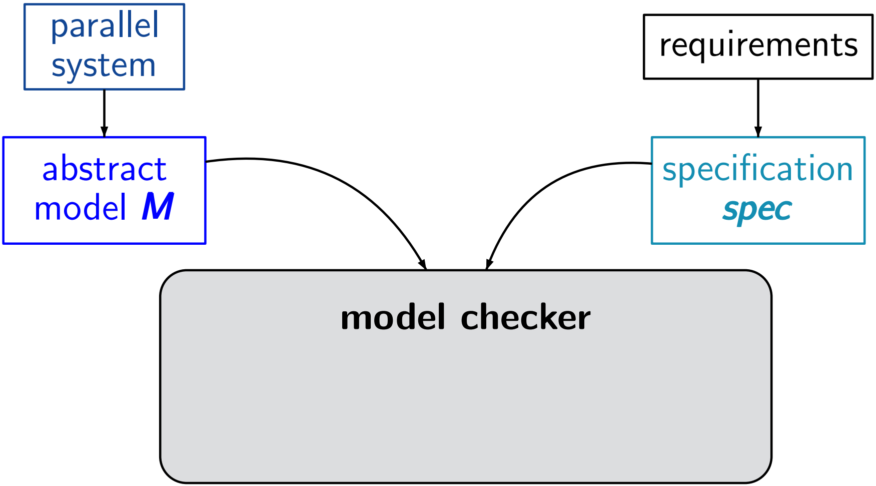
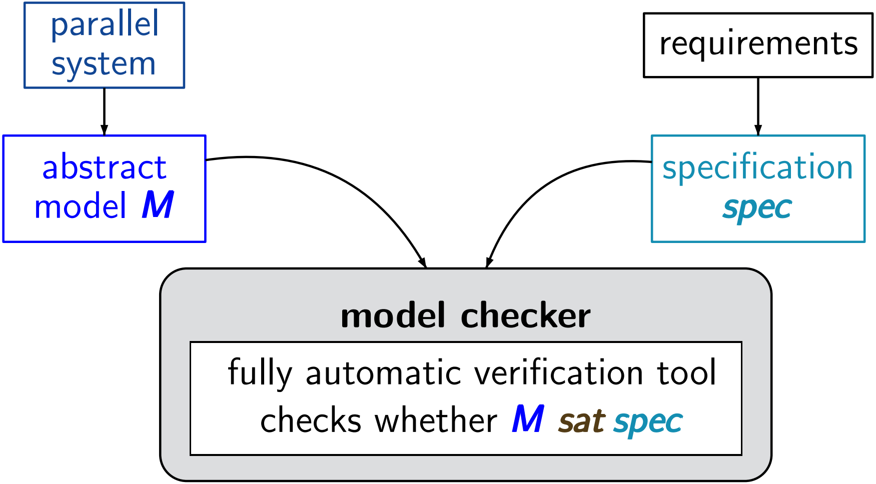
Model checking

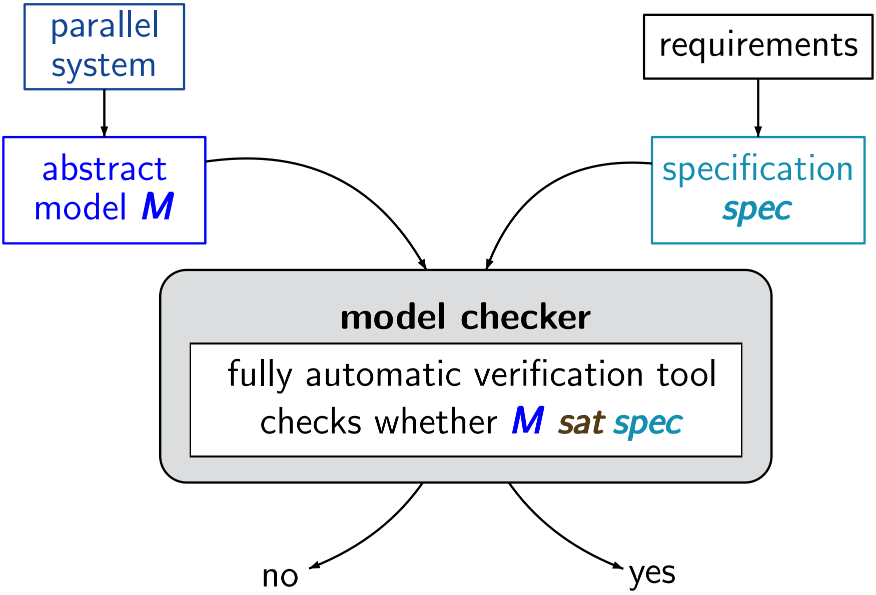
Model checking



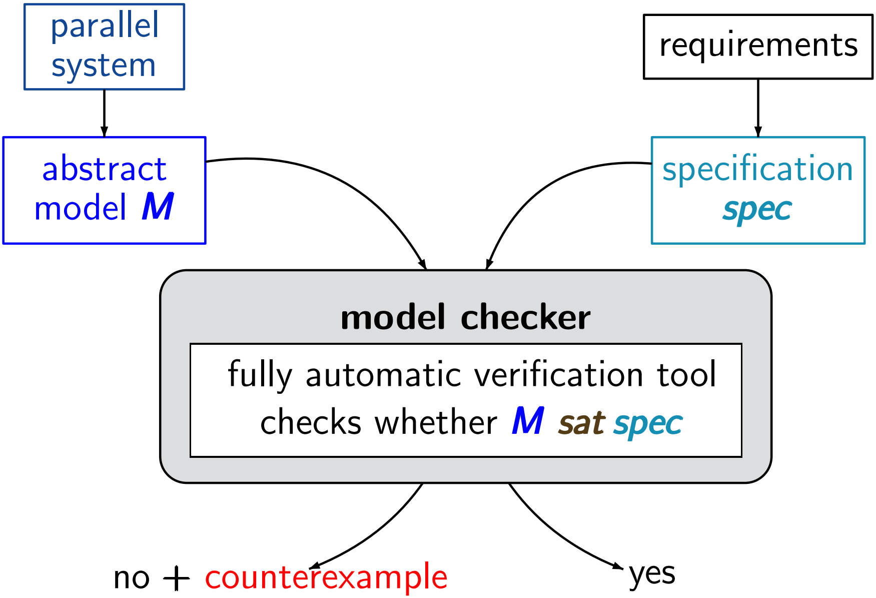
Model checking



Model checking

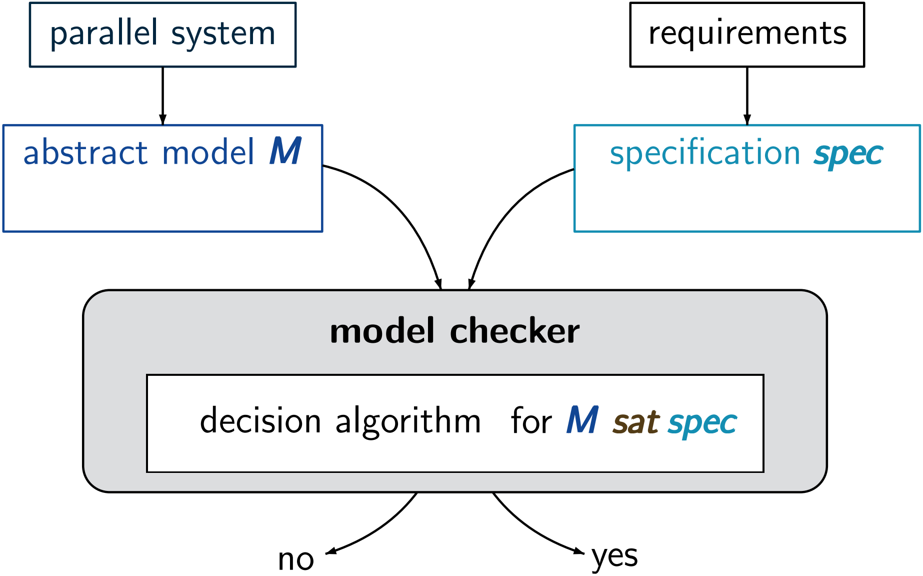


Model checking

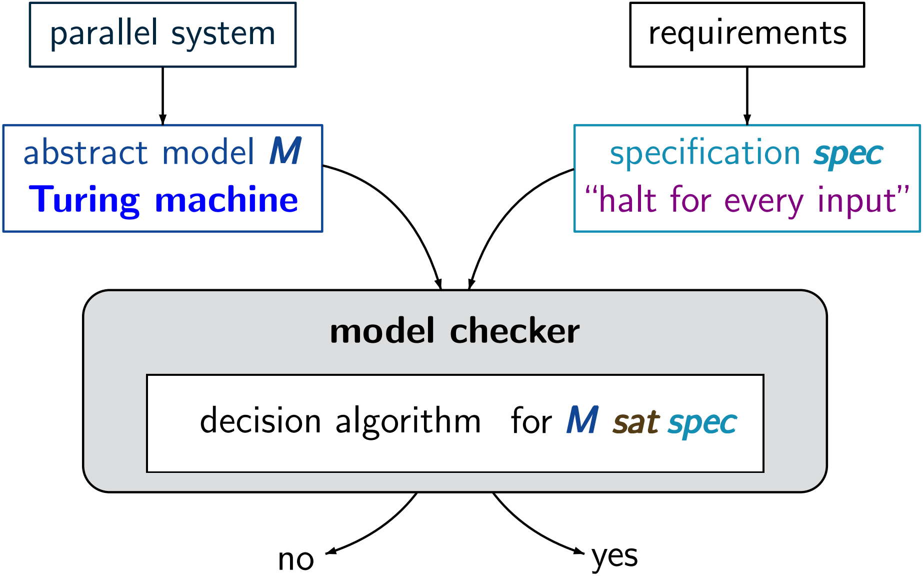


Decidability of the model checking problem?

