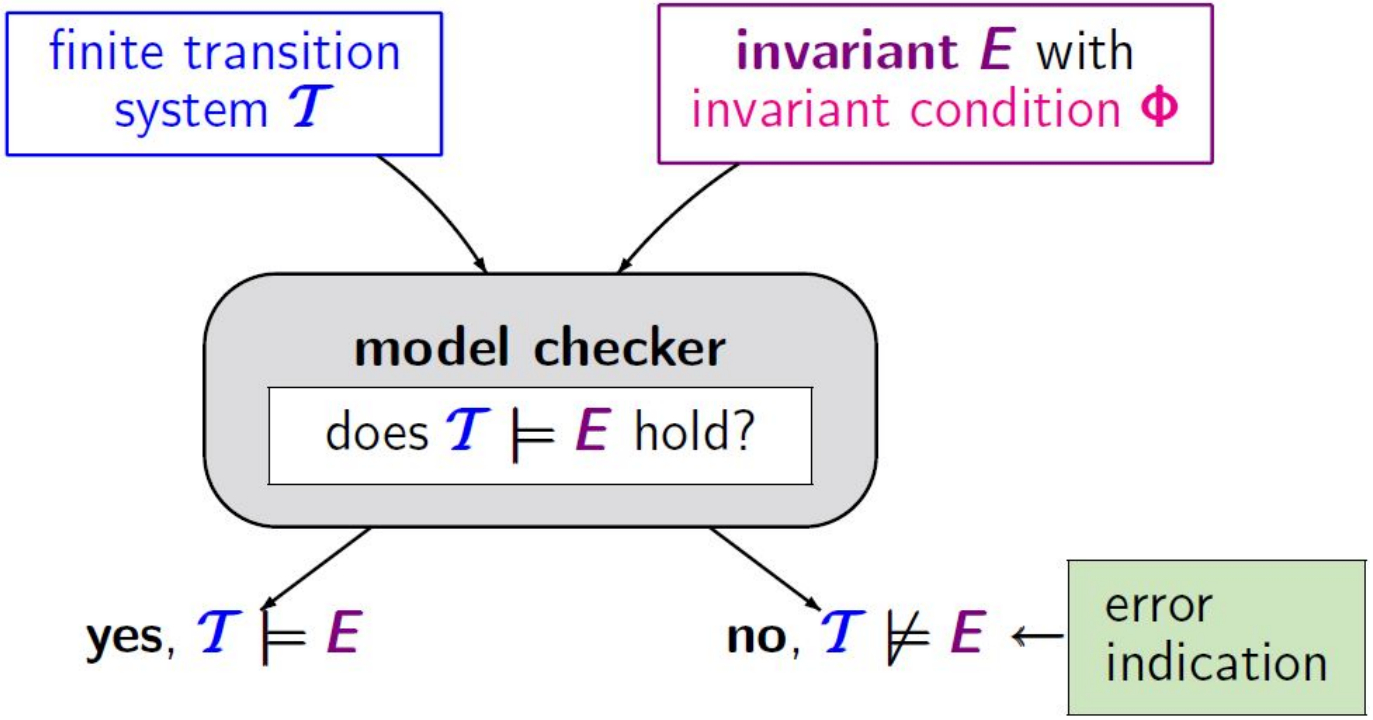
i.e., Φ holds in all initial states and is invariant under all transitions

Invariant checking



perform a graph analysis (**DFS** or **BFS**) to check whether $s \models \Phi$ for all $s \in \mathit{Reach}(\mathcal{T})$

Invariant checking



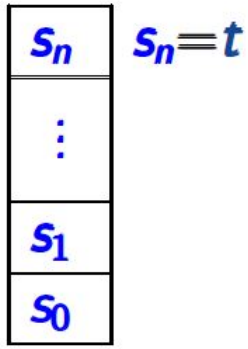
error indication: initial path fragment $s_0\, s_1 \ldots s_{n-1} s_n$
such that $s_i \models \Phi$ for $0 \leq i < n$ and $s_n \not\models \Phi$

DFS-based invariant checking

input: finite transition system $\mathcal{T}$, invariant condition Φ

$\pi := \varnothing \longleftarrow$ stack for error indication

```
FOR ALL s0 ∈ S0 DO
      IF DFS(s0, Φ) THEN
         return "no" and reverse(π)
      FI
OD
return "yes"
```



$DFS(s_0, \Phi)$ returns "true" iff depth-first search from state s_0 leads to some state t with $t \not\models \Phi$

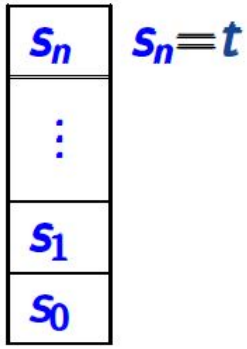
DFS-based invariant checking

input: finite transition system $\mathcal{T}$, invariant condition Φ

$U := \emptyset \longleftarrow$ stores the "processed" states

$\pi := \emptyset \longleftarrow$ stack for error indication

```
FOR ALL s0 ∈ S0 DO
      IF DFS(s0, Φ) THEN
          return "no" and reverse(π)
      FI
OD
return "yes"
```



$DFS(s_0, \Phi)$ returns "true" iff depth-first search from state s_0 leads to some state t with $t \not\models \Phi$

Recursive algorithm DFS (s, φ)

"searches" for a path fragment $s \ldots t$ with $t \not\models \Phi$

```
IF s ∉ U THEN
    IF s ⊭ Φ THEN return "true" FI
    IF s ⊨ Φ THEN
        insert s in U;

    FI
FI
return "false"
```

Recursive algorithm DFS (s, φ)

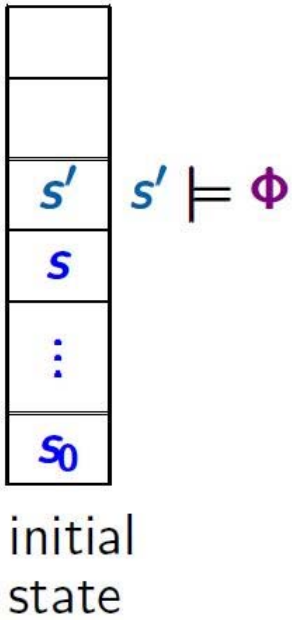
"searches" for a path fragment $s \ldots t$ with $t \not\models \Phi$

```
IF s ∉ U THEN
    IF s ⊭ Φ THEN return "true" FI
    IF s ⊨ Φ THEN
        insert s in U;
        FOR ALL s' ∈ Post(s) DO
            IF DFS(s', Φ) THEN
                return "true" FI
        OD
    FI
FI
return "false"
```

Recursive algorithm DFS (s, φ)

"searches" for a path fragment $s \ldots t$ with $t \not\models \Phi$

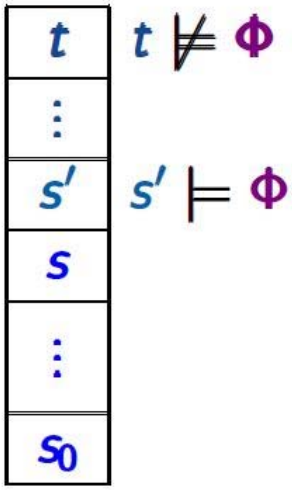
```
Push(π, s);
IF s ∉ U THEN
    IF s ⊭ Φ THEN return "true" FI
    IF s ⊨ Φ THEN
        insert s in U;
        FOR ALL s' ∈ Post(s) DO
            IF DFS(s', Φ) THEN
                return "true" FI
        OD
    FI
FI
Pop(π); return "false"
```



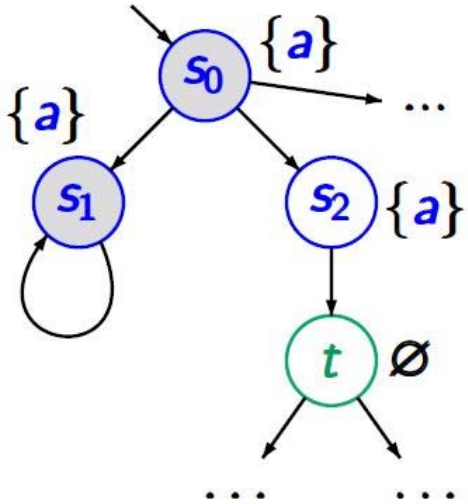
Recursive algorithm DFS (s, φ)

"searches" for a path fragment $s \ldots s' \ldots t$ with $t \not\models \Phi$

```
Push(π, s);
IF s ∉ U THEN
    IF s ⊭ Φ THEN return "true" FI
    IF s ⊨ Φ THEN
        insert s in U;
        FOR ALL s' ∈ Post(s) DO
            IF DFS(s', Φ) THEN
                return "true" FI
        OD
    FI
FI
Pop(π); return "false"
```



Example: invariant checking



invariant condition a

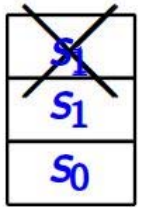
$$s_0, s_1, s_2 \models a$$
$$t \not\models a$$

$DFS(s_0, a)$

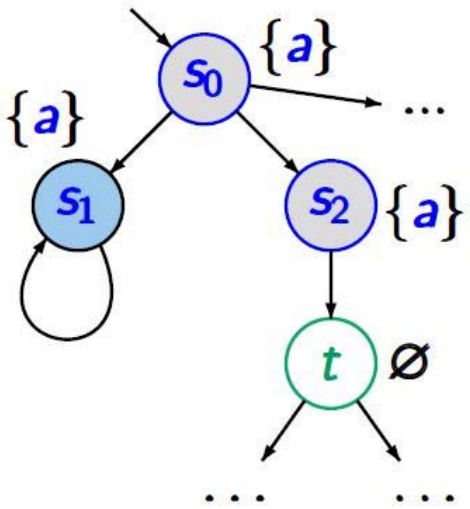
$DFS(s_1, a)$

$\boxed{DFS(s_1, a)}$

stack π

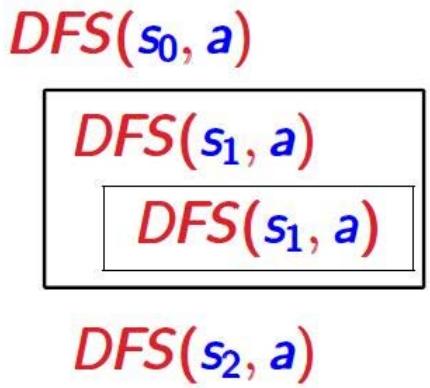


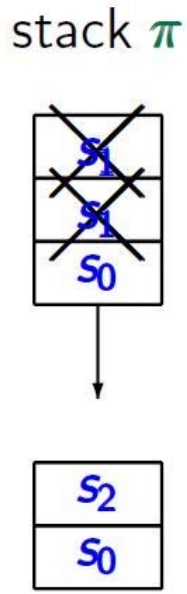
Example: invariant checking



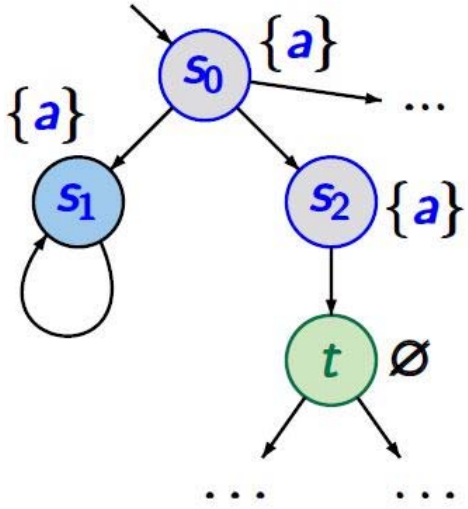
invariant condition a

$$s_0, s_1, s_2 \models a$$
$$t \not\models a$$





Example: invariant checking



invariant condition a

$$s_0, s_1, s_2 \models a$$
$$t \not\models a$$

$DFS(s_0, a)$

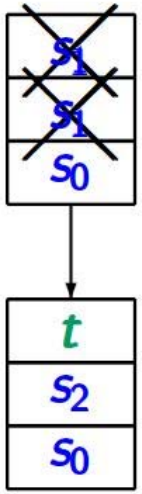
$DFS(s_1, a)$

$DFS(s_1, a)$

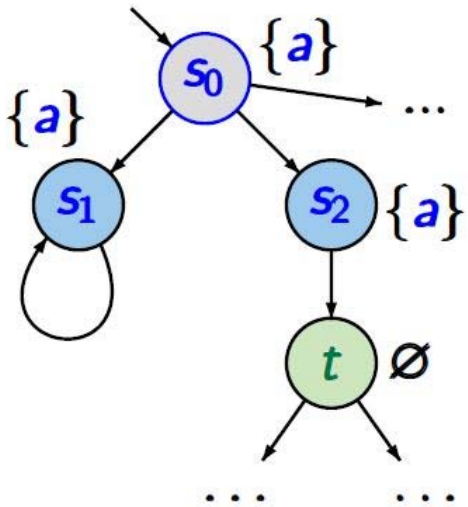
$DFS(s_2, a)$

$DFS(t, a)$

stack π



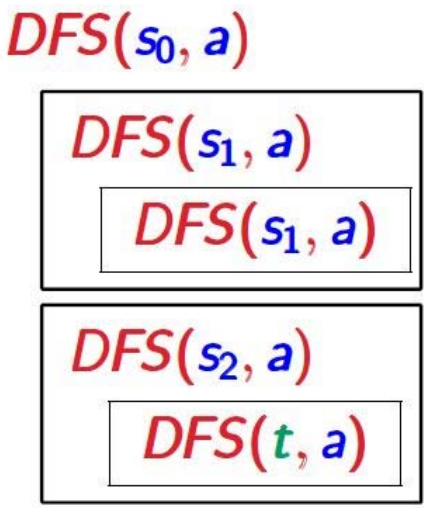
Example: invariant checking

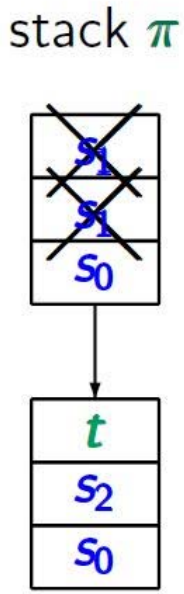


invariant condition a

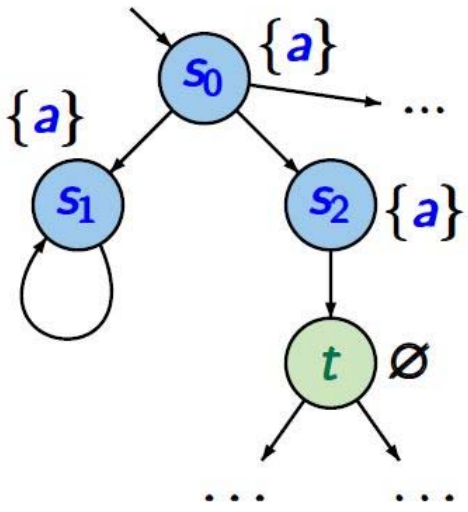
$s_0, s_1, s_2 \models a$

$t \not\models a$



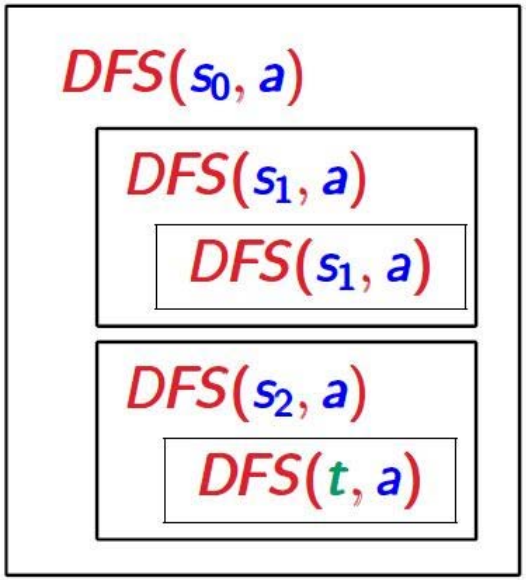


Example: invariant checking

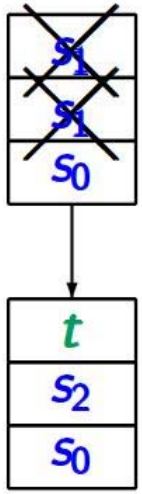


invariant condition a

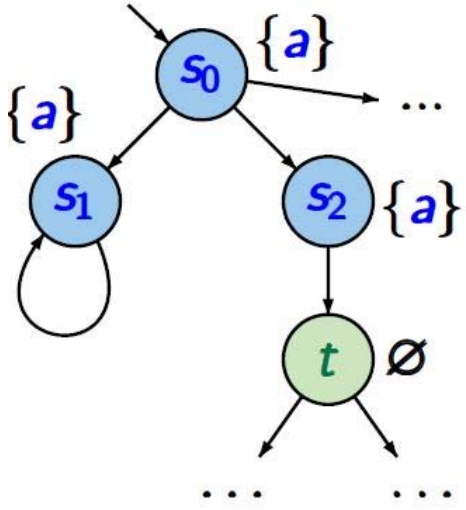
$s_0, s_1, s_2 \models a$
$t \not\models a$



stack π



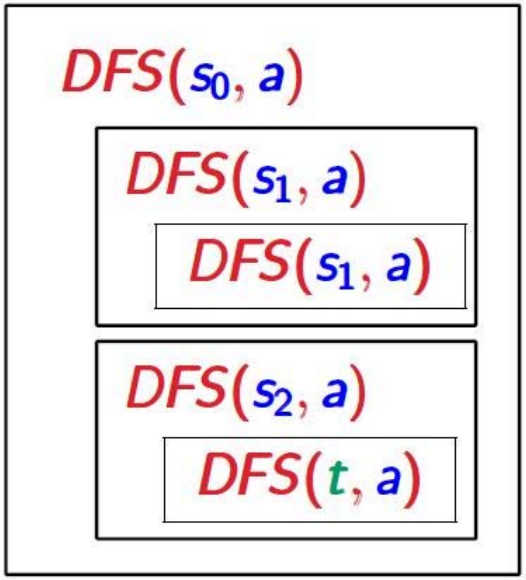
Example: invariant checking



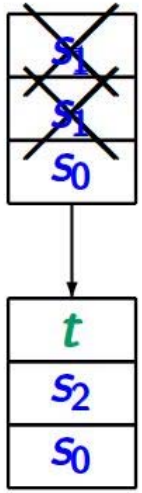
invariant condition a

$s_0, s_1, s_2 \models a$

$t \not\models a$



stack π



$s_0 \not\models$ "always a" $\leftarrow$ error indication: $s_0\ s_2\ t$

Safety properties

state that "nothing bad will happen"

invariants: ⟵ "no **bad state** will be reached"

- mutual exclusion: *never* $crit_1 \wedge crit_2$
- deadlock freedom: *never* $\bigwedge_{0 \leq i < n} wait_i$

---

other safety properties: ⟵ "no **bad prefix**"

- German traffic lights:
  every red phase is preceded by a yellow phase
- beverage machine:
  the total number of entered coins is never less than the total number of released drinks

Trace relations and properties

trace inclusion

$\mathit{Traces}(\mathcal{T}) \subseteq \mathit{Traces}(\mathcal{T}')$ iff

for all LT properties E: $\mathcal{T}' \models E \Longrightarrow \mathcal{T} \models E$

finite trace inclusion

$\mathit{Traces}_{fin}(\mathcal{T}) \subseteq \mathit{Traces}_{fin}(\mathcal{T}')$ iff

for all safety properties E: $\mathcal{T}' \models E \Longrightarrow \mathcal{T} \models E$

Trace relations and properties

trace equivalence

$\mathit{Traces}(\mathcal{T}) = \mathit{Traces}(\mathcal{T}')$ iff

$\mathcal{T}$ and $\mathcal{T}'$ satisfy the same LT properties

finite trace equivalence

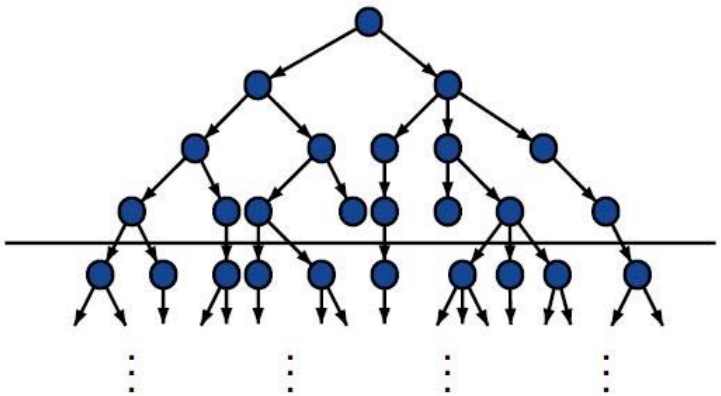
$\mathit{Traces}_{fin}(\mathcal{T}) = \mathit{Traces}_{fin}(\mathcal{T}')$ iff

$\mathcal{T}$ and $\mathcal{T}'$ satisfy the same safety properties

Tracesfin vs traces

finite TS $\mathcal{T}'$

paths from state t_0
(unfolded into a tree)



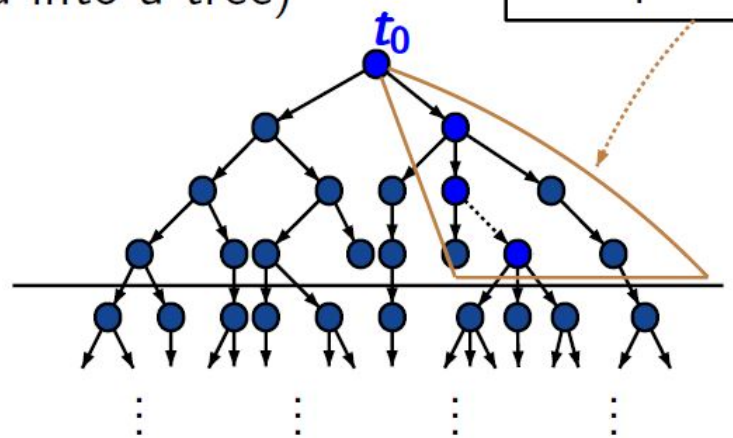
finite until
depth $\leq n$

Tracesfin vs traces

finite TS $\mathcal{T}'$

paths from state t_0
(unfolded into a tree)

contains all path fragments
with trace $A_0\, A_1 \ldots A_n$
in particular: $t_0\, t_1 \ldots t_n$



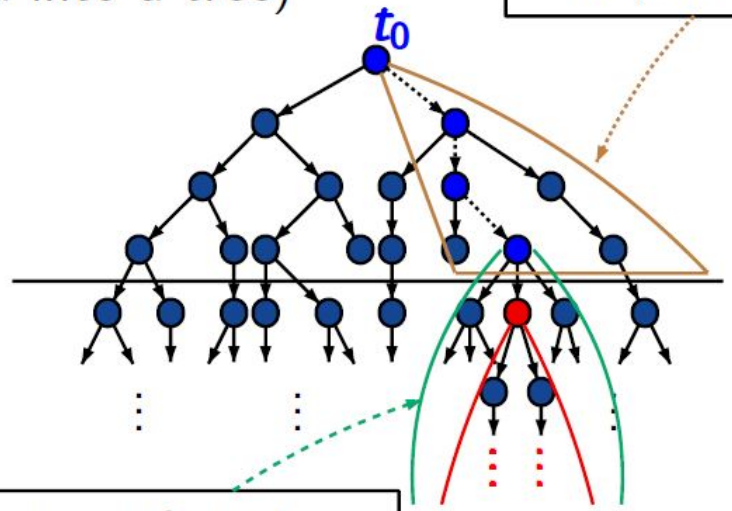
finite until
depth $\leq n$

Tracesfin vs traces

finite TS $\mathcal{T}'$

paths from state t_0
(unfolded into a tree)

contains all path fragments
with trace $A_0\, A_1 \ldots A_n$
in particular: $t_0\, t_1 \ldots t_n$



finite until
depth $\leq n$

contains infinitely
many path fragments
$t_n\, s^m_{n+1} \ldots s^m_m$

there exists $t_{n+1} \in Post(t_n)$
s.t. $t_{n+1} = s^m_{n+1}$ for
infinitely many m

"liveness: something good will happen."

"event ***a*** will occur eventually"

e.g., termination for sequential programs

"event ***a*** will occur infinitely many times"

e.g., starvation freedom for dining philosophers

"whenever event ***b*** occurs then event ***a*** will occur sometimes in the future"

e.g., every waiting process enters eventually its critical section

Which property type?

- Each philosopher thinks infinitely often.
- Two philosophers next to each other never eat at the same time.

Which property type?

- Each philosopher thinks infinitely often.

liveness

- Two philosophers next to each other never eat at the same time.

- Whenever a philosopher eats then he has been thinking at some time before.

Which property type?

- Each philosopher thinks infinitely often.

liveness

- Two philosophers next to each other never eat at the same time.

invariant

- Whenever a philosopher eats then he has been thinking at some time before.

Which property type?

- Each philosopher thinks infinitely often.

  liveness

- Two philosophers next to each other never eat at the same time.

  invariant

- Whenever a philosopher eats then he has been thinking at some time before.

  safety

- Whenever a philosopher eats then he will think some time afterwards.

Which property type?

- Each philosopher thinks infinitely often.

  liveness

- Two philosophers next to each other never eat at the same time.

  invariant

- Whenever a philosopher eats then he has been thinking at some time before.

  safety

- Whenever a philosopher eats then he will think some time afterwards.

  liveness

- Between two eating phases of philosopher i lies at least one eating phase of philosopher $i+1$.

Which property type?

- Each philosopher thinks infinitely often.

  liveness

- Two philosophers next to each other never eat at the same time.

  invariant

- Whenever a philosopher eats then he has been thinking at some time before.

  safety

- Whenever a philosopher eats then he will think some time afterwards.

  liveness

- Between two eating phases of philosopher i lies at least one eating phase of philosopher $i+1$.

  safety