Overview

Introduction

Modelling parallel systems

Linear Time Properties

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

Regular Properties

Linear Temporal Logic

Computation-Tree Logic

Equivalences and Abstraction

Classification of LT-properties

safety properties *"nothing bad will happen"*

liveness properties *"something good will happen"*

Classification of LT-properties

safety properties *"nothing bad will happen"*

examples:

- mutual exclusion
- deadlock freedom
- "every red phase is preceded by a yellow phase"

liveness properties *"something good will happen"*

Classification of LT-properties

safety properties *"nothing bad will happen"*

examples:

- mutual exclusion
- deadlock freedom
- "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"

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

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

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

Propositional logic

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

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

semantics: interpretation over a subsets of AP

Propositional logic

$$\Phi \ ::= \ \textit{true} \mid a \mid \Phi_1 \wedge \Phi_2 \mid \neg\Phi \mid \Phi_1 \vee \Phi_2 \mid \Phi_1 \rightarrow \Phi_2 \mid \ldots$$

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

semantics: Let $A \subseteq AP$

$$\begin{array}{lll} A \models \textit{true} & & \\ A \models a & \text{iff} & a \in A \\ A \models \Phi_1 \wedge \Phi_2 & \text{iff} & A \models \Phi_1 \text{ and } A \models \Phi_2 \\ A \models \neg\Phi & \text{iff} & A \not\models \Phi \end{array}$$

Propositional logic

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

↑

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

$$\Phi \;::=\; \textit{true} \mid a \mid \Phi_1 \wedge \Phi_2 \mid \neg\Phi \mid \Phi_1 \vee \Phi_2 \mid \Phi_1 \rightarrow \Phi_2 \mid \ldots$$

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

semantics: Let $A \subseteq AP$

$$\begin{array}{llll} A \models \textit{true} & & & \\ A \models a & \text{iff} & a \in A & \\ A \models \Phi_1 \wedge \Phi_2 & \text{iff} & A \models \Phi_1 \text{ and } A \models \Phi_2 & \\ A \models \neg\Phi & \text{iff} & A \not\models \Phi & \end{array}$$

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

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

Let E be an LT property over AP.

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

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

Φ is called the invariant condition of E.

Examples for invariants

mutual exclusion (safety):

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

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

Examples for invariants

mutual exclusion (safety):

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

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

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

Examples for invariants

mutual exclusion (safety):

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

invariant condition: $\Phi = \neg crit_1 \vee \neg crit_2$

deadlock freedom for 5 dining philosophers:

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

invariant condition:

$$\Phi = \neg wait_0 \vee \neg wait_1 \vee \neg wait_2 \vee \neg wait_3 \vee \neg wait_4$$

here: $AP = \{wait_j : 0 \leq j \leq 4\} \cup \{\ldots\}$

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.

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

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.

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

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

$\mathcal{T} \models E$ iff $trace(\pi) \in E$ for all $\pi \in Paths(\mathcal{T})$

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.

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

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

$\mathcal{T} \models E$ iff $trace(\pi) \in E$ for all $\pi \in Paths(\mathcal{T})$

iff $s \models \Phi$ for all states s on a path of $\mathcal{T}$

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.

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

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

$\mathcal{T} \models E$ iff $trace(\pi) \in E$ for all $\pi \in Paths(\mathcal{T})$

iff $s \models \Phi$ for all states s on a path of $\mathcal{T}$

iff $s \models \Phi$ for all states $s \in Reach(\mathcal{T})$

↑ set of reachable states in $\mathcal{T}$

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.

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

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

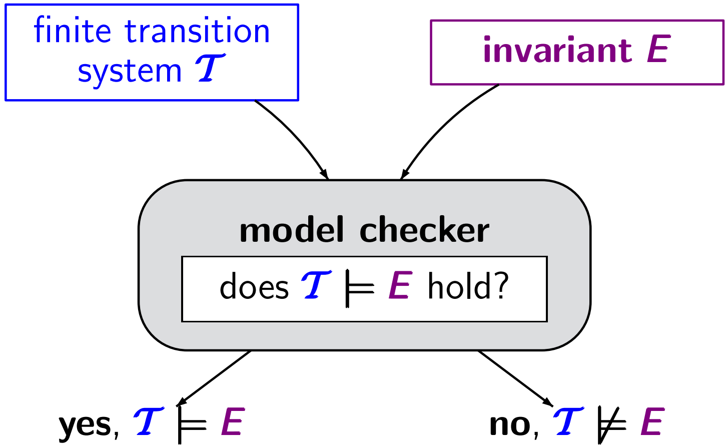
$\mathcal{T} \models E$ iff $trace(\pi) \in E$ for all $\pi \in Paths(\mathcal{T})$

iff $s \models \Phi$ for all states s on a path of $\mathcal{T}$

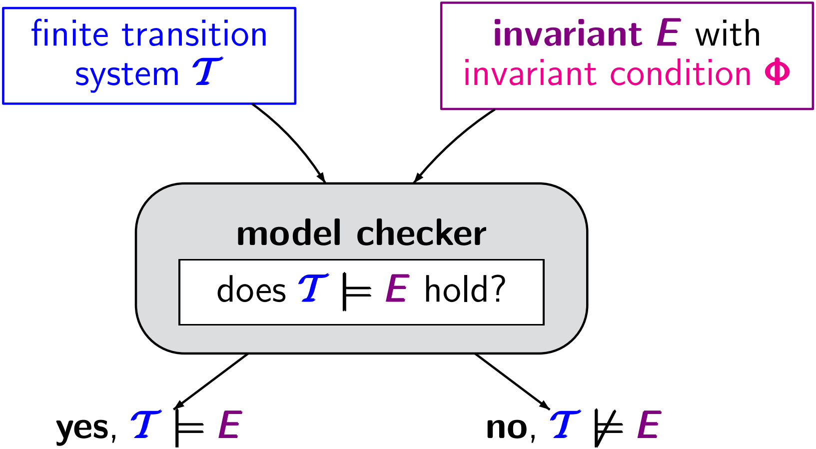
iff $s \models \Phi$ for all states $s \in Reach(\mathcal{T})$

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

Invariant checking

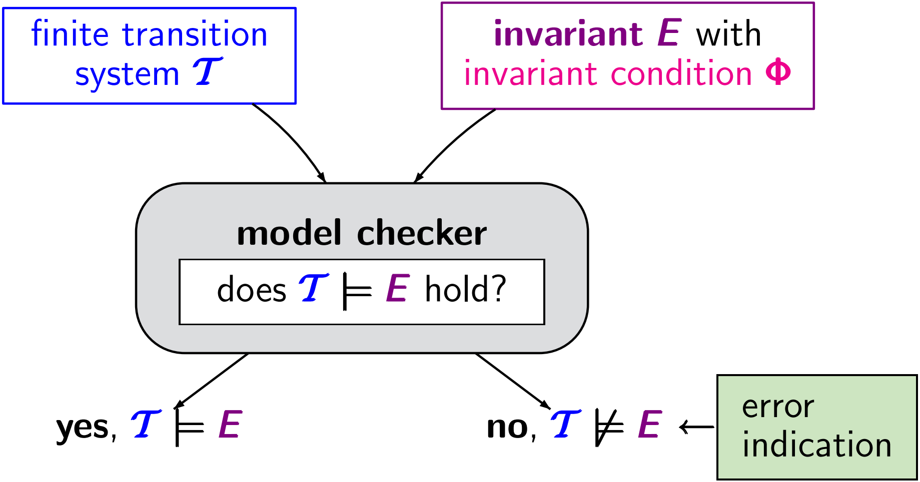


Invariant checking



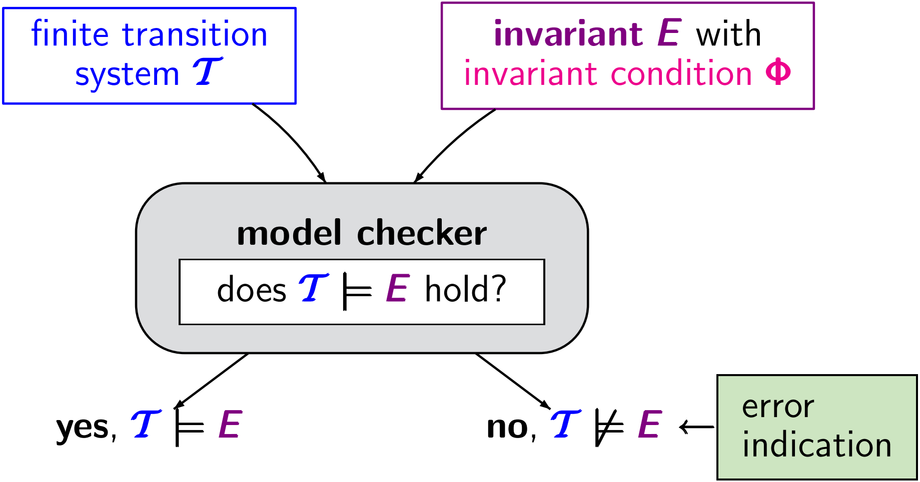
perform a graph analysis (**DFS** or **BFS**) to check whether $s \models \Phi$ for all $s \in Reach(\mathcal{T})$

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

DFS-based invariant checking

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

```
FOR ALL s_0 ∈ S_0 DO
      IF DFS(s_0, Φ) THEN
          return "no"
      FI
OD
return "yes"
```

DFS-based invariant checking

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

```
FOR ALL s_0 ∈ S_0 DO
      IF DFS(s_0, Φ) THEN
           return "no"
      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

```
π := ∅  ⟵  stack for error indication
FOR ALL s0 ∈ S0 DO
      IF DFS(s0, Φ) THEN
            return "no" and reverse(π)
      FI
OD
return "yes"
```

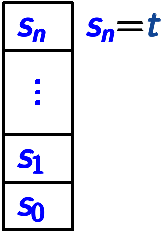
$DFS(s_0, \Phi)$ returns "true" iff depth-first search from state s_0 leads to some state t with $t \not\models \Phi$

DFS-based invariant checking

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

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

FOR ALL $s_0 \in S_0$ DO
    IF $DFS(s_0, \Phi)$ THEN
        return "no" and $reverse(\pi)$
    FI
OD
return "yes"



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

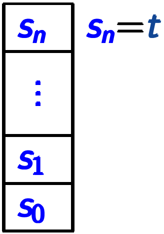
DFS-based invariant checking

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

$U := \varnothing$ ⟵ stores the "processed" states

$\pi := \varnothing$ ⟵ 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 $DFS(s, \Phi)$

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

Recursive algorithm $DFS(s, \Phi)$

"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

               ⋮

     FI
FI
return "false"
```

Recursive algorithm $DFS(s, \Phi)$

"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 $DFS(s, \Phi)$

"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 $DFS(s, \Phi)$

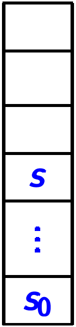
"searches" for a path fragment $s \dots t$ with $t \not\models \Phi$

```
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 $DFS(s, \Phi)$

"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"
```

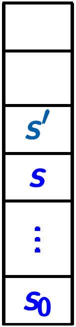


initial state

Recursive algorithm $DFS(s, \Phi)$

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

$Push(\pi, s)$;
IF $s \notin U$ THEN
    IF $s \not\models \Phi$ THEN return "true" FI
    IF $s \models \Phi$ THEN
        insert s in U;
        FOR ALL $s' \in Post(s)$ DO
            IF $DFS(s', \Phi)$ THEN
                return "true" FI
        OD
    FI
FI
$Pop(\pi)$; return "false"

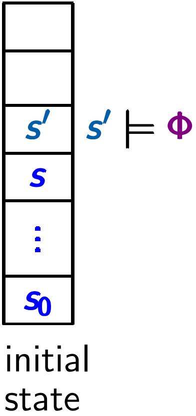


initial state

Recursive algorithm $DFS(s, \Phi)$

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

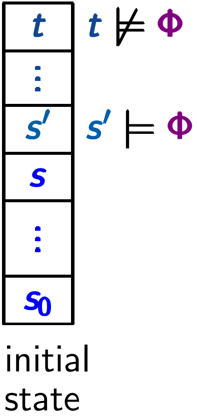
$Push(\pi, s)$;
IF $s \notin U$ THEN
    IF $s \not\models \Phi$ THEN return "true" FI
    IF $s \models \Phi$ THEN
        insert s in U;
        FOR ALL $s' \in Post(s)$ DO
            IF $DFS(s', \Phi)$ THEN
                return "true" FI
        OD
    FI
FI
$Pop(\pi)$; return "false"



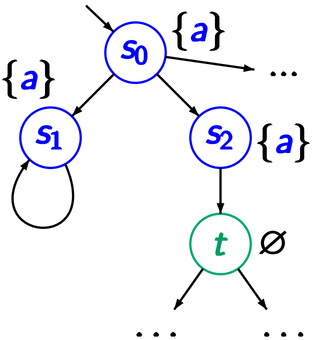
Recursive algorithm $DFS(s, \Phi)$

"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

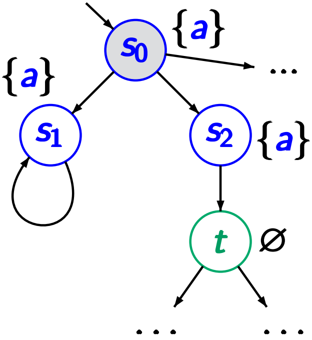


invariant
condition a

$s_0, s_1, s_2 \models a$

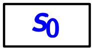
$t \not\models a$

Example: invariant checking



$DFS(s_0, a)$

stack π

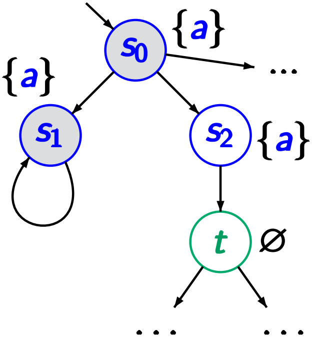


invariant
condition a

$s_0, s_1, s_2 \models a$

$t \not\models a$

Example: invariant checking



$DFS(s_0, a)$

$DFS(s_1, a)$

stack π

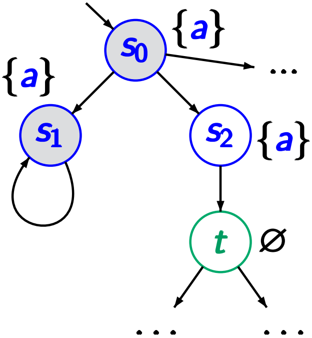
s_1
s_0

invariant condition a

$s_0, s_1, s_2 \models a$

$t \not\models a$

Example: invariant checking



invariant
condition a

$s_0, s_1, s_2 \models a$

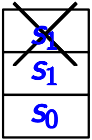
$t \not\models a$

$DFS(s_0, a)$

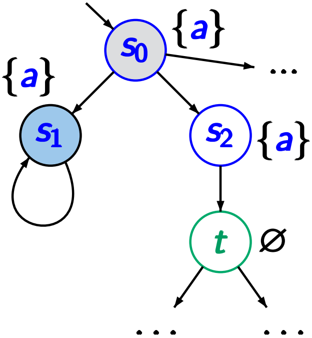
$DFS(s_1, a)$

$DFS(s_1, a)$

stack π



Example: invariant checking

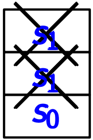


$DFS(s_0, a)$

$DFS(s_1, a)$

$DFS(s_1, a)$

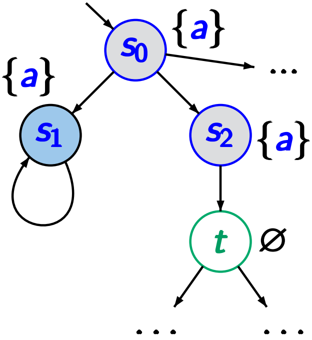
stack π



invariant condition a

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

Example: invariant checking

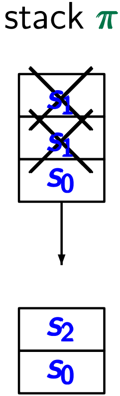


$DFS(s_0, a)$

$DFS(s_1, a)$

$DFS(s_1, a)$

$DFS(s_2, a)$

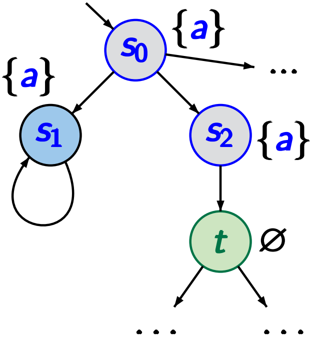


invariant condition a

$s_0, s_1, s_2 \models a$

$t \not\models a$

Example: invariant checking



invariant condition a

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

$DFS(s_0, a)$

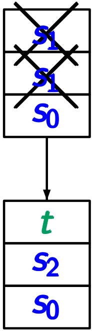
$DFS(s_1, a)$

$DFS(s_1, a)$

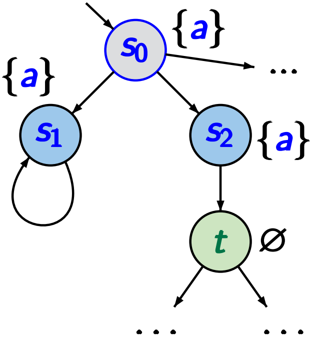
$DFS(s_2, a)$

$DFS(t, a)$

stack π



Example: invariant checking



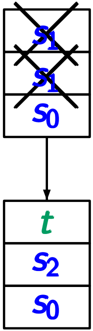
invariant condition a

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

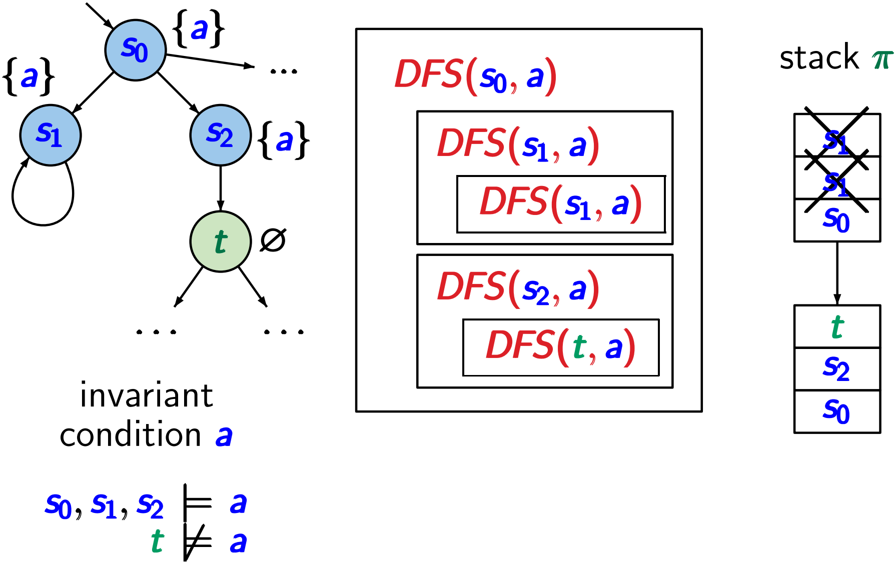
$DFS(s_0, a)$

- $DFS(s_1, a)$
  - $DFS(s_1, a)$
- $DFS(s_2, a)$
  - $DFS(t, a)$

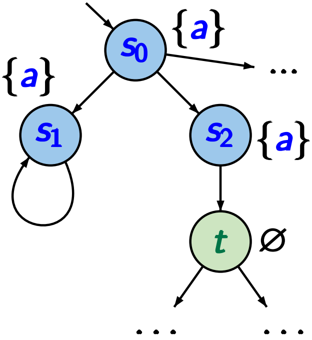
stack π



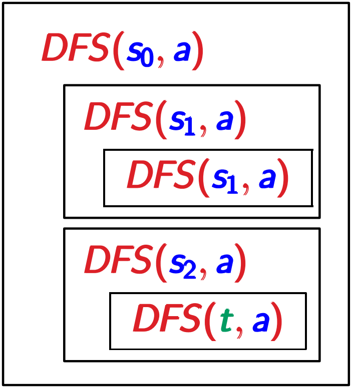
Example: invariant checking



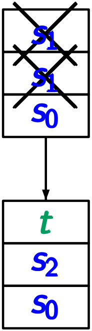
Example: invariant checking



invariant condition a



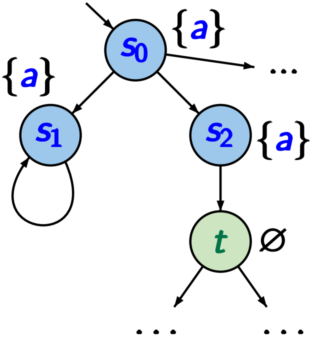
stack π



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

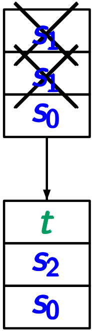
$s_0 \not\models$ "always a"

Example: invariant checking



$DFS(s_0, a)$
- $DFS(s_1, a)$
  - $DFS(s_1, a)$
- $DFS(s_2, a)$
  - $DFS(t, a)$

stack π



invariant condition a

$s_0, s_1, s_2 \models a$

$t \not\models a$

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

Introduction

Modelling parallel systems

Linear Time Properties

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

Regular Properties

Linear Temporal Logic

Computation-Tree Logic

Equivalences and Abstraction

Safety properties

state that “nothing bad will happen”

Safety properties

state that "nothing bad will happen"

invariants:

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

---

other safety properties:

- 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

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:

- 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

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* $\bigwedge_{0 \le i < n} wait_i$

---

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

Bad prefixes of safety properties

- traffic lights:

every red phase is preceded by a yellow phase

↑

bad prefix: finite trace fragment where a red phase appears without being preceded by a yellow phase

e.g., … {●} {●}

Bad prefixes of safety properties

- traffic lights:

  every red phase is preceded by a yellow phase

  ↑

  bad prefix: finite trace fragment where a red phase appears without being preceded by a yellow phase
  e.g., ... {●} {●}

- beverage machine:

  the total number of entered coins is never less than the total number of released drinks

  ↑

  bad prefix, e.g., {*pay*} {*drink*} {*drink*}

Definition of safety properties

Let E be a LT property over AP, i.e., $E \subseteq (2^{AP})^\omega$.

Definition of safety properties

Let E be a LT property over AP, i.e., $E \subseteq (2^{AP})^\omega$.

E is called a safety property if for all words

$$\sigma = A_0 A_1 A_2 \ldots \in (2^{AP})^\omega \setminus E$$

there exists a finite prefix $A_0 A_1 \ldots A_n$ of σ such that *none* of the words $A_0 A_1 \ldots A_n B_{n+1} B_{n+2} B_{n+3} \ldots$ belongs to E, i.e.,

$$E \cap \{\sigma' \in (2^{AP})^\omega : A_0 \ldots A_n \text{ is a prefix of } \sigma'\} = \varnothing$$

Definition of safety properties

Let E be a LT property over AP, i.e., $E \subseteq (2^{AP})^\omega$.

E is called a safety property if for all words

$$\sigma = A_0 A_1 A_2 \ldots \in (2^{AP})^\omega \setminus E$$

there exists a finite prefix $A_0 A_1 \ldots A_n$ of σ such that *none* of the words $A_0 A_1 \ldots A_n B_{n+1} B_{n+2} B_{n+3} \ldots$ belongs to E, i.e.,

$$E \cap \{\sigma' \in (2^{AP})^\omega : A_0 \ldots A_n \text{ is a prefix of } \sigma'\} = \varnothing$$

Such words $A_0 A_1 \ldots A_n$ are called bad prefixes for E.

Definition of safety properties

Let E be a LT property over AP, i.e., $E \subseteq (2^{AP})^\omega$.

E is called a safety property if for all words

$$\sigma = A_0 A_1 A_2 \ldots \in (2^{AP})^\omega \setminus E$$

there exists a finite prefix $A_0 A_1 \ldots A_n$ of σ such that *none* of the words $A_0 A_1 \ldots A_n B_{n+1} B_{n+2} B_{n+3} \ldots$ belongs to E, i.e.,

$$E \cap \{\sigma' \in (2^{AP})^\omega : A_0 \ldots A_n \text{ is a prefix of } \sigma'\} = \varnothing$$

Such words $A_0 A_1 \ldots A_n$ are called bad prefixes for E.

$E =$ set of all infinite words that do *not* have a bad prefix

Definition of safety properties

Let E be a LT property over AP, i.e., $E \subseteq (2^{AP})^\omega$.

E is called a safety property if for all words

$$\sigma = A_0 A_1 A_2 \ldots \in (2^{AP})^\omega \setminus E$$

there exists a finite prefix $A_0 A_1 \ldots A_n$ of σ such that *none* of the words $A_0 A_1 \ldots A_n B_{n+1} B_{n+2} B_{n+3} \ldots$ belongs to E, i.e.,

$$E \cap \{\sigma' \in (2^{AP})^\omega : A_0 \ldots A_n \text{ is a prefix of } \sigma'\} = \varnothing$$

Such words $A_0 A_1 \ldots A_n$ are called bad prefixes for E.

$$BadPref_E \overset{\text{def}}{=} \text{set of bad prefixes for } E$$

Let E be a LT property over AP, i.e., $E \subseteq (2^{AP})^\omega$.

E is called a safety property if for all words

$$\sigma = A_0 A_1 A_2 \ldots \in (2^{AP})^\omega \setminus E$$

there exists a finite prefix $A_0 A_1 \ldots A_n$ of σ such that *none* of the words $A_0 A_1 \ldots A_n B_{n+1} B_{n+2} B_{n+3} \ldots$ belongs to E, i.e.,

$$E \cap \{\sigma' \in (2^{AP})^\omega : A_0 \ldots A_n \text{ is a prefix of } \sigma'\} = \varnothing$$

Such words $A_0 A_1 \ldots A_n$ are called bad prefixes for E.

$$BadPref_E \overset{\text{def}}{=} \text{set of bad prefixes for } E \subseteq (2^{AP})^+$$

Definition of safety properties

Let E be a LT property over AP, i.e., $E \subseteq (2^{AP})^\omega$.

E is called a safety property if for all words

$$\sigma = A_0 A_1 A_2 \ldots \in (2^{AP})^\omega \setminus E$$

there exists a finite prefix $A_0 A_1 \ldots A_n$ of σ such that *none* of the words $A_0 A_1 \ldots A_n B_{n+1} B_{n+2} B_{n+3} \ldots$ belongs to E, i.e.,

$$E \cap \{\sigma' \in (2^{AP})^\omega : A_0 \ldots A_n \text{ is a prefix of } \sigma'\} = \varnothing$$

Such words $A_0 A_1 \ldots A_n$ are called bad prefixes for E.

$BadPref_E \stackrel{\text{def}}{=}$ set of bad prefixes for $E \subseteq (2^{AP})^+$

↑

briefly: $BadPref$

Let E be a LT property over AP, i.e., $E \subseteq (2^{AP})^\omega$.

E is called a safety property if for all words

$$\sigma \;=\; A_0\,A_1\,A_2\ldots \in \left(2^{AP}\right)^\omega \setminus E$$

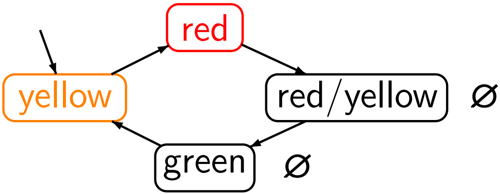
there exists a finite prefix $A_0\,A_1\ldots A_n$ of σ such that *none* of the words $A_0\,A_1\ldots A_n\,B_{n+1}\,B_{n+2}\,B_{n+3}\ldots$ belongs to E, i.e.,

$$E \cap \left\{\sigma' \in (2^{AP})^\omega : A_0\ldots A_n \text{ is a prefix of } \sigma'\right\} = \varnothing$$

Such words $A_0\,A_1\,\ldots\,A_n$ are called bad prefixes for E.

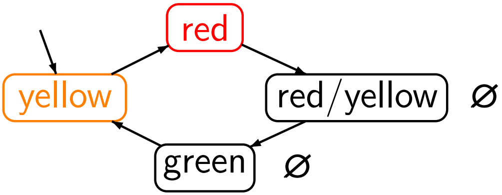
minimal bad prefixes: any word $A_0\ldots A_i\ldots A_n \in BadPref$ s.t. no proper prefix $A_0\ldots A_i$ is a bad prefix for E

Safety property for a traffic light



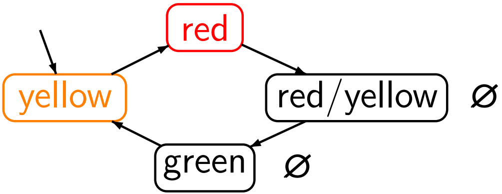
$AP = \{red, yellow\}$

Safety property for a traffic light



"every red phase is preceded by a yellow phase"

Safety property for a traffic light



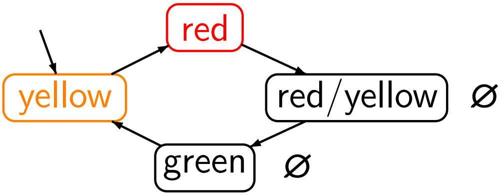
"every red phase is preceded by a yellow phase"

hence: $\mathcal{T} \models E$

E = set of all infinite words $A_0\, A_1\, A_2 \ldots$ over 2^{AP} such that for all $i \in \mathbb{N}$:

$$red \in A_i \implies i \geq 1 \text{ and } yellow \in A_{i-1}$$

Safety property for a traffic light

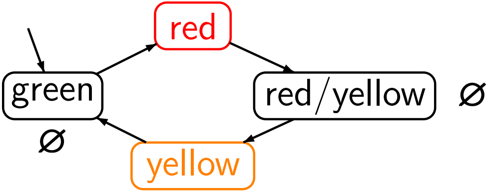


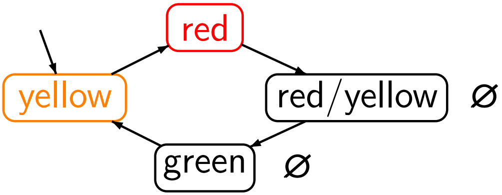
"every red phase is preceded by a yellow phase"

hence: $\mathcal{T} \models E$

E = set of all infinite words $A_0\, A_1\, A_2 \ldots$ over 2^{AP} such that for all $i \in \mathbb{N}$:

$$red \in A_i \implies i \geq 1 \text{ and } yellow \in A_{i-1}$$



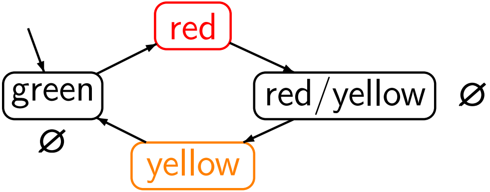


"every red phase is preceded by a yellow phase"

hence: $\mathcal{T} \models E$

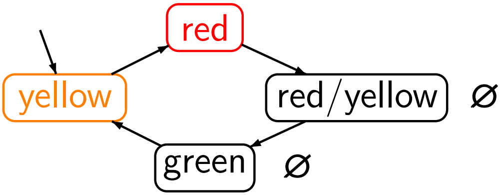
E = set of all infinite words $A_0\,A_1\,A_2\ldots$ over 2^{AP} such that for all $i \in \mathbb{N}$:

$$red \in A_i \implies i \geq 1 \text{ and } yellow \in A_{i-1}$$



"there is a red phase that is not preceded by a yellow phase"

Safety property for a traffic light

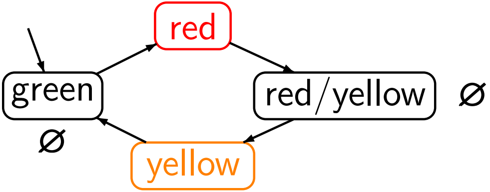


"every red phase is preceded by a yellow phase"

hence: $\mathcal{T} \models E$

E = set of all infinite words $A_0 A_1 A_2 \ldots$ over 2^{AP} such that for all $i \in \mathbb{N}$:

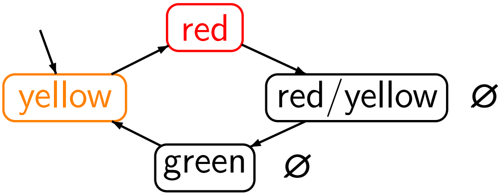
$$red \in A_i \implies i \geq 1 \text{ and } yellow \in A_{i-1}$$



"there is a red phase that is not preceded by a yellow phase"

hence: $\mathcal{T} \not\models E$

Safety property for a traffic light

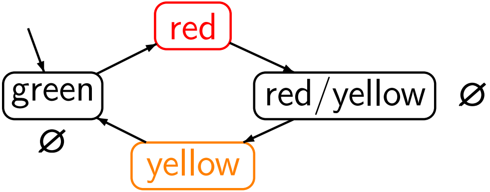


"every red phase is preceded by a yellow phase"

hence: $\mathcal{T} \models E$

E = set of all infinite words $A_0\, A_1\, A_2 \ldots$ over 2^{AP} such that for all $i \in \mathbb{N}$:

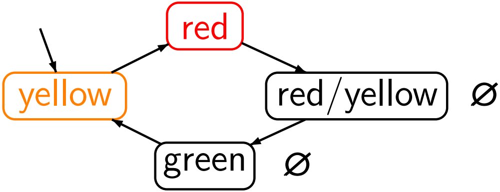
$$red \in A_i \implies i \geq 1 \text{ and } yellow \in A_{i-1}$$



$\mathcal{T} \not\models E$

bad prefix, e.g., $\varnothing\,\{red\}\,\varnothing\,\{yellow\}$

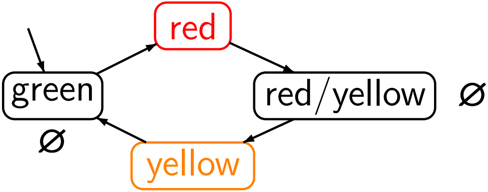
Safety property for a traffic light



"every red phase is preceded by a yellow phase"

hence: $\mathcal{T} \models E$

$$E \;=\; \text{set of all infinite words } A_0\,A_1\,A_2\ldots \text{ over } 2^{AP} \text{ such that for all } i \in \mathbb{N}\text{:}$$
$$red \in A_i \implies i \geq 1 \text{ and } yellow \in A_{i-1}$$

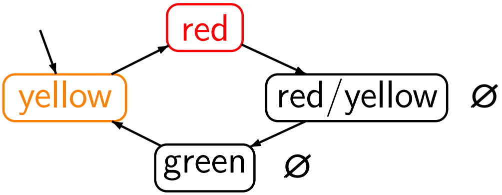


$\mathcal{T} \not\models E$

minimal bad prefix:

$\varnothing\,\{red\}$

Safety property for a traffic light



"every red phase is preceded by a yellow phase"

hence: $\mathcal{T} \models E$

E = set of all infinite words $A_0 A_1 A_2 \ldots$ over 2^{AP} such that for all $i \in \mathbb{N}$:

$$red \in A_i \implies i \geq 1 \text{ and } yellow \in A_{i-1}$$

is a safety property over $AP = \{red, yellow\}$ with

$BadPref$ = set of all finite words $A_0 A_1 \ldots A_n$ over 2^{AP} s.t. for some $i \in \{0, \ldots, n\}$:

$$red \in A_i \wedge (i{=}0 \vee yellow \notin A_{i-1})$$

Satisfaction of safety properties

Satisfaction of safety properties

Let $E \subseteq (2^{AP})^\omega$ be a safety property, $\mathcal{T}$ a TS over AP.

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

$Traces(\mathcal{T}) \quad = \quad$ set of traces of $\mathcal{T}$

Satisfaction of safety properties

Let $E \subseteq (2^{AP})^\omega$ be a safety property, $\mathcal{T}$ a TS over AP.

$$\begin{aligned} \mathcal{T} \models E \quad & \text{iff} \quad Traces(\mathcal{T}) \subseteq E \\ & \text{iff} \quad Traces_{fin}(\mathcal{T}) \cap BadPref = \varnothing \end{aligned}$$

$BadPref$ = set of all bad prefixes of E

$Traces(\mathcal{T})$ = set of traces of $\mathcal{T}$

$Traces_{fin}(\mathcal{T})$ = set of finite traces of $\mathcal{T}$

$= \{ \, trace(\widehat{\pi}) : \widehat{\pi}$ is an initial, finite path fragment of $\mathcal{T} \}$

Satisfaction of safety properties

Let $E \subseteq (2^{AP})^\omega$ be a safety property, $\mathcal{T}$ a TS over AP.

$$\begin{aligned} \mathcal{T} \models E \quad &\text{iff} \quad Traces(\mathcal{T}) \subseteq E \\ &\text{iff} \quad Traces_{fin}(\mathcal{T}) \cap BadPref = \varnothing \\ &\text{iff} \quad Traces_{fin}(\mathcal{T}) \cap MinBadPref = \varnothing \end{aligned}$$

$BadPref$ = set of all bad prefixes of E

$MinBadPref$ = set of all minimal bad prefixes of E

$Traces(\mathcal{T})$ = set of traces of $\mathcal{T}$

$Traces_{fin}(\mathcal{T})$ = set of finite traces of $\mathcal{T}$

$= \{ \, trace(\widehat{\pi}) : \widehat{\pi}$ is an initial, finite path fragment of $\mathcal{T} \}$

Correct or wrong?

Every invariant is a safety property.

Correct or wrong?

IS2.5-13

Every invariant is a safety property.

correct.

Correct or wrong?

Every invariant is a safety property.

correct.

Let E be an invariant with invariant condition Φ.

Every invariant is a safety property.

correct.

Let E be an invariant with invariant condition Φ.

- bad prefixes for E: finite words $A_0 \dots A_i \dots A_n$ s.t.

$$A_i \not\models \Phi \text{ for some } i \in \{0, 1, \dots, n\}$$

Every invariant is a safety property.

correct.

Let E be an invariant with invariant condition Φ.

- bad prefixes for E: finite words $A_0 \ldots A_i \ldots A_n$ s.t.

$$A_i \not\models \Phi \text{ for some } i \in \{0, 1, \ldots, n\}$$

- minimal bad prefixes for E:
  finite words $A_0\, A_1 \ldots A_{n-1}\, A_n$ such that

$$A_i \models \Phi \text{ for } i = 0, 1, \ldots, n{-}1, \text{ and } A_n \not\models \Phi$$

Correct or wrong?

$\varnothing$ is a safety property

Correct or wrong?

$\varnothing$ is a safety property

correct

Correct or wrong?

$\varnothing$ is a safety property

correct

- all finite words $A_0 \ldots A_n \in (2^{AP})^+$ are bad prefixes

Correct or wrong?

$\varnothing$ is a safety property

correct

- all finite words $A_0 \ldots A_n \in (2^{AP})^+$ are bad prefixes
- $\varnothing$ is even an invariant (invariant condition ***false***)

Correct or wrong?

$\varnothing$ is a safety property

correct

- all finite words $A_0 \ldots A_n \in (2^{AP})^+$ are bad prefixes
- $\varnothing$ is even an invariant (invariant condition ***false***)

$(2^{AP})^\omega$ is a safety property

$\varnothing$ is a safety property

correct

- all finite words $A_0 \dots A_n \in (2^{AP})^+$ are bad prefixes
- $\varnothing$ is even an invariant (invariant condition ***false***)

$(2^{AP})^\omega$ is a safety property

correct

$\varnothing$ is a safety property

correct

- all finite words $A_0 \ldots A_n \in (2^{AP})^+$ are bad prefixes
- $\varnothing$ is even an invariant (invariant condition ***false***)

$(2^{AP})^\omega$ is a safety property

correct

$$\text{“For all words} \in \underbrace{(2^{AP})^\omega \setminus (2^{AP})^\omega}_{=\varnothing} \ldots\text{”}$$

Prefix closure

Prefix closure

For a given infinite word $\sigma = A_0\, A_1\, A_2\, \ldots$, let

$$pref(\sigma) \stackrel{\text{def}}{=} \text{set of all nonempty, finite prefixes of } \sigma$$
$$= \{ A_0\, A_1\, \ldots A_n \;:\; n \geq 0 \}$$

Prefix closure

For a given infinite word $\sigma = A_0\, A_1\, A_2\, \ldots$, let

$$pref(\sigma) \stackrel{\text{def}}{=} \text{set of all nonempty, finite prefixes of } \sigma$$
$$= \{ A_0\, A_1 \ldots A_n : n \geq 0 \}$$

For $E \subseteq (2^{AP})^\omega$, let $pref(E) \stackrel{\text{def}}{=} \bigcup_{\sigma \in E} pref(\sigma)$

Prefix closure

For a given infinite word $\sigma = A_0\, A_1\, A_2\, \ldots$, let

$$pref(\sigma) \stackrel{\text{def}}{=} \text{set of all nonempty, finite prefixes of } \sigma$$
$$= \{ A_0\, A_1 \ldots A_n : n \geq 0 \}$$

For $E \subseteq (2^{AP})^\omega$, let $pref(E) \stackrel{\text{def}}{=} \bigcup_{\sigma \in E} pref(\sigma)$

Given an LT property E, the prefix closure of E is:

$$cl(E) \stackrel{\text{def}}{=} \{\sigma \in (2^{AP})^\omega : pref(\sigma) \subseteq pref(E)\}$$

Prefix closure and safety

For any infinite word $\sigma \in (2^{AP})^\omega$, let

$$pref(\sigma) \;=\; \text{set of all nonempty, finite prefixes of } \sigma$$

For any LT property $E \subseteq (2^{AP})^\omega$, let

$$pref(E) \;=\; \bigcup_{\sigma \in E} pref(\sigma) \text{ and}$$

$$cl(E) \;=\; \{\sigma \in (2^{AP})^\omega : pref(\sigma) \subseteq pref(E)\}$$

For any infinite word $\sigma \in \left(2^{AP}\right)^{\omega}$, let

$$pref(\sigma) \;=\; \text{set of all nonempty, finite prefixes of } \sigma$$

For any LT property $E \subseteq \left(2^{AP}\right)^{\omega}$, let

$$pref(E) \;=\; \bigcup_{\sigma \in E} pref(\sigma) \text{ and}$$

$$cl(E) \;=\; \{\sigma \in (2^{AP})^{\omega} : pref(\sigma) \subseteq pref(E)\}$$

Theorem:

E is a safety property iff $cl(E) = E$

Safety and finite trace inclusion

remind: LT properties and trace inclusion:

If $\mathcal{T}_1$ and $\mathcal{T}_2$ are TS over AP then:

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

iff for all LT properties E: $\mathcal{T}_2 \models E \implies \mathcal{T}_1 \models E$

Safety and finite trace inclusion

remind: LT properties and trace inclusion:

If $\mathcal{T}_1$ and $\mathcal{T}_2$ are TS over AP then:

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

iff for all LT properties E: $\mathcal{T}_2 \models E \implies \mathcal{T}_1 \models E$

safety properties and finite trace inclusion:

If $\mathcal{T}_1$ and $\mathcal{T}_2$ are TS over AP then:

$$Traces_{fin}(\mathcal{T}_1) \subseteq Traces_{fin}(\mathcal{T}_2)$$

iff for all safety properties E: $\mathcal{T}_2 \models E \implies \mathcal{T}_1 \models E$

Safety and finite trace inclusion

$$Traces_{fin}(\mathcal{T}_1) \subseteq Traces_{fin}(\mathcal{T}_2)$$

iff for all safety properties E: $\mathcal{T}_2 \models E \implies \mathcal{T}_1 \models E$

Safety and finite trace inclusion

$$Traces_{fin}(\mathcal{T}_1) \subseteq Traces_{fin}(\mathcal{T}_2)$$

iff for all safety properties E: $\mathcal{T}_2 \models E \implies \mathcal{T}_1 \models E$

Proof "$\Longrightarrow$": obvious, as for safety property E:

$$\mathcal{T} \models E \quad \text{iff} \quad Traces_{fin}(\mathcal{T}) \cap BadPref = \varnothing$$

Safety and finite trace inclusion

$$Traces_{fin}(\mathcal{T}_1) \subseteq Traces_{fin}(\mathcal{T}_2)$$

iff for all safety properties E: $\mathcal{T}_2 \models E \implies \mathcal{T}_1 \models E$

Proof "$\Longrightarrow$": obvious, as for safety property E:

$$\mathcal{T} \models E \quad \text{iff} \quad Traces_{fin}(\mathcal{T}) \cap BadPref = \varnothing$$

Hence:

If $\mathcal{T}_2 \models E$ and $Traces_{fin}(\mathcal{T}_1) \subseteq Traces_{fin}(\mathcal{T}_2)$ then:

$$Traces_{fin}(\mathcal{T}_1) \subseteq Traces_{fin}(\mathcal{T}_2)$$

iff for all safety properties E: $\mathcal{T}_2 \models E \implies \mathcal{T}_1 \models E$

Proof "$\Longrightarrow$": obvious, as for safety property E:

$$\mathcal{T} \models E \quad \text{iff} \quad Traces_{fin}(\mathcal{T}) \cap BadPref = \varnothing$$

Hence:

If $\mathcal{T}_2 \models E$ and $Traces_{fin}(\mathcal{T}_1) \subseteq Traces_{fin}(\mathcal{T}_2)$ then:

$$Traces_{fin}(\mathcal{T}_1) \cap BadPref \subseteq Traces_{fin}(\mathcal{T}_2) \cap BadPref = \varnothing$$

Safety and finite trace inclusion

$$Traces_{fin}(\mathcal{T}_1) \subseteq Traces_{fin}(\mathcal{T}_2)$$

iff for all safety properties E: $\mathcal{T}_2 \models E \implies \mathcal{T}_1 \models E$

Proof "$\Longrightarrow$": obvious, as for safety property E:

$$\mathcal{T} \models E \quad \text{iff} \quad Traces_{fin}(\mathcal{T}) \cap BadPref = \varnothing$$

Hence:

If $\mathcal{T}_2 \models E$ and $Traces_{fin}(\mathcal{T}_1) \subseteq Traces_{fin}(\mathcal{T}_2)$ then:

$$Traces_{fin}(\mathcal{T}_1) \cap BadPref \subseteq Traces_{fin}(\mathcal{T}_2) \cap BadPref = \varnothing$$

and therefore $\mathcal{T}_1 \models E$

Safety and finite trace inclusion

$$Traces_{fin}(\mathcal{T}_1) \subseteq Traces_{fin}(\mathcal{T}_2)$$

iff for all safety properties E: $\mathcal{T}_2 \models E \implies \mathcal{T}_1 \models E$

Proof "$\Longleftarrow$": consider the LT property

$$E = cl(Traces(\mathcal{T}_2))$$

Safety and finite trace inclusion

$$Traces_{fin}(\mathcal{T}_1) \subseteq Traces_{fin}(\mathcal{T}_2)$$

iff for all safety properties E: $\mathcal{T}_2 \models E \implies \mathcal{T}_1 \models E$

Proof "$\Longleftarrow$": consider the LT property

$$E = cl(Traces(\mathcal{T}_2)) = \{\sigma : pref(\sigma) \subseteq Traces_{fin}(\mathcal{T}_2)\}$$

Safety and finite trace inclusion

$$Traces_{fin}(\mathcal{T}_1) \subseteq Traces_{fin}(\mathcal{T}_2)$$

iff for all safety properties E: $\mathcal{T}_2 \models E \implies \mathcal{T}_1 \models E$

Proof "$\Longleftarrow$": consider the LT property

$$E = cl(Traces(\mathcal{T}_2)) = \{\sigma : pref(\sigma) \subseteq Traces_{fin}(\mathcal{T}_2)\}$$

↑

for each transition system $\mathcal{T}$:

$$pref(Traces(\mathcal{T})) = Traces_{fin}(\mathcal{T})$$

Safety and finite trace inclusion

$$Traces_{fin}(\mathcal{T}_1) \subseteq Traces_{fin}(\mathcal{T}_2)$$

iff for all safety properties E: $\mathcal{T}_2 \models E \implies \mathcal{T}_1 \models E$

Proof "$\Longleftarrow$": consider the LT property

$$E = cl(Traces(\mathcal{T}_2)) = \{\sigma : pref(\sigma) \subseteq Traces_{fin}(\mathcal{T}_2)\}$$

Then, E is a safety property

Safety and finite trace inclusion

$$Traces_{fin}(\mathcal{T}_1) \subseteq Traces_{fin}(\mathcal{T}_2)$$

iff for all safety properties E: $\mathcal{T}_2 \models E \implies \mathcal{T}_1 \models E$

Proof "$\Longleftarrow$": consider the LT property

$$E = cl(Traces(\mathcal{T}_2)) = \{\sigma : pref(\sigma) \subseteq Traces_{fin}(\mathcal{T}_2)\}$$

Then, E is a safety property

as $cl(E) = E$

Safety and finite trace inclusion

$Traces_{fin}(\mathcal{T}_1) \subseteq Traces_{fin}(\mathcal{T}_2)$

iff for all safety properties E: $\mathcal{T}_2 \models E \implies \mathcal{T}_1 \models E$

Proof "$\Longleftarrow$": consider the LT property

$$E = cl(Traces(\mathcal{T}_2)) = \{\sigma : pref(\sigma) \subseteq Traces_{fin}(\mathcal{T}_2)\}$$

Then, E is a safety property

as $cl(E) = E$

set of bad prefixes: $(2^{AP})^+ \setminus Traces_{fin}(\mathcal{T}_2)$

Safety and finite trace inclusion

$$Traces_{fin}(\mathcal{T}_1) \subseteq Traces_{fin}(\mathcal{T}_2)$$

iff for all safety properties E: $\mathcal{T}_2 \models E \implies \mathcal{T}_1 \models E$

Proof "$\Longleftarrow$": consider the LT property

$$E = cl(Traces(\mathcal{T}_2)) = \{\sigma : pref(\sigma) \subseteq Traces_{fin}(\mathcal{T}_2)\}$$

Then, E is a safety property and $\mathcal{T}_2 \models E$.

Safety and finite trace inclusion

$$\mathit{Traces}_{\mathit{fin}}(\mathcal{T}_1) \subseteq \mathit{Traces}_{\mathit{fin}}(\mathcal{T}_2)$$

iff for all safety properties E: $\mathcal{T}_2 \models E \implies \mathcal{T}_1 \models E$

Proof "$\Longleftarrow$": consider the LT property

$$E = \mathit{cl}(\mathit{Traces}(\mathcal{T}_2)) = \{\sigma : \mathit{pref}(\sigma) \subseteq \mathit{Traces}_{\mathit{fin}}(\mathcal{T}_2)\}$$

Then, E is a safety property and $\mathcal{T}_2 \models E$.

By assumption: $\mathcal{T}_1 \models E$

Safety and finite trace inclusion

$$Traces_{fin}(\mathcal{T}_1) \subseteq Traces_{fin}(\mathcal{T}_2)$$

iff for all safety properties E: $\mathcal{T}_2 \models E \implies \mathcal{T}_1 \models E$

Proof "$\Longleftarrow$": consider the LT property

$$E = cl(Traces(\mathcal{T}_2)) = \{\sigma : pref(\sigma) \subseteq Traces_{fin}(\mathcal{T}_2)\}$$

Then, E is a safety property and $\mathcal{T}_2 \models E$.

By assumption: $\mathcal{T}_1 \models E$ and therefore $Traces(\mathcal{T}_1) \subseteq E$.

Safety and finite trace inclusion

$$Traces_{fin}(\mathcal{T}_1) \subseteq Traces_{fin}(\mathcal{T}_2)$$

iff for all safety properties E: $\mathcal{T}_2 \models E \implies \mathcal{T}_1 \models E$

Proof "$\Longleftarrow$": consider the LT property

$$E = cl(Traces(\mathcal{T}_2)) = \{\sigma : pref(\sigma) \subseteq Traces_{fin}(\mathcal{T}_2)\}$$

Then, E is a safety property and $\mathcal{T}_2 \models E$.

By assumption: $\mathcal{T}_1 \models E$ and therefore $Traces(\mathcal{T}_1) \subseteq E$.

Hence: $Traces_{fin}(\mathcal{T}_1) = pref(Traces(\mathcal{T}_1))$

Safety and finite trace inclusion

$$Traces_{fin}(\mathcal{T}_1) \subseteq Traces_{fin}(\mathcal{T}_2)$$

iff for all safety properties E: $\mathcal{T}_2 \models E \implies \mathcal{T}_1 \models E$

Proof "$\Longleftarrow$": consider the LT property

$$E = cl(Traces(\mathcal{T}_2)) = \{\sigma : pref(\sigma) \subseteq Traces_{fin}(\mathcal{T}_2)\}$$

Then, E is a safety property and $\mathcal{T}_2 \models E$.

By assumption: $\mathcal{T}_1 \models E$ and therefore $Traces(\mathcal{T}_1) \subseteq E$.

Hence: $Traces_{fin}(\mathcal{T}_1) = pref(Traces(\mathcal{T}_1))$

$$\subseteq pref(E)$$

Safety and finite trace inclusion

$$Traces_{fin}(\mathcal{T}_1) \subseteq Traces_{fin}(\mathcal{T}_2)$$

iff for all safety properties E: $\mathcal{T}_2 \models E \implies \mathcal{T}_1 \models E$

Proof "$\Longleftarrow$": consider the LT property

$$E = cl(Traces(\mathcal{T}_2)) = \{\sigma : pref(\sigma) \subseteq Traces_{fin}(\mathcal{T}_2)\}$$

Then, E is a safety property and $\mathcal{T}_2 \models E$.

By assumption: $\mathcal{T}_1 \models E$ and therefore $Traces(\mathcal{T}_1) \subseteq E$.

Hence: $Traces_{fin}(\mathcal{T}_1) = pref(Traces(\mathcal{T}_1))$

$$\subseteq pref(E) = pref(cl(Traces(\mathcal{T}_2)))$$

Safety and finite trace inclusion

$$Traces_{fin}(\mathcal{T}_1) \subseteq Traces_{fin}(\mathcal{T}_2)$$

iff for all safety properties E: $\mathcal{T}_2 \models E \implies \mathcal{T}_1 \models E$

Proof "$\Longleftarrow$": consider the LT property

$$E = cl(Traces(\mathcal{T}_2)) = \{\sigma : pref(\sigma) \subseteq Traces_{fin}(\mathcal{T}_2)\}$$

Then, E is a safety property and $\mathcal{T}_2 \models E$.

By assumption: $\mathcal{T}_1 \models E$ and therefore $Traces(\mathcal{T}_1) \subseteq E$.

Hence: $Traces_{fin}(\mathcal{T}_1) = pref(Traces(\mathcal{T}_1))$

$$\subseteq pref(E) = pref(cl(Traces(\mathcal{T}_2)))$$

$$= Traces_{fin}(\mathcal{T}_2)$$

Safety and finite trace equivalence

Safety and finite trace equivalence

safety properties and finite trace inclusion:

If $\mathcal{T}_1$ and $\mathcal{T}_2$ are TS over AP then:

$$Traces_{fin}(\mathcal{T}_1) \subseteq Traces_{fin}(\mathcal{T}_2)$$

iff for all safety properties E: $\mathcal{T}_2 \models E \implies \mathcal{T}_1 \models E$

Safety and finite trace equivalence

safety properties and finite trace inclusion:

If $\mathcal{T}_1$ and $\mathcal{T}_2$ are TS over AP then:

$$Traces_{fin}(\mathcal{T}_1) \subseteq Traces_{fin}(\mathcal{T}_2)$$

iff for all safety properties E: $\mathcal{T}_2 \models E \implies \mathcal{T}_1 \models E$

safety properties and finite trace equivalence:

If $\mathcal{T}_1$ and $\mathcal{T}_2$ are TS over AP then:

$$Traces_{fin}(\mathcal{T}_1) = Traces_{fin}(\mathcal{T}_2)$$

iff $\mathcal{T}_1$ and $\mathcal{T}_2$ satisfy the same safety properties

Summary: 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

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

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

Summary: 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

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

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

correct or wrong?

If $\mathit{Traces}(\mathcal{T}) \subseteq \mathit{Traces}(\mathcal{T}')$
then $\mathit{Traces_{fin}}(\mathcal{T}) \subseteq \mathit{Traces_{fin}}(\mathcal{T}')$.

If $\mathit{Traces}(\mathcal{T}) \subseteq \mathit{Traces}(\mathcal{T}')$
then $\mathit{Traces}_{\mathit{fin}}(\mathcal{T}) \subseteq \mathit{Traces}_{\mathit{fin}}(\mathcal{T}')$.

correct, since

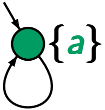
$$\begin{aligned} \mathit{Traces}_{\mathit{fin}}(\mathcal{T}) &= \text{set of all finite nonempty prefixes} \\ & \text{of words in } \mathit{Traces}(\mathcal{T}) \\ &= \mathit{pref}(\mathit{Traces}(\mathcal{T})) \end{aligned}$$

correct or wrong?

If $Traces(\mathcal{T}) \subseteq Traces(\mathcal{T}')$
then $Traces_{fin}(\mathcal{T}) \subseteq Traces_{fin}(\mathcal{T}')$.

correct, since

$$Traces_{fin}(\mathcal{T}) = \text{set of all finite nonempty prefixes of words in } Traces(\mathcal{T})$$
$$= pref(Traces(\mathcal{T}))$$



$$Traces(\mathcal{T}) = \{ \{a\}^{\omega} \}$$
$$Traces_{fin}(\mathcal{T}) = \{ \{a\}^{n} : n \geq 1 \}$$

Finite trace relations versus trace relations

is trace equivalence the same as
finite trace equivalence **?**

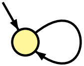
Finite trace relations versus trace relations

is trace equivalence the same as
finite trace equivalence **?**

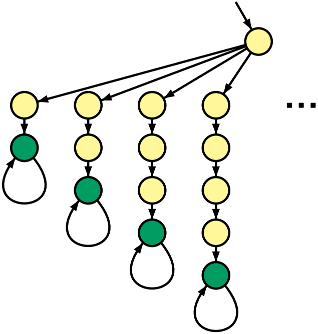
answer: **no**

Finite trace relations versus trace relations

$\mathcal{T}$



$\mathcal{T}'$



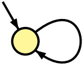
$\bigcirc \mathrel{\widehat{=}} \varnothing$ $\quad \bullet \mathrel{\widehat{=}} \{b\}$

set of propositions

$AP = \{b\}$

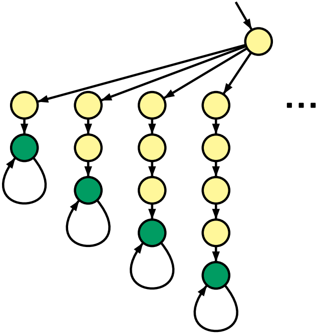
Finite trace relations versus trace relations

$\mathcal{T}$



$Traces(\mathcal{T}) \quad = \{\varnothing^\omega\}$

$\mathcal{T}'$

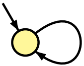


$\bigcirc \ \widehat{=}\ \varnothing \qquad \bullet \ \widehat{=}\ \{b\}$

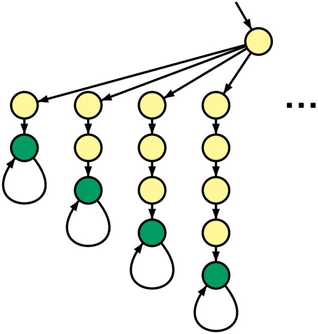
set of propositions
$AP = \{b\}$

Finite trace relations versus trace relations

$\mathcal{T}$



$\mathcal{T}'$



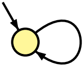
$$Traces(\mathcal{T}) = \{\varnothing^\omega\}$$
$$Traces_{fin}(\mathcal{T}) = \{\varnothing^n : n \geq 0\}$$

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

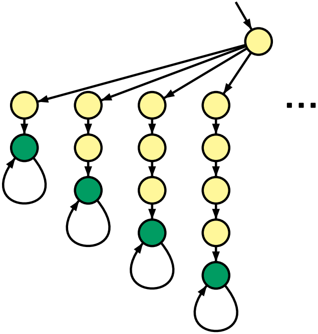
set of propositions
$AP = \{b\}$

Finite trace relations versus trace relations

$\mathcal{T}$



$\mathcal{T}'$



$$Traces(\mathcal{T}) = \{\varnothing^\omega\}$$

$$Traces_{fin}(\mathcal{T}) = \{\varnothing^n : n \geq 0\}$$

$$Traces(\mathcal{T}') = \{\varnothing^n\{b\}^\omega : n \geq 2\}$$

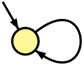
 $\widehat{=} \varnothing$  $\widehat{=} \{b\}$

set of propositions

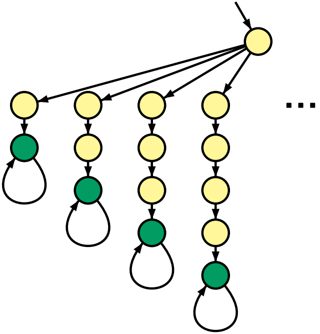
$AP = \{b\}$

Finite trace relations versus trace relations

$\mathcal{T}$



$\mathcal{T}'$



$$Traces(\mathcal{T}) = \{\varnothing^\omega\}$$

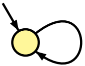
$$Traces_{fin}(\mathcal{T}) = \{\varnothing^n : n \geq 0\}$$

$$Traces(\mathcal{T}') = \{\varnothing^n\{b\}^\omega : n \geq 2\}$$

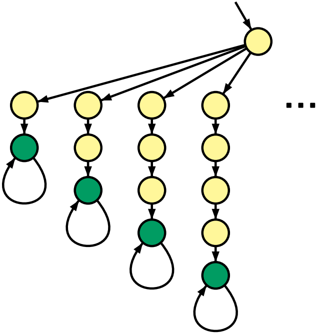
$$Traces_{fin}(\mathcal{T}') = \{\varnothing^n : n \geq 0\} \cup \{\varnothing^n\{b\}^m : n \geq 2 \wedge m \geq 1\}$$

Finite trace relations versus trace relations

$\mathcal{T}$



$\mathcal{T}'$



$$Traces(\mathcal{T}) = \{\varnothing^\omega\}$$

$$Traces_{fin}(\mathcal{T}) = \{\varnothing^n : n \geq 0\}$$

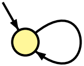
$$Traces(\mathcal{T}') = \{\varnothing^n \{b\}^\omega : n \geq 2\}$$

$$Traces_{fin}(\mathcal{T}') = \{\varnothing^n : n \geq 0\} \cup \{\varnothing^n \{b\}^m : n \geq 2 \wedge m \geq 1\}$$

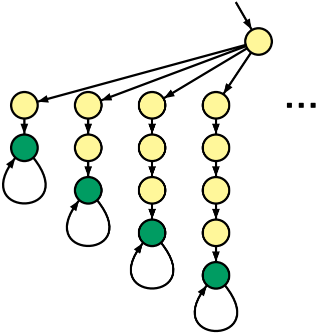
$Traces(\mathcal{T}) \not\subseteq Traces(\mathcal{T}')$, but
$Traces_{fin}(\mathcal{T}) \subseteq Traces_{fin}(\mathcal{T}')$

Finite trace relations versus trace relations

$\mathcal{T}$



$\mathcal{T}'$



$$Traces(\mathcal{T}) = \{\varnothing^\omega\}$$
$$Traces_{fin}(\mathcal{T}) = \{\varnothing^n : n \geq 0\}$$
$$Traces(\mathcal{T}') = \{\varnothing^n \{b\}^\omega : n \geq 2\}$$
$$Traces_{fin}(\mathcal{T}') = \{\varnothing^n : n \geq 0\} \cup \{\varnothing^n \{b\}^m : n \geq 2 \wedge m \geq 1\}$$

$Traces(\mathcal{T}) \not\subseteq Traces(\mathcal{T}')$, but
$Traces_{fin}(\mathcal{T}) \subseteq Traces_{fin}(\mathcal{T}')$

LT property
$E \mathrel{\hat{=}}$ "eventually b"

$\mathcal{T} \not\models E$, $\mathcal{T}' \models E$

Finite trace and trace inclusion

Suppose that $\mathcal{T}$ and $\mathcal{T}'$ are TS over AP such that

(1) $\mathcal{T}$ has no terminal states,

(2) $\mathcal{T}'$ is finite.

Finite trace and trace inclusion

Suppose that $\mathcal{T}$ and $\mathcal{T}'$ are TS over AP such that

(1) $\mathcal{T}$ has no terminal states,
i.e., all paths of $\mathcal{T}$ are infinite

(2) $\mathcal{T}'$ is finite.

Finite trace and trace inclusion

Suppose that $\mathcal{T}$ and $\mathcal{T}'$ are TS over AP such that

(1) $\mathcal{T}$ has no terminal states,
i.e., all paths of $\mathcal{T}$ are infinite

(2) $\mathcal{T}'$ is finite.

Then:

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

$$\text{iff} \quad Traces_{fin}(\mathcal{T}) \subseteq Traces_{fin}(\mathcal{T}')$$

Finite trace and trace inclusion

Suppose that $\mathcal{T}$ and $\mathcal{T}'$ are TS over *AP* such that

(1) $\mathcal{T}$ has no terminal states,
i.e., all paths of $\mathcal{T}$ are infinite

(2) $\mathcal{T}'$ is finite.

Then:

$$Traces(\mathcal{T}) \subseteq Traces(\mathcal{T}')$$
$$\text{iff} \quad Traces_{fin}(\mathcal{T}) \subseteq Traces_{fin}(\mathcal{T}')$$

"$\Longrightarrow$": holds for all transition systems,
no matter whether (1) and (2) hold

Suppose that $\mathcal{T}$ and $\mathcal{T}'$ are TS over AP such that

(1) $\mathcal{T}$ has no terminal states, i.e., all paths of $\mathcal{T}$ are infinite

(2) $\mathcal{T}'$ is finite.

Then:

$$Traces(\mathcal{T}) \subseteq Traces(\mathcal{T}') \quad \text{iff} \quad Traces_{fin}(\mathcal{T}) \subseteq Traces_{fin}(\mathcal{T}')$$

"$\Longrightarrow$": holds for all transition systems

"$\Longleftarrow$": suppose that (1) and (2) hold and that

(3) $Traces_{fin}(\mathcal{T}) \subseteq Traces_{fin}(\mathcal{T}')$

Show that $Traces(\mathcal{T}) \subseteq Traces(\mathcal{T}')$

Finite trace and trace inclusion

Suppose that $\mathcal{T}$ and $\mathcal{T}'$ are TS over ***AP*** such that

(1) $\mathcal{T}$ has no terminal states

(2) $\mathcal{T}'$ is finite

(3) $Traces_{fin}(\mathcal{T}) \subseteq Traces_{fin}(\mathcal{T}')$

Then $Traces(\mathcal{T}) \subseteq Traces(\mathcal{T}')$

Proof:

Finite trace and trace inclusion

Suppose that $\mathcal{T}$ and $\mathcal{T}'$ are TS over AP such that

(1) $\mathcal{T}$ has no terminal states

(2) $\mathcal{T}'$ is finite

(3) $Traces_{fin}(\mathcal{T}) \subseteq Traces_{fin}(\mathcal{T}')$

Then $Traces(\mathcal{T}) \subseteq Traces(\mathcal{T}')$

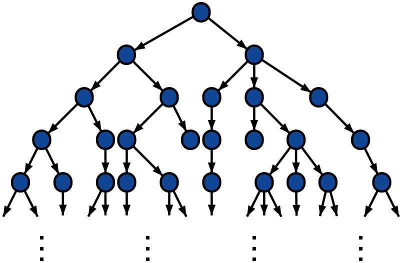
Proof: Pick some path $\pi = s_0\, s_1\, s_2 \ldots$ in $\mathcal{T}$ and show that there exists a path

$$\pi' = t_0\, t_1\, t_2 \ldots \text{ in } \mathcal{T}'$$

such that $trace(\pi) = trace(\pi')$

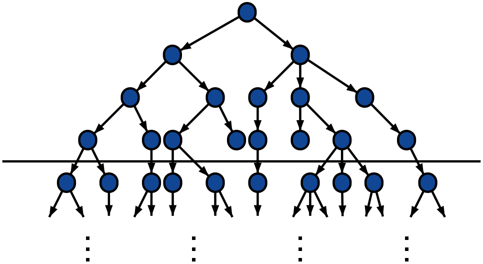
finite TS $\mathcal{T}'$

paths from state t_0
(unfolded into a tree)



finite TS $\mathcal{T}'$

paths from state t_0
(unfolded into a tree)



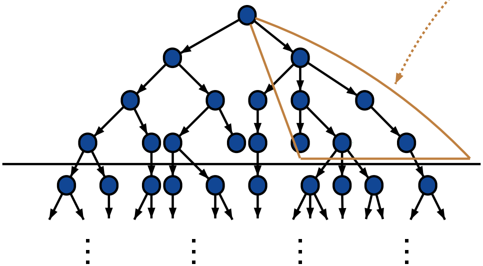
finite until depth $\leq n$

Tracesfin versus traces

finite TS $\mathcal{T}'$

paths from state t_0
(unfolded into a tree)

contains all path fragments with trace $A_0\, A_1 \ldots A_n$



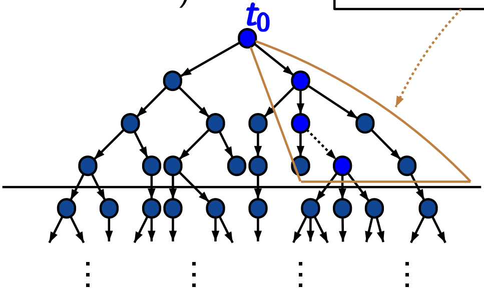
finite until depth $\leq n$

Tracesfin versus 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: $t_0\, t_1 \ldots t_n$



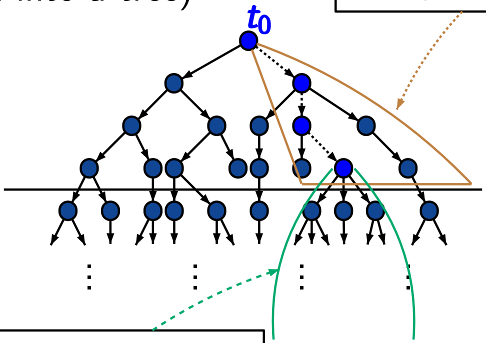
finite until
depth $\leq n$

Tracesfin versus traces

finite TS $\mathcal{T}'$

paths from state t_0
(unfolded into a tree)

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



finite until
depth $\leq n$

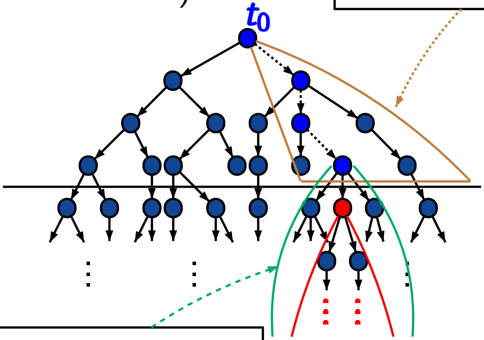
contains infinitely
many path fragments
$t_n\, s^m_{n+1} \dots s^m_m$

Tracesfin versus 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

Finite trace and trace inclusion

Suppose that $\mathcal{T}$ and $\mathcal{T}'$ are TS over AP such that

(1) $\mathcal{T}$ has no terminal states

(2) $\mathcal{T}'$ is finite ⟵ image-finiteness is sufficient

(3) $Traces_{fin}(\mathcal{T}) \subseteq Traces_{fin}(\mathcal{T}')$

Then $Traces(\mathcal{T}) \subseteq Traces(\mathcal{T}')$

Finite trace and trace inclusion

Suppose that $\mathcal{T}$ and $\mathcal{T}'$ are TS over AP such that

(1) $\mathcal{T}$ has no terminal states

(2) $\mathcal{T}'$ is finite ⟵ image-finiteness is sufficient

(3) $Traces_{fin}(\mathcal{T}) \subseteq Traces_{fin}(\mathcal{T}')$

Then $Traces(\mathcal{T}) \subseteq Traces(\mathcal{T}')$

image-finiteness of $\mathcal{T}' = (S', Act, \rightarrow, S'_0, AP, L')$:

Finite trace and trace inclusion

Suppose that $\mathcal{T}$ and $\mathcal{T}'$ are TS over AP such that

(1) $\mathcal{T}$ has no terminal states

(2) $\mathcal{T}'$ is finite ⟵ image-finiteness is sufficient

(3) $Traces_{fin}(\mathcal{T}) \subseteq Traces_{fin}(\mathcal{T}')$

Then $Traces(\mathcal{T}) \subseteq Traces(\mathcal{T}')$

image-finiteness of $\mathcal{T}' = (S', Act, \rightarrow, S'_0, AP, L')$:

- for each $A \in 2^{AP}$ and state $s \in S'$:

  $\{t \in Post(s) : L'(t) = A\}$ is finite

Suppose that $\mathcal{T}$ and $\mathcal{T}'$ are TS over AP such that

(1) $\mathcal{T}$ has no terminal states

(2) $\mathcal{T}'$ is finite ⟵ image-finiteness is sufficient

(3) $Traces_{fin}(\mathcal{T}) \subseteq Traces_{fin}(\mathcal{T}')$

Then $Traces(\mathcal{T}) \subseteq Traces(\mathcal{T}')$

image-finiteness of $\mathcal{T}' = (S', Act, \rightarrow, S'_0, AP, L')$:

- for each $A \in 2^{AP}$ and state $s \in S'$:

  $\{t \in Post(s) : L'(t) = A\}$ is finite

- for each $A \in 2^{AP}$: $\{s_0 \in S'_0 : L'(s_0) = A\}$ is finite

Trace equivalence vs. finite trace equivalence

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

Trace equivalence vs. finite trace equivalence

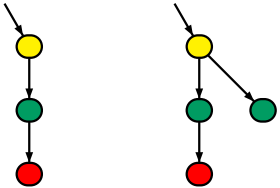
Whenever $\mathit{Traces}(\mathcal{T}) = \mathit{Traces}(\mathcal{T}')$ then
$\mathit{Traces}_{\mathit{fin}}(\mathcal{T}) = \mathit{Traces}_{\mathit{fin}}(\mathcal{T}')$

while the reverse direction does not hold in general
(even not for finite transition systems)

Trace equivalence vs. finite trace equivalence

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

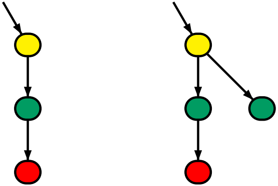
while the reverse direction does not hold in general
(even not for finite transition systems)



Trace equivalence vs. finite trace equivalence

Whenever $Traces(\mathcal{T}) = Traces(\mathcal{T}')$ then
$Traces_{fin}(\mathcal{T}) = Traces_{fin}(\mathcal{T}')$

while the reverse direction does not hold in general
(even not for finite transition systems)



finite trace equivalent,
but *not* trace equivalent

Trace equivalence vs. finite trace equivalence

Whenever $Traces(\mathcal{T}) = Traces(\mathcal{T}')$ then $Traces_{fin}(\mathcal{T}) = Traces_{fin}(\mathcal{T}')$

The reverse implication holds under additional assumptions, e.g.,

- if $\mathcal{T}$ and $\mathcal{T}'$ are finite and have no terminal states
- or, if $\mathcal{T}$ and $\mathcal{T}'$ are AP-deterministic