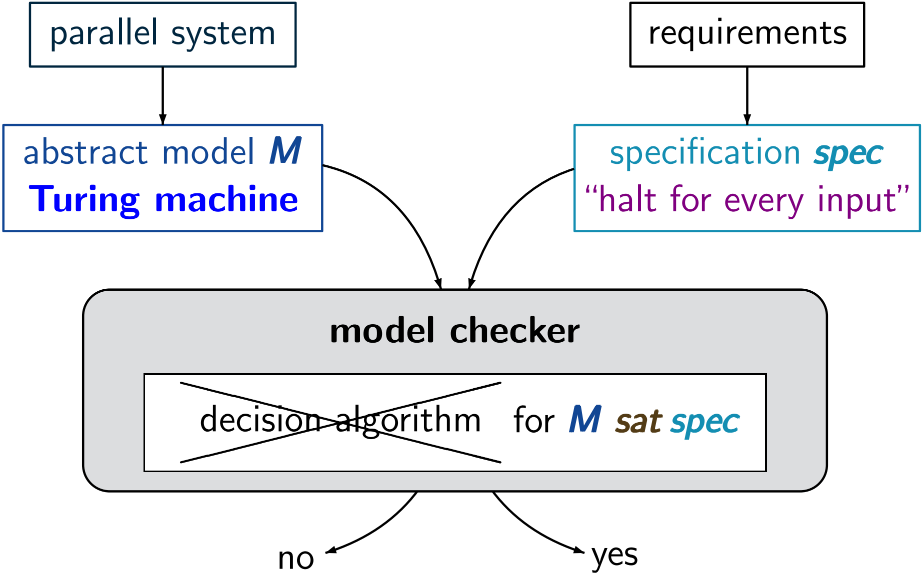
Decidability of the model checking problem?



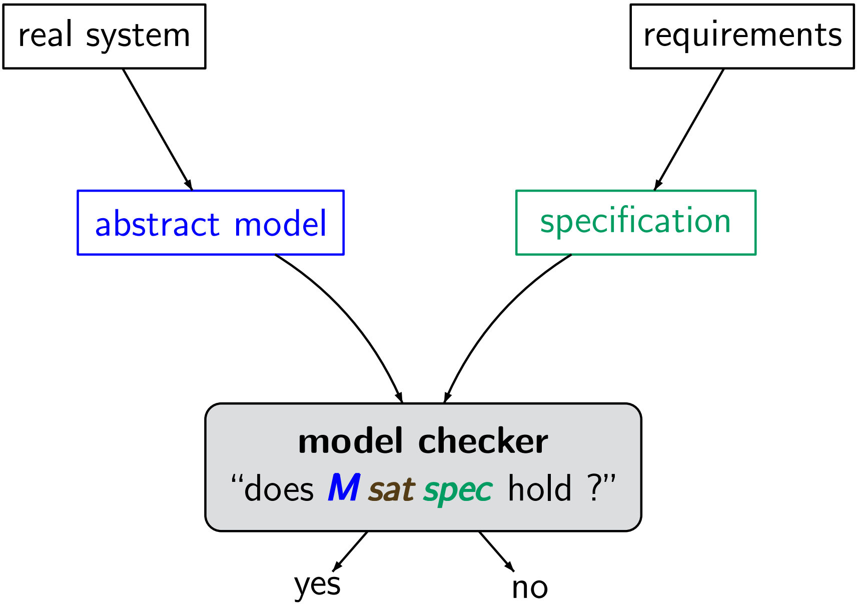
Decidability of the model checking problem?



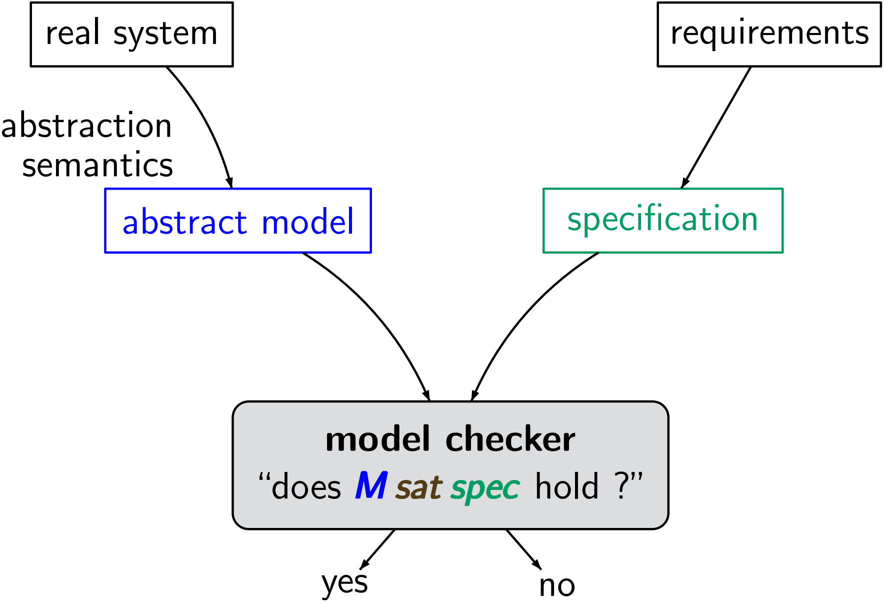
General model checking problem is undecidable



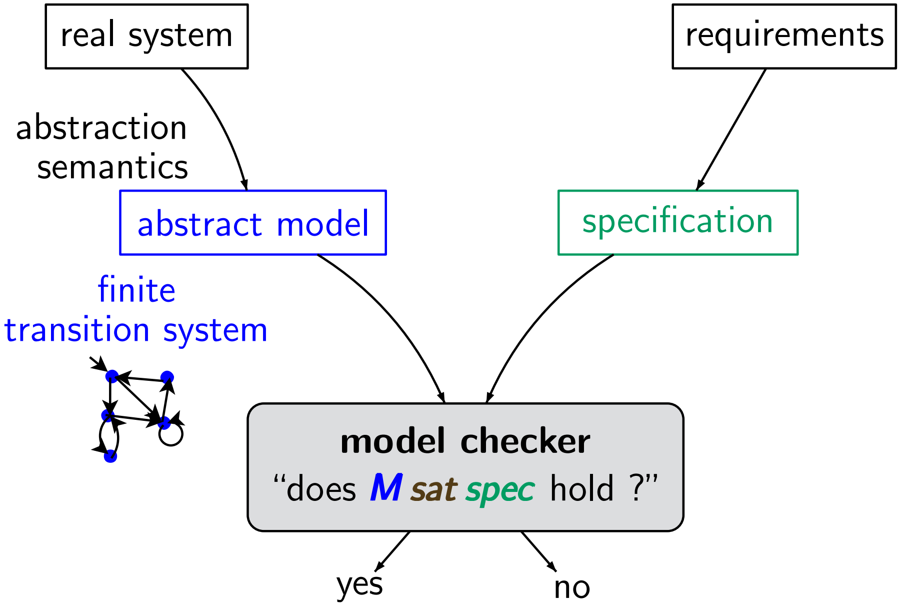
To ensure decidability . . .



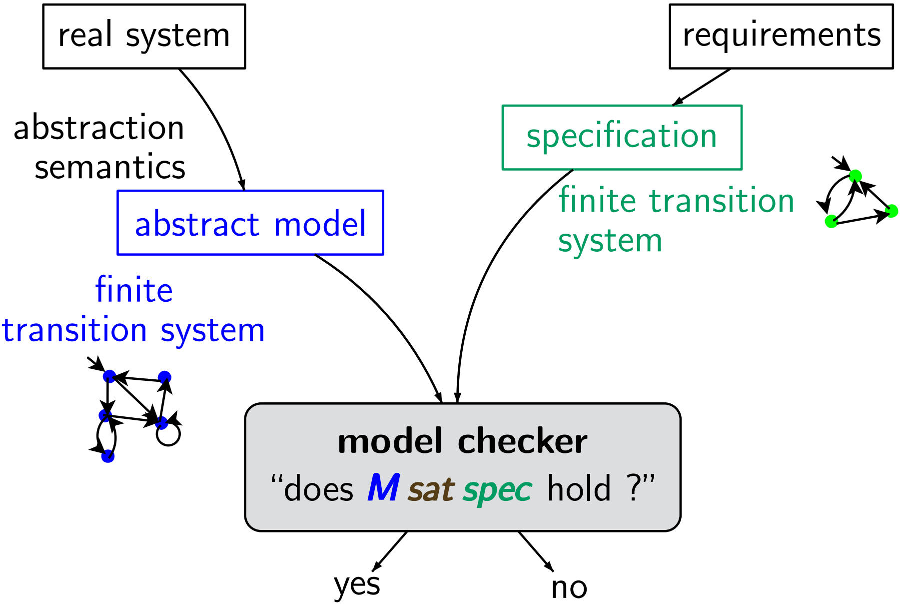
To ensure decidability . . .



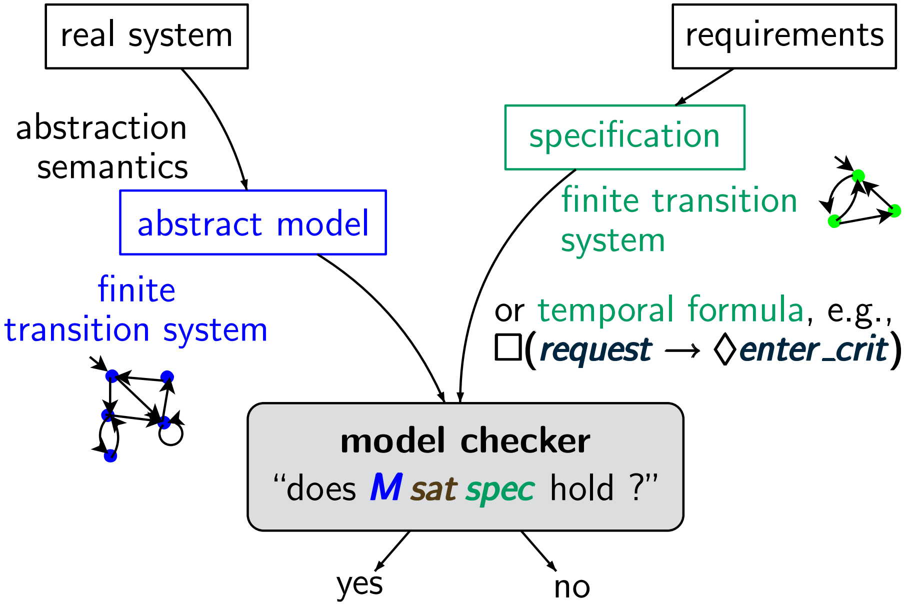
To ensure decidability . . .



To ensure decidability . . .



To ensure decidability . . .



Model checking vs other validation techniques

The validation techniques (testing, simulation, deductive verification, model checking) are complementary to each other.

Model checking vs other validation techniques

The validation techniques (testing, simulation, deductive verification, model checking) are complementary to each other.

model checking

- most efficient validation technique, fully automatic
- but mostly only applicable for finite models with "small" (or "sufficiently structured") state space
- industrial applications:
  - hardware systems
  - communication protocols
  - coordination protocols for distributed systems
  - ⋮

Historical notes

1976	Keller	transition systems (TS) to model parallel systems
1977	Pnueli	temporal logic to specify parallel systems
1981	Clarke/Emerson Queille/Sifakis	first model checker

Historical notes

1976	Keller	transition systems (TS) to model parallel systems
1977	Pnueli	temporal logic **LTL** to specify parallel systems
1981	Clarke/Emerson Queille/Sifakis	first model checker for **CTL**

Historical notes

1976	Keller	transition systems (TS) to model parallel systems
1977	Pnueli	temporal logic **LTL** to specify parallel systems
1981	Clarke/Emerson Queille/Sifakis	first model checker for **CTL**
1983	Kanellakis/Smolka	model checking for homogeneous TS-based specifications
1985	Lichtenstein/Pnueli	model checking for **LTL**
1986	Vardi/Wolper	model checking for **LTL**

Historical notes

1976	Keller	transition systems
1977	Pnueli	temporal logic **LTL**
1981	Clarke/Emerson	first model checker
	Queille/Sifakis	for **CTL**
⋮	⋮	⋮
1985	Lichtenstein/Pnueli	model checking
1986	Vardi/Wolper	for **LTL**

state explosion problem

state space of industrial systems too large to be handled by naïve implementations of model checking algorithms

1976	Keller	transition systems
1977	Pnueli	temporal logic **LTL**
1981	Clarke/Emerson Queille/Sifakis	first model checker for **CTL**
⋮	⋮	⋮
1985	Lichtenstein/Pnueli	} model checking
1986	Vardi/Wolper	for **LTL**

state explosion problem

ca. since 1990
"advanced techniques"

Historical notes

1976	Keller	transition systems
1977	Pnueli	temporal logic **LTL**
1981	Clarke/Emerson Queille/Sifakis	first model checker for **CTL**
⋮	⋮	⋮
1985	Lichtenstein/Pnueli	model checking for **LTL**
1986	Vardi/Wolper	(model checking for **LTL**)

state explosion problem

ca. since 1990
"advanced techniques"

symbolic model checking with BDDs
partial order reduction
⋮

Historical notes

1976	Keller	transition systems
1977	Pnueli	temporal logic **LTL**
1981	Clarke/Emerson Queille/Sifakis	first model checker for **CTL**
⋮	⋮	⋮
1985	Lichtenstein/Pnueli	model checking for **LTL**
1986	Vardi/Wolper	

state explosion problem

ca. since 1990
"advanced techniques"

symbolic model checking with BDDs
partial order reduction
⋮

model checking for infinite systems, quantitative analysis, e.g., real-time systems, probabilistic systems

Transition system (TS)

A transition system is a tuple

$$\mathcal{T} = (S, Act, \longrightarrow, S_0, AP, L)$$

Transition system (TS)

A transition system is a tuple

$$\mathcal{T} = (S, Act, \longrightarrow, S_0, AP, L)$$

- S is the state space, i.e., set of states,

Transition system (TS)

A transition system is a tuple

$$\mathcal{T} = (S, Act, \longrightarrow, S_0, AP, L)$$

- S is the state space, i.e., set of states,
- Act is a set of actions,

Transition system (TS)

A transition system is a tuple

$$\mathcal{T} = (S, Act, \longrightarrow, S_0, AP, L)$$

- S is the state space, i.e., set of states,
- Act is a set of actions,
- $\longrightarrow \subseteq S \times Act \times S$ is the transition relation,

Transition system (TS)

A transition system is a tuple

$$\mathcal{T} = (S, Act, \longrightarrow, S_0, AP, L)$$

- S is the state space, i.e., set of states,
- Act is a set of actions,
- $\longrightarrow \subseteq S \times Act \times S$ is the transition relation,

i.e., transitions have the form $s \xrightarrow{\alpha} s'$ where $s, s' \in S$ and $\alpha \in Act$

Transition system (TS)

A transition system is a tuple

$$\mathcal{T} = (S, Act, \longrightarrow, S_0, AP, L)$$

- S is the state space, i.e., set of states,
- Act is a set of actions,
- $\longrightarrow \subseteq S \times Act \times S$ is the transition relation,

  i.e., transitions have the form $s \xrightarrow{\alpha} s'$ where $s, s' \in S$ and $\alpha \in Act$

- $S_0 \subseteq S$ the set of initial states,

Transition system (TS)

A transition system is a tuple

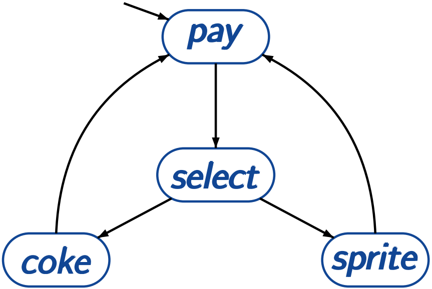
$$\mathcal{T} = (S, Act, \longrightarrow, S_0, AP, L)$$

- S is the state space, i.e., set of states,
- Act is a set of actions,
- $\longrightarrow \subseteq S \times Act \times S$ is the transition relation,

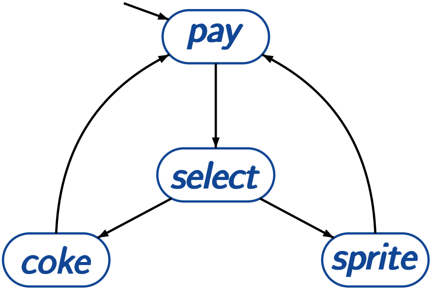
i.e., transitions have the form $s \xrightarrow{\alpha} s'$ where $s, s' \in S$ and $\alpha \in Act$

- $S_0 \subseteq S$ the set of initial states,
- AP a set of atomic propositions,
- $L : S \rightarrow 2^{AP}$ the labeling function

Transition system for beverage machine



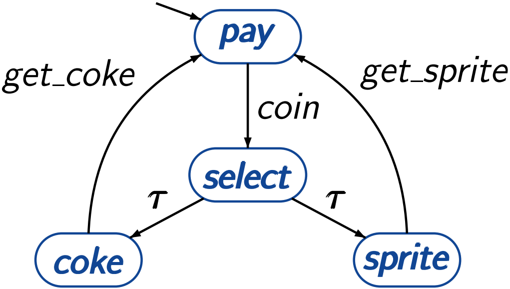
Transition system for beverage machine



state space $S = \{pay, select, coke, sprite\}$

set of initial states: $S_0 = \{pay\}$

Transition system for beverage machine



actions:

coin

τ

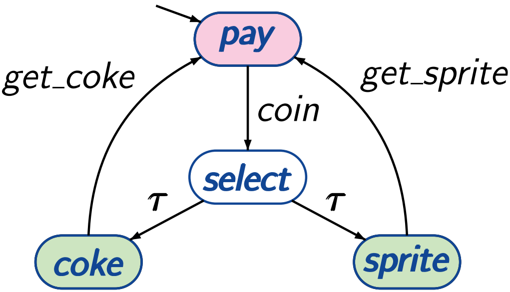
get_sprite

get_coke

state space $S = \{pay, select, coke, sprite\}$

set of initial states: $S_0 = \{pay\}$

Transition system for beverage machine



actions:
coin
τ
get_sprite
get_coke

state space $S = \{pay, select, coke, sprite\}$

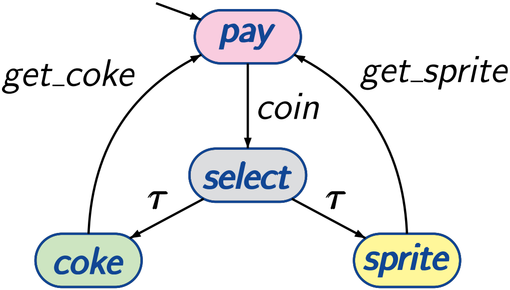
set of initial states: $S_0 = \{pay\}$

set of atomic propositions: $AP = \{pay, drink\}$

labeling function: $L(coke) = L(sprite) = \{drink\}$

$L(pay) = \{pay\}, \; L(select) = \emptyset$

Transition system for beverage machine



actions:
coin
τ
get_sprite
get_coke

state space $S = \{pay, select, coke, sprite\}$

set of initial states: $S_0 = \{pay\}$

set of atomic propositions: $AP = S$

labeling function: $L(s) = \{s\}$ for each state s

"Behaviour" of transition systems

possible behaviours of a TS result from:

```
select nondeterministically an initial state s ∈ S0
WHILE  s is non-terminal DO
      select nondeterministically a transition s --α--> s'
      execute the action α and put s := s'
OD
```

"Behaviour" of transition systems

possible behaviours of a TS result from:

```
select nondeterministically an initial state s ∈ S_0
WHILE  s is non-terminal DO
       select nondeterministically a transition s --α--> s'
       execute the action α and put s := s'
OD
```

executions: maximal "transition sequences"

$$s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} s_2 \xrightarrow{\alpha_3} \ldots \text{ with } s_0 \in S_0$$

"Behaviour" of transition systems

possible behaviours of a TS result from:

```
select nondeterministically an initial state s ∈ S_0
WHILE  s is non-terminal DO
       select nondeterministically a transition s --α--> s'
       execute the action α and put s := s'
OD
```

executions: maximal "transition sequences"

$$s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} s_2 \xrightarrow{\alpha_3} \ldots \text{ with } s_0 \in S_0$$

reachable fragment:

$Reach(\mathcal{T})$ = set of all states that are reachable from an initial state through some execution

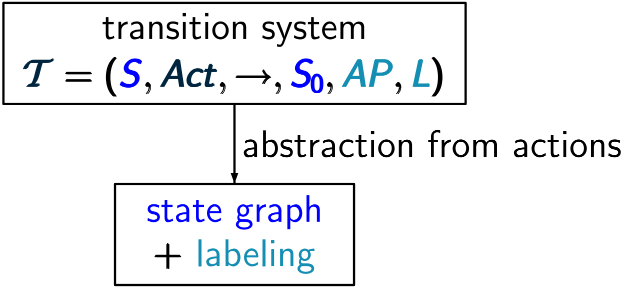
Linear-time vs branching-time

Linear-time vs branching-time

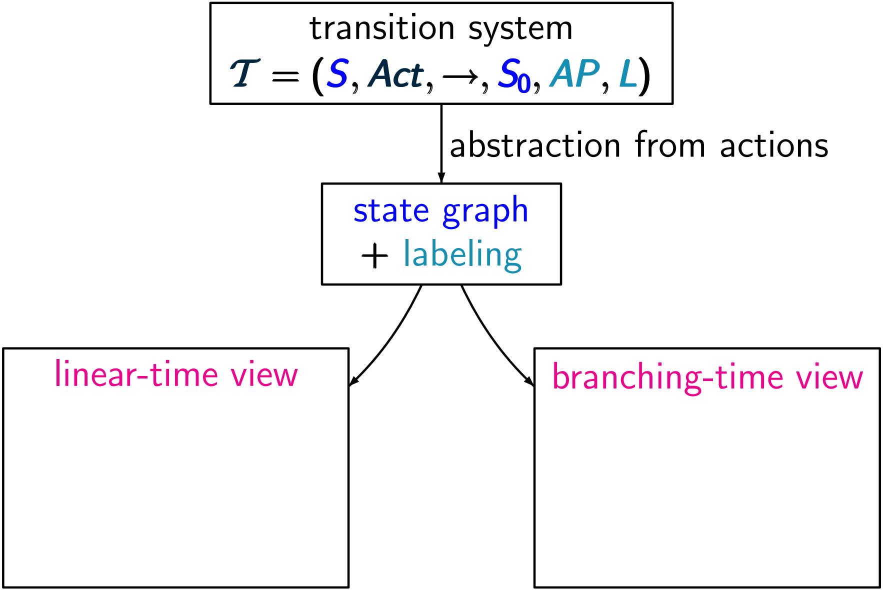
transition system

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

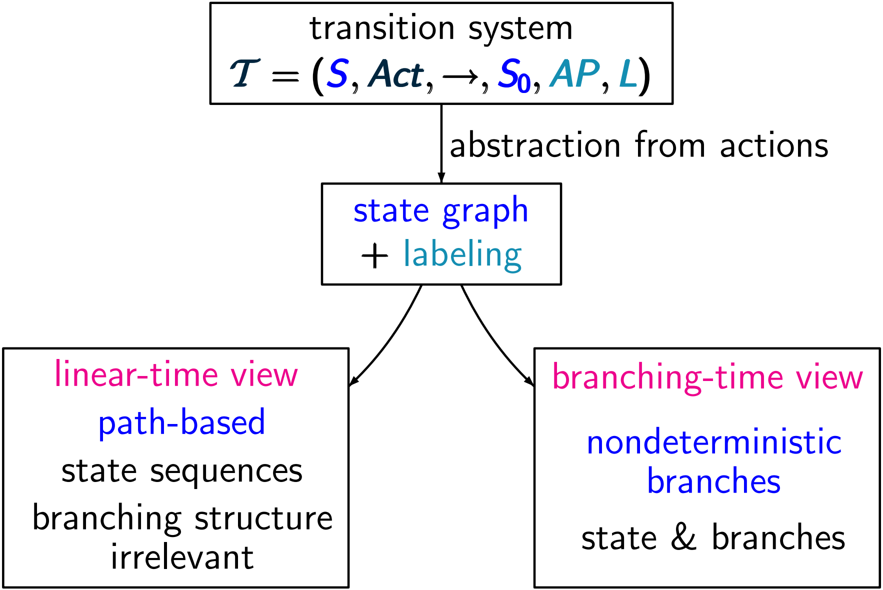
Linear-time vs branching-time



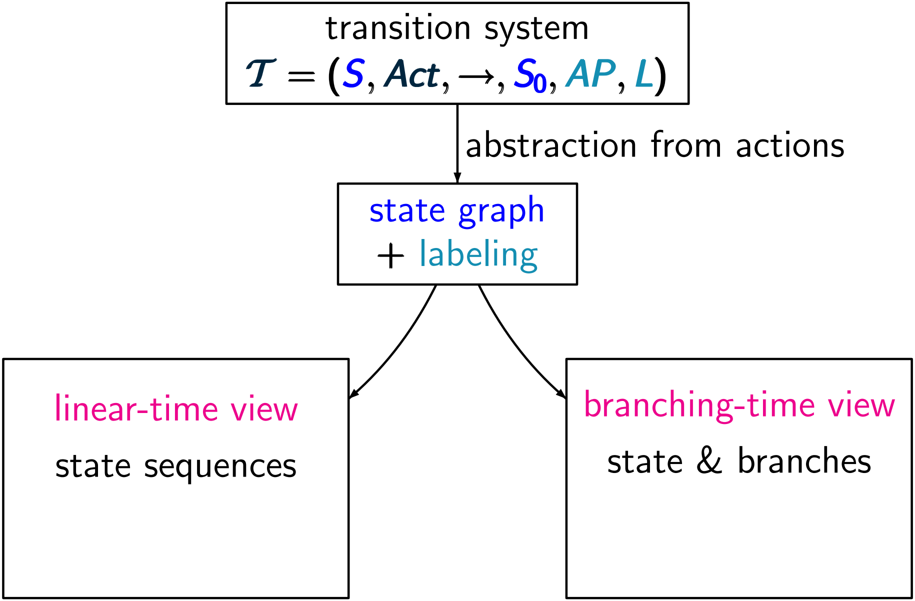
Linear-time vs branching-time



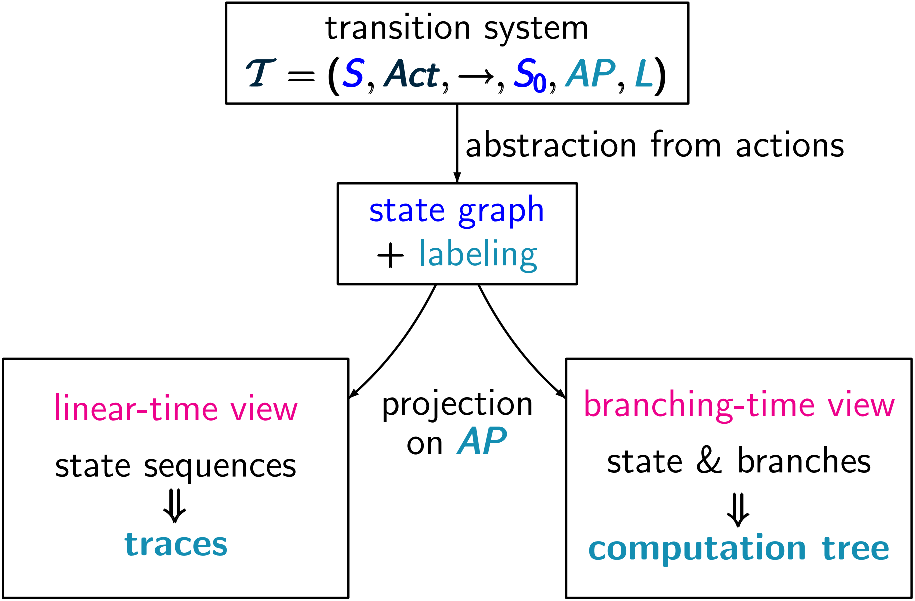
Linear-time vs branching-time



Linear-time vs branching-time



Linear-time vs branching-time



Traces

LTB2.4-4

for TS with labeling function $L : S \to 2^{AP}$

execution: states + actions

$s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} s_2 \xrightarrow{\alpha_3} \ldots$ infinite or finite

↓

paths: sequences of states

$s_0\, s_1\, s_2 \ldots$ infinite or $s_0\, s_1 \ldots s_n$ finite

Traces

for TS with labeling function $L : S \to 2^{AP}$

execution: states + actions

$s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} s_2 \xrightarrow{\alpha_3} \ldots$ infinite or finite

↓

paths: sequences of states

$s_0\, s_1\, s_2 \ldots$ infinite or $s_0\, s_1 \ldots s_n$ finite

↓

traces: sequences of sets of atomic propositions

$L(s_0)\, L(s_1)\, L(s_2)\, \ldots$

Traces

for TS with labeling function $L : S \to 2^{AP}$

execution: states + actions

$s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} s_2 \xrightarrow{\alpha_3} \ldots$ infinite or finite

↓

paths: sequences of states

$s_0\, s_1\, s_2 \ldots$ infinite or $s_0\, s_1 \ldots s_n$ finite

↓

traces: sequences of sets of atomic propositions

$L(s_0)\, L(s_1)\, L(s_2)\ \ldots \quad \in (2^{AP})^\omega \cup (2^{AP})^+$

Traces

for TS with labeling function $L : S \to 2^{AP}$

execution: states + actions

$s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} s_2 \xrightarrow{\alpha_3} \ldots$ infinite or finite

↓

paths: sequences of states

$s_0\, s_1\, s_2 \ldots$ infinite or $s_0\, s_1 \ldots s_n$ finite

↓

traces: sequences of sets of atomic propositions

$L(s_0)\, L(s_1)\, L(s_2)\, \ldots \quad \in (2^{AP})^\omega \cup (2^{AP})^+$

for simplicity: we often assume that the given TS has **no terminal states**

Traces

for TS with labeling function $L : S \to 2^{AP}$

execution: states + actions

$s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} s_2 \xrightarrow{\alpha_3} \ldots$ infinite or ~~finite~~

↓

paths: sequences of states

$s_0\, s_1\, s_2 \ldots$ infinite or ~~$s_0\, s_1 \ldots s_n$ finite~~

↓

traces: sequences of sets of atomic propositions

$L(s_0)\, L(s_1)\, L(s_2)\, \ldots \in (2^{AP})^\omega \cup$ ~~$(2^{AP})^+$~~

for simplicity: we often assume that the given TS has **no terminal states**

Traces of a transition system

Let $\mathcal{T}$ be a TS

$$Traces(\mathcal{T}) \stackrel{\text{def}}{=} \{trace(\pi) : \pi \in Paths(\mathcal{T})\}$$

$$Traces_{fin}(\mathcal{T}) \stackrel{\text{def}}{=} \{trace(\widehat{\pi}) : \widehat{\pi} \in Paths_{fin}(\mathcal{T})\}$$

Traces of a transition system

Let $\mathcal{T}$ be a TS

$$\mathit{Traces}(\mathcal{T}) \stackrel{\text{def}}{=} \{\mathit{trace}(\pi) : \pi \in \mathit{Paths}(\mathcal{T})\}$$

↑ initial, maximal path fragment

$$\mathit{Traces}_{\mathit{fin}}(\mathcal{T}) \stackrel{\text{def}}{=} \{\mathit{trace}(\widehat{\pi}) : \widehat{\pi} \in \mathit{Paths}_{\mathit{fin}}(\mathcal{T})\}$$

↑ initial, finite path fragment

Traces of a transition system

Let $\mathcal{T}$ be a TS ⟵ *without* terminal states

$$Traces(\mathcal{T}) \stackrel{\text{def}}{=} \{trace(\pi) : \pi \in Paths(\mathcal{T})\}$$

↑ initial, infinite path fragment

$$Traces_{fin}(\mathcal{T}) \stackrel{\text{def}}{=} \{trace(\widehat{\pi}) : \widehat{\pi} \in Paths_{fin}(\mathcal{T})\}$$

↑ initial, finite path fragment

Traces of a transition system

Let $\mathcal{T}$ be a TS ⟵ *without* terminal states

$$Traces(\mathcal{T}) \stackrel{\text{def}}{=} \{trace(\pi) : \pi \in Paths(\mathcal{T})\} \subseteq (2^{AP})^{\omega}$$

↑ initial, infinite path fragment

$$Traces_{fin}(\mathcal{T}) \stackrel{\text{def}}{=} \{trace(\widehat{\pi}) : \widehat{\pi} \in Paths_{fin}(\mathcal{T})\} \subseteq (2^{AP})^{*}$$

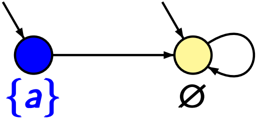
↑ initial, finite path fragment

Example: traces

Let $\mathcal{T}$ be a TS without terminal states.

$$Traces(\mathcal{T}) \stackrel{\text{def}}{=} \{trace(\pi) : \pi \in Paths(\mathcal{T})\} \subseteq (2^{AP})^{\omega}$$

$$Traces_{fin}(\mathcal{T}) \stackrel{\text{def}}{=} \{trace(\widehat{\pi}) : \widehat{\pi} \in Paths_{fin}(\mathcal{T})\} \subseteq (2^{AP})^{*}$$



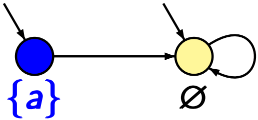
TS $\mathcal{T}$ with a single atomic proposition a

Example: traces

Let $\mathcal{T}$ be a TS without terminal states.

$$Traces(\mathcal{T}) \stackrel{\text{def}}{=} \{trace(\pi) : \pi \in Paths(\mathcal{T})\} \subseteq (2^{AP})^{\omega}$$

$$Traces_{fin}(\mathcal{T}) \stackrel{\text{def}}{=} \{trace(\widehat{\pi}) : \widehat{\pi} \in Paths_{fin}(\mathcal{T})\} \subseteq (2^{AP})^{*}$$

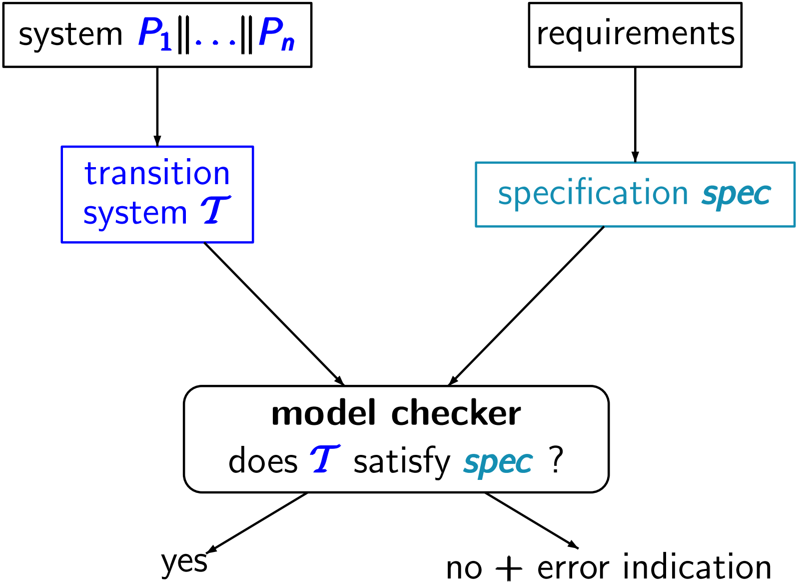


TS $\mathcal{T}$ with a single atomic proposition a

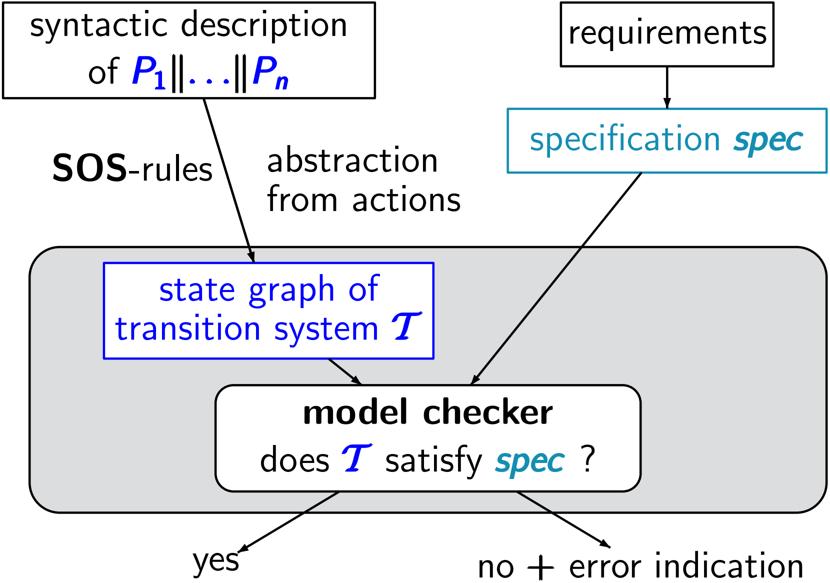
$$Traces(\mathcal{T}) = \{\, \{a\}\varnothing^{\omega}, \varnothing^{\omega} \,\}$$

$$Traces_{fin}(\mathcal{T}) = \{\{a\}\varnothing^{n} : n \geq 0\} \cup \{\varnothing^{m} : m \geq 1\}$$

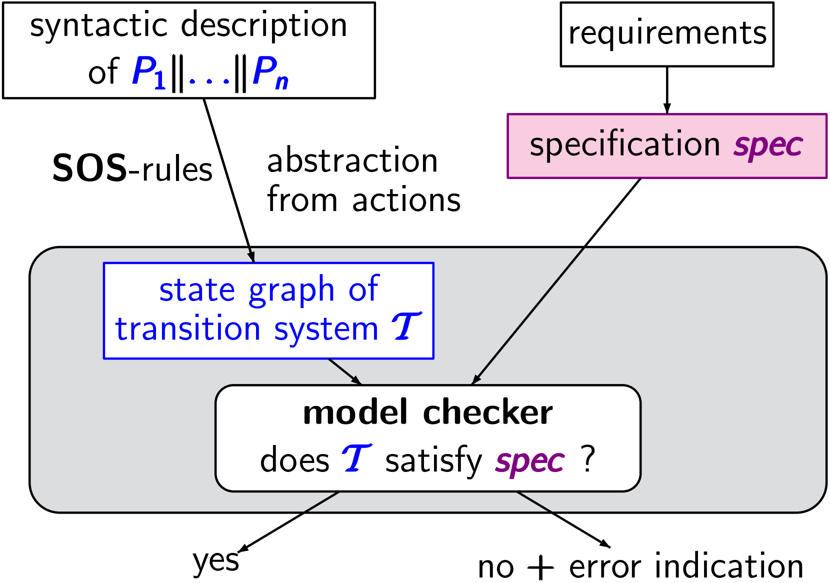
Model checking



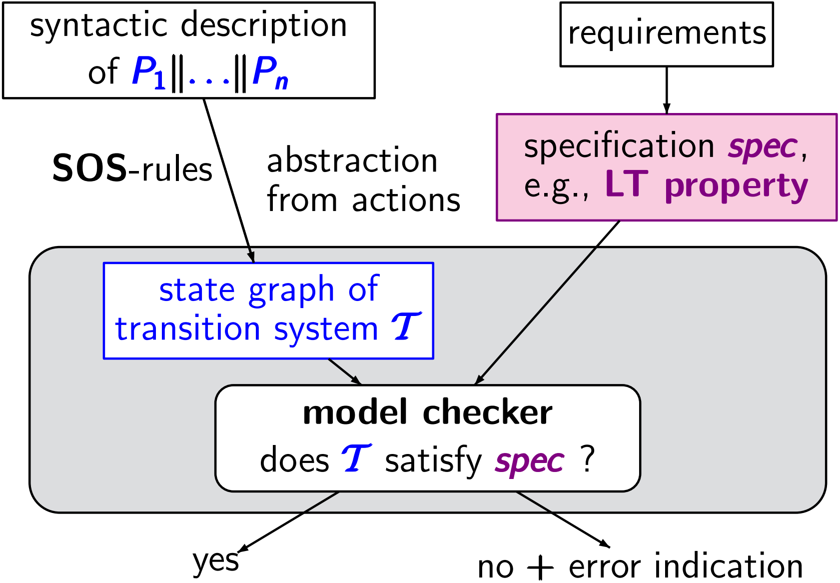
Model checking



Model checking



Model checking



Linear-time properties (LT properties)

Linear-time properties (LT properties)

for TS over AP without terminal states

> An LT property over AP is a language E of infinite words over the alphabet $\Sigma = 2^{AP}$,

Linear-time properties (LT properties)

for TS over AP without terminal states

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

Linear-time properties (LT properties)

for TS over AP without terminal states

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

E.g., for mutual exclusion problems and
$AP = \{crit_1, crit_2, \ldots\}$

safety:

$MUTEX =$ set of all infinite words $A_0\, A_1\, A_2 \ldots$ over 2^{AP} such that for all $i \in \mathbb{N}$:
$crit_1 \notin A_i$ or $crit_2 \notin A_i$

Satisfaction relation for LT properties

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., $E \subseteq \left(2^{AP}\right)^{\omega}$.

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., $E \subseteq \left(2^{AP}\right)^{\omega}$.

Satisfaction relation $\models$ for TS:

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

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

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., $E \subseteq \left(2^{AP}\right)^{\omega}$.

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

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

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

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

LT properties and trace inclusion

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

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

LT properties and trace inclusion

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

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:

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

LT properties and trace inclusion

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

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:

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

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

LT properties and trace inclusion

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

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

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

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

(2) for all LT-properties E over AP: whenever $\mathcal{T}_2 \models E$ then $\mathcal{T}_1 \models E$

LT properties and trace inclusion

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

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

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

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

(2) for all LT-properties E over AP: whenever $\mathcal{T}_2 \models E$ then $\mathcal{T}_1 \models E$

$(1) \Longrightarrow (2)$: $\checkmark$

LT properties and trace inclusion

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

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

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

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

(2) for all LT-properties E over AP: whenever $\mathcal{T}_2 \models E$ then $\mathcal{T}_1 \models E$

(2) $\Longrightarrow$ (1): consider $E = Traces(\mathcal{T}_2)$

Trace equivalence

LTB2.4-21A

Trace equivalence

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

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

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 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**

LT 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 $\mathcal{T}_2 \models E$

Linear Temporal Logic (LTL)

Linear Temporal Logic (LTL)

$$\varphi \ ::= \ \mathit{true} \ \big| \ a \ \big| \ \varphi_1 \wedge \varphi_2 \ \big| \ \neg\varphi$$

where $a \in AP$

Linear Temporal Logic (LTL)

$$\varphi \ ::= \ \textit{true} \ \big| \ a \ \big| \ \varphi_1 \wedge \varphi_2 \ \big| \ \neg\varphi \ \big| \ \bigcirc \varphi$$

where $a \in AP$ $\qquad \bigcirc \mathrel{\widehat{=}} \text{next}$

Linear Temporal Logic (LTL)

$$\varphi \ ::= \ \textit{true} \ \big| \ a \ \big| \ \varphi_1 \wedge \varphi_2 \ \big| \ \neg\varphi \ \big| \ \bigcirc \varphi \ \big| \ \varphi_1 \, \mathsf{U} \, \varphi_2$$

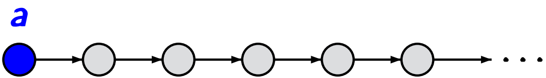
where $a \in AP$ $\qquad$ $\bigcirc \,\widehat{=}\,$ next $\qquad$ $\mathsf{U} \,\widehat{=}\,$ until

Linear Temporal Logic (LTL)

$$\varphi \ ::= \ \mathit{true} \ \mid \ a \ \mid \ \varphi_1 \wedge \varphi_2 \ \mid \ \neg\varphi \ \mid \ \bigcirc \varphi \ \mid \ \varphi_1 \, \mathsf{U} \, \varphi_2$$

where $a \in AP$ $\qquad \bigcirc \mathrel{\widehat{=}}$ next $\qquad \mathsf{U} \mathrel{\widehat{=}}$ until

atomic proposition $a \in AP$

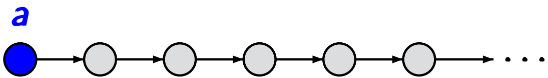


Linear Temporal Logic (LTL)

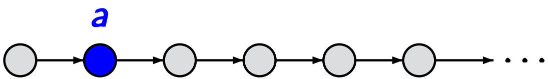
$$\varphi \ ::= \ \mathit{true} \ \big| \ a \ \big| \ \varphi_1 \wedge \varphi_2 \ \big| \ \neg\varphi \ \big| \ \bigcirc \varphi \ \big| \ \varphi_1 \, \mathsf{U} \, \varphi_2$$

where $a \in AP$ $\quad \bigcirc \mathrel{\widehat{=}}$ next $\quad \mathsf{U} \mathrel{\widehat{=}}$ until

atomic proposition $a \in AP$



next operator $\bigcirc a$

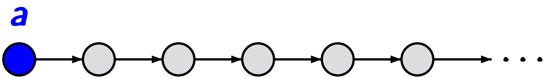


Linear Temporal Logic (LTL)

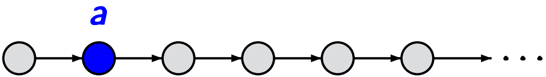
$$\varphi \ ::= \ \mathit{true} \ \big| \ a \ \big| \ \varphi_1 \wedge \varphi_2 \ \big| \ \neg\varphi \ \big| \ \bigcirc \varphi \ \big| \ \varphi_1 \, \mathsf{U} \, \varphi_2$$

where $a \in AP$ $\quad$ $\bigcirc \mathrel{\widehat{=}}$ next $\quad$ $\mathsf{U} \mathrel{\widehat{=}}$ until

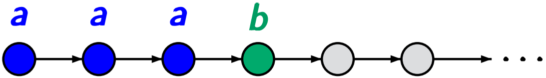
atomic proposition $a \in AP$



next operator $\bigcirc a$



until operator $a \, \mathsf{U} \, b$



Derived operators in LTL

$$\varphi \ ::= \ \mathit{true} \ \big| \ a \ \big| \ \varphi_1 \wedge \varphi_2 \ \big| \ \neg\varphi \ \big| \ \bigcirc \varphi \ \big| \ \varphi_1 \, \mathsf{U} \, \varphi_2$$

derived operators:

$\vee, \rightarrow, \ldots$ as usual

$$\varphi \ ::= \ \textit{true} \mid a \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid \bigcirc \varphi \mid \varphi_1 \, \mathsf{U} \, \varphi_2$$

derived operators:

$\vee, \rightarrow, \ldots$ as usual

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

Derived operators in LTL

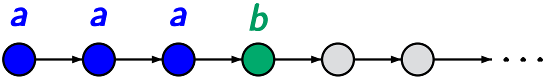
$$\varphi \;::=\; \textit{true} \;\big|\; a \;\big|\; \varphi_1 \wedge \varphi_2 \;\big|\; \neg\varphi \;\big|\; \bigcirc\,\varphi \;\big|\; \varphi_1 \,\mathsf{U}\, \varphi_2$$

derived operators:

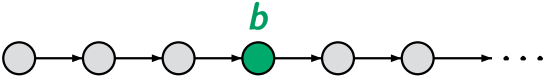
$\vee, \rightarrow, \ldots$ as usual

$\Diamond\varphi \overset{\text{def}}{=} \textit{true}\,\mathsf{U}\,\varphi$ eventually

until operator
$a\,\mathsf{U}\,b$



eventually
$\Diamond b$



Derived operators in LTL

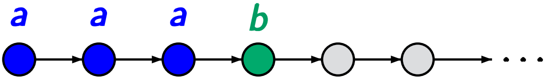
$$\varphi \ ::= \ \mathit{true} \ \big| \ a \ \big| \ \varphi_1 \wedge \varphi_2 \ \big| \ \neg\varphi \ \big| \ \bigcirc \varphi \ \big| \varphi_1 \, \mathsf{U} \, \varphi_2$$

derived operators:

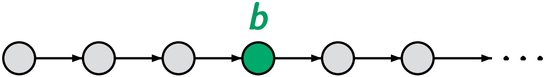
$\vee, \rightarrow, \ldots$ as usual

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

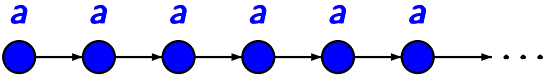
until operator
$a \, \mathsf{U} \, b$



eventually
$\Diamond b$

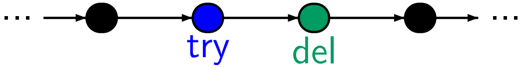


always
$\Box a$



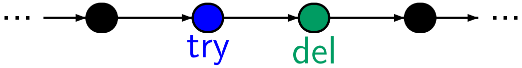
Next $\bigcirc$, until U and eventually $\Diamond$

$\Box\,(\text{try\_to\_send} \rightarrow \bigcirc\, \text{delivered})$

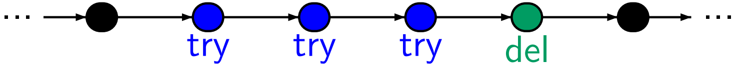


Next $\bigcirc$, until U and eventually $\Diamond$

$\Box\,(\text{try\_to\_send} \longrightarrow \bigcirc\ \text{delivered})$

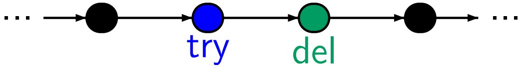


$\Box\,(\text{try\_to\_send} \longrightarrow \text{try\_to\_send}\ \mathbf{U}\ \text{delivered})$

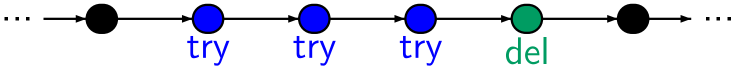


Next $\bigcirc$, until U and eventually $\Diamond$

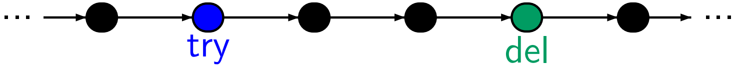
$\square\,(\text{try\_to\_send} \rightarrow \bigcirc\ \text{delivered})$



$\square\,(\text{try\_to\_send} \rightarrow \text{try\_to\_send}\ \mathsf{U}\ \text{delivered})$



$\square\,(\text{try\_to\_send} \rightarrow \Diamond\ \text{delivered})$



Examples for LTL formulas

$$\varphi \;::=\; \textit{true} \;\big|\; a \;\big|\; \varphi_1 \wedge \varphi_2 \;\big|\; \neg\varphi \;\big|\; \bigcirc \varphi \;\big|\; \varphi_1 \,\mathsf{U}\, \varphi_2$$

eventually

$$\Diamond\varphi \stackrel{\text{def}}{=} \textit{true}\,\mathsf{U}\,\varphi$$

always

$$\Box\varphi \stackrel{\text{def}}{=} \neg\Diamond\neg\varphi$$

Examples for LTL formulas

$$\varphi \ ::= \ \mathit{true} \mid a \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid \bigcirc \varphi \mid \varphi_1 \,\mathsf{U}\, \varphi_2$$

eventually

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

always

$$\Box\varphi \stackrel{\text{def}}{=} \neg\Diamond\neg\varphi$$

Examples for LTL formulas:

mutual exclusion: $\Box(\neg \mathit{crit}_1 \vee \neg \mathit{crit}_2)$

Examples for LTL formulas

$$\varphi \ ::= \ \mathit{true} \ \big| \ a \ \big| \ \varphi_1 \wedge \varphi_2 \ \big| \ \neg\varphi \ \big| \ \bigcirc \varphi \ \big| \ \varphi_1 \, \mathsf{U} \, \varphi_2$$

eventually

$$\Diamond\varphi \overset{\text{def}}{=} \mathit{true} \, \mathsf{U} \, \varphi$$

always

$$\Box\varphi \overset{\text{def}}{=} \neg\Diamond\neg\varphi$$

Examples for LTL formulas:

mutual exclusion: $\Box(\neg \mathit{crit}_1 \vee \neg \mathit{crit}_2)$

railroad-crossing: $\Box(\mathit{train\_is\_near} \rightarrow \mathit{gate\_is\_closed})$

$$\varphi ::= \mathit{true} \mid a \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid \bigcirc \varphi \mid \varphi_1 \, \mathsf{U} \, \varphi_2$$

eventually

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

always

$$\Box\varphi \stackrel{\text{def}}{=} \neg\Diamond\neg\varphi$$

Examples for LTL formulas:

mutual exclusion: $\Box(\neg \mathit{crit}_1 \vee \neg \mathit{crit}_2)$

railroad-crossing: $\Box(\mathit{train\_is\_near} \rightarrow \mathit{gate\_is\_closed})$

progress property: $\Box(\mathit{request} \rightarrow \Diamond \mathit{response})$

$$\varphi \ ::= \ \textit{true} \mid a \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid \bigcirc \varphi \mid \varphi_1 \,\mathsf{U}\, \varphi_2$$

eventually

$$\Diamond\varphi \stackrel{\text{def}}{=} \textit{true}\,\mathsf{U}\,\varphi$$

always

$$\Box\varphi \stackrel{\text{def}}{=} \neg\Diamond\neg\varphi$$

Examples for LTL formulas:

mutual exclusion: $\Box(\neg crit_1 \vee \neg crit_2)$

railroad-crossing: $\Box(train\_is\_near \rightarrow gate\_is\_closed)$

progress property: $\Box(request \rightarrow \Diamond response)$

traffic light: $\Box(yellow \vee \bigcirc\neg red)$

$$\varphi \ ::= \ \mathit{true} \mid a \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid \bigcirc \varphi \mid \varphi_1 \, \mathsf{U} \, \varphi_2$$

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

always $\Box\varphi \overset{\text{def}}{=} \neg\Diamond\neg\varphi$

Infinitely often and eventually forever

$$\varphi \;::=\; \textit{true} \;\big|\; a \;\big|\; \varphi_1 \wedge \varphi_2 \;\big|\; \neg\varphi \;\big|\; \bigcirc\varphi \;\big|\; \varphi_1 \,\mathsf{U}\, \varphi_2$$

eventually $\Diamond\varphi \overset{\text{def}}{=} \textit{true}\,\mathsf{U}\,\varphi$

always $\Box\varphi \overset{\text{def}}{=} \neg\Diamond\neg\varphi$

infinitely often $\Box\Diamond\varphi$

Infinitely often and eventually forever

$$\varphi \ ::= \ \mathit{true} \ \big| \ a \ \big| \ \varphi_1 \wedge \varphi_2 \ \big| \ \neg\varphi \ \big| \ \bigcirc \varphi \ \big| \ \varphi_1 \, \mathsf{U} \, \varphi_2$$

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

always $\Box\varphi \stackrel{\text{def}}{=} \neg\Diamond\neg\varphi$

infinitely often $\Box\Diamond\varphi$

e.g., unconditional fairness $\Box\Diamond \mathit{crit}_i$

strong fairness $\Box\Diamond \mathit{wait}_i \longrightarrow \Box\Diamond \mathit{crit}_i$

Infinitely often and eventually forever

$$\varphi \ ::= \ \mathit{true} \ \big| \ a \ \big| \ \varphi_1 \wedge \varphi_2 \ \big| \ \neg\varphi \ \big| \ \bigcirc \varphi \ \big| \ \varphi_1 \,\mathsf{U}\, \varphi_2$$

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

always $\Box\varphi \overset{\text{def}}{=} \neg\Diamond\neg\varphi$

infinitely often $\Box\Diamond\varphi$

eventually forever $\Diamond\Box\varphi$

e.g.,	unconditional fairness	$\Box\Diamond \mathit{crit}_i$
	strong fairness	$\Box\Diamond \mathit{wait}_i \longrightarrow \Box\Diamond \mathit{crit}_i$
	weak fairness	$\Diamond\Box \mathit{wait}_i \longrightarrow \Box\Diamond \mathit{crit}_i$

LTL-semantics

LTL-semantics

interpretation of **LTL** formulas over traces, i.e.,
infinite words over 2^{AP}

LTL-semantics

interpretation of **LTL** formulas over traces, i.e.,
infinite words over 2^{AP}

formalized by a satisfaction relation $\models$ for

- LTL formulas and
- infinite words $\sigma = A_0\, A_1\, A_2\, \ldots \in \left(2^{AP}\right)^{\omega}$

Semantics of LTL over infinite words

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

Semantics of LTL over infinite words

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

$\sigma \models \mathit{true}$

Semantics of LTL over infinite words

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

$\sigma \models \mathit{true}$

$\sigma \models a$ iff $A_0 \models a$,i.e., $a \in A_0$

Semantics of LTL over infinite words

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

$\sigma \models \mathit{true}$

$\sigma \models a$ iff $A_0 \models a$,i.e., $a \in A_0$

$\sigma \models \varphi_1 \wedge \varphi_2$ iff $\sigma \models \varphi_1$ and $\sigma \models \varphi_2$

Semantics of LTL over infinite words

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

$\sigma \models \mathit{true}$

$\sigma \models a$ iff $A_0 \models a$,i.e., $a \in A_0$

$\sigma \models \varphi_1 \wedge \varphi_2$ iff $\sigma \models \varphi_1$ and $\sigma \models \varphi_2$

$\sigma \models \neg\varphi$ iff $\sigma \not\models \varphi$

Semantics of LTL over infinite words

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

$$
\begin{array}{lll}
\sigma \models \mathit{true} & & \\
\sigma \models a & \text{iff} & A_0 \models a \text{ ,i.e., } a \in A_0 \\
\sigma \models \varphi_1 \wedge \varphi_2 & \text{iff} & \sigma \models \varphi_1 \text{ and } \sigma \models \varphi_2 \\
\sigma \models \neg\varphi & \text{iff} & \sigma \not\models \varphi \\
\sigma \models \bigcirc\varphi & \text{iff} & \mathit{suffix}(\sigma, 1) = A_1\, A_2\, A_3 \ldots \models \varphi
\end{array}
$$

Semantics of LTL over infinite words

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

$\sigma \models \mathit{true}$

$\sigma \models a$ iff $A_0 \models a$,i.e., $a \in A_0$

$\sigma \models \varphi_1 \wedge \varphi_2$ iff $\sigma \models \varphi_1$ and $\sigma \models \varphi_2$

$\sigma \models \neg\varphi$ iff $\sigma \not\models \varphi$

$\sigma \models \bigcirc\varphi$ iff $\mathit{suffix}(\sigma, 1) = A_1 \, A_2 \, A_3 \, \ldots \models \varphi$

$\sigma \models \varphi_1 \, \mathsf{U} \, \varphi_2$ iff there exists $j \geq 0$ such that

$\mathit{suffix}(\sigma, j) = A_j \, A_{j+1} \, A_{j+2} \, \ldots \models \varphi_2$ and

$\mathit{suffix}(\sigma, i) = A_i \, A_{i+1} \, A_{i+2} \, \ldots \models \varphi_1$ for $0 \leq i < j$

LTL semantics over TS

LTL semantics over TS

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

define satisfaction relation $\models$ for

- **LTL** formulas over AP
- the maximal path fragments and states of $\mathcal{T}$

LTL semantics over TS

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

define satisfaction relation $\models$ for

- **LTL** formulas over AP
- the maximal path fragments and states of $\mathcal{T}$

assumption: $\mathcal{T}$ has no terminal states, i.e.,
all maximal path fragments in $\mathcal{T}$ are infinite

LTL semantics over paths of TS

LTL semantics over paths of TS

given: TS $\mathcal{T} = (S, Act, \rightarrow, S_0, AP, L)$
without terminal states

LTL formula φ over AP

LTL semantics over paths of TS

given: TS $\mathcal{T} = (S, Act, \to, S_0, AP, L)$
without terminal states

LTL formula φ over AP

interpretation of φ over infinite path fragments

$$\pi = s_0\, s_1\, s_2 \ldots \models \varphi \quad \text{iff} \quad trace(\pi) \models \varphi$$

LTL semantics over paths of TS

given: TS $\mathcal{T} = (S, Act, \rightarrow, S_0, AP, L)$
without terminal states

LTL formula φ over AP

interpretation of φ over infinite path fragments

$$\pi = s_0 \, s_1 \, s_2 \ldots \models \varphi \quad \text{iff} \quad trace(\pi) \models \varphi$$

$$\text{iff} \quad trace(\pi) \in Words(\varphi)$$

LTL semantics over paths of TS

given: TS $\mathcal{T} = (S, Act, \rightarrow, S_0, AP, L)$
without terminal states

LTL formula φ over AP

interpretation of φ over infinite path fragments

$$\pi = s_0\, s_1\, s_2 \ldots \models \varphi \quad \text{iff} \quad trace(\pi) \models \varphi$$
$$\text{iff} \quad trace(\pi) \in Words(\varphi)$$

remind: LT property of an LTL formula:

$$Words(\varphi) = \left\{\sigma \in \left(2^{AP}\right)^{\omega} : \sigma \models \varphi\right\}$$

LTL semantics over the states of a TS

LTLSF3.1-SEM-STATES

LTL semantics over the states of a TS

given: TS $\mathcal{T} = (S, Act, \rightarrow, S_0, AP, L)$
without terminal states

LTL formula φ over AP

interpretation of φ over infinite path fragments

$$\pi = s_0\, s_1\, s_2 \ldots \models \varphi \quad \text{iff} \quad trace(\pi) \models \varphi$$

interpretation of φ over states:

$$s \models \varphi \quad \text{iff} \quad trace(\pi) \models \varphi \text{ for all } \pi \in Paths(s)$$

LTL semantics over the states of a TS

given: TS $\mathcal{T} = (S, Act, \rightarrow, S_0, AP, L)$
without terminal states

LTL formula φ over AP

interpretation of φ over infinite path fragments

$$\pi = s_0\, s_1\, s_2 \ldots \models \varphi \quad \text{iff} \quad trace(\pi) \models \varphi$$

interpretation of φ over states:

$$s \models \varphi \quad \text{iff} \quad trace(\pi) \models \varphi \text{ for all } \pi \in Paths(s)$$
$$\text{iff} \quad s \models Words(\varphi)$$

LTL semantics over the states of a TS

given: TS $\mathcal{T} = (S, Act, \rightarrow, S_0, AP, L)$
without terminal states

LTL formula φ over AP

interpretation of φ over infinite path fragments

$$\pi = s_0\, s_1\, s_2 \ldots \models \varphi \quad \text{iff} \quad trace(\pi) \models \varphi$$

interpretation of φ over states:

$$s \models \varphi \quad \text{iff} \quad trace(\pi) \models \varphi \text{ for all } \pi \in Paths(s)$$

$$\text{iff} \quad s \models Words(\varphi)$$

↑

satisfaction relation for LT properties

LTL semantics over the states of a TS

given: TS $\mathcal{T} = (S, Act, \rightarrow, S_0, AP, L)$
without terminal states

LTL formula φ over AP

interpretation of φ over infinite path fragments

$$\pi = s_0\, s_1\, s_2 \ldots \models \varphi \quad \text{iff} \quad trace(\pi) \models \varphi$$

interpretation of φ over states:

$$\begin{aligned} s \models \varphi \quad & \text{iff} \quad trace(\pi) \models \varphi \text{ for all } \pi \in Paths(s) \\ & \text{iff} \quad s \models Words(\varphi) \\ & \text{iff} \quad Traces(s) \subseteq Words(\varphi) \end{aligned}$$

Interpretation of LTL formulas over TS

Interpretation of LTL formulas over TS

given: TS $\mathcal{T} = (S, Act, \rightarrow, S_0, AP, L)$
without terminal states

LTL formula φ over AP

$\mathcal{T} \models \varphi$ iff $s_0 \models \varphi$ for all $s_0 \in S_0$

Interpretation of LTL formulas over TS

given: TS $\mathcal{T} = (S, Act, \rightarrow, S_0, AP, L)$
without terminal states

LTL formula φ over AP

$\mathcal{T} \models \varphi$ iff $s_0 \models \varphi$ for all $s_0 \in S_0$

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

Interpretation of LTL formulas over TS

given: TS $\mathcal{T} = (S, Act, \rightarrow, S_0, AP, L)$
without terminal states

LTL formula φ over AP

$\mathcal{T} \models \varphi$ iff $s_0 \models \varphi$ for all $s_0 \in S_0$

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

iff $Traces(\mathcal{T}) \subseteq Words(\varphi)$

Interpretation of LTL formulas over TS

given: TS $\mathcal{T} = (S, Act, \rightarrow, S_0, AP, L)$
without terminal states

LTL formula φ over AP

$\mathcal{T} \models \varphi$	iff	$s_0 \models \varphi$ for all $s_0 \in S_0$
	iff	$trace(\pi) \models \varphi$ for all $\pi \in Paths(\mathcal{T})$
	iff	$Traces(\mathcal{T}) \subseteq Words(\varphi)$
	iff	$\mathcal{T} \models Words(\varphi)$

Interpretation of LTL formulas over TS

given: TS $\mathcal{T} = (S, Act, \rightarrow, S_0, AP, L)$
without terminal states

LTL formula φ over AP

$$
\begin{aligned}
\mathcal{T} \models \varphi \quad & \text{iff} \quad s_0 \models \varphi \text{ for all } s_0 \in S_0 \\
& \text{iff} \quad trace(\pi) \models \varphi \text{ for all } \pi \in Paths(\mathcal{T}) \\
& \text{iff} \quad Traces(\mathcal{T}) \subseteq Words(\varphi) \\
& \text{iff} \quad \mathcal{T} \models Words(\varphi)
\end{aligned}
$$

↑ satisfaction relation for LT properties

Linear-time implementation relations

Linear-time implementation relations

finite trace inclusion and equivalence:

e.g., $\mathit{Tracesfin}(\mathcal{T}_1) \subseteq \mathit{Tracesfin}(\mathcal{T}_2)$

trace inclusion and trace equivalence:

e.g., $\mathit{Traces}(\mathcal{T}_1) \subseteq \mathit{Traces}(\mathcal{T}_2)$

Linear-time implementation relations

finite trace inclusion and equivalence:

e.g., $\mathit{Tracesfin}(\mathcal{T}_1) \subseteq \mathit{Tracesfin}(\mathcal{T}_2)$

preserves all linear-time **safety** properties

trace inclusion and trace equivalence:

e.g., $\mathit{Traces}(\mathcal{T}_1) \subseteq \mathit{Traces}(\mathcal{T}_2)$

Linear-time implementation relations

finite trace inclusion and equivalence:

e.g., $\mathit{Tracesfin}(\mathcal{T}_1) \subseteq \mathit{Tracesfin}(\mathcal{T}_2)$

preserves all linear-time **safety** properties

trace inclusion and trace equivalence:

e.g., $\mathit{Traces}(\mathcal{T}_1) \subseteq \mathit{Traces}(\mathcal{T}_2)$

preserves all **LTL** properties

Linear-time implementation relations

finite trace inclusion and equivalence:

e.g., $\mathit{Tracesfin}(\mathcal{T}_1) \subseteq \mathit{Tracesfin}(\mathcal{T}_2)$

preserves all linear-time **safety** properties

trace inclusion and trace equivalence:

e.g., $\mathit{Traces}(\mathcal{T}_1) \subseteq \mathit{Traces}(\mathcal{T}_2)$

preserves all **LTL** properties

* none of the LT relations is compatible with **CTL**

Linear-time implementation relations

finite trace inclusion and equivalence:

e.g., $\mathit{Tracesfin}(\mathcal{T}_1) \subseteq \mathit{Tracesfin}(\mathcal{T}_2)$

preserves all linear-time **safety** properties

trace inclusion and trace equivalence:

e.g., $\mathit{Traces}(\mathcal{T}_1) \subseteq \mathit{Traces}(\mathcal{T}_2)$

preserves all **LTL** properties

- ∗ none of the LT relations is compatible with **CTL**
- ∗ checking LT relations is computationally hard

Linear-time implementation relations

finite trace inclusion and equivalence:

e.g., $Tracesfin(\mathcal{T}_1) \subseteq Tracesfin(\mathcal{T}_2)$

preserves all linear-time **safety** properties

trace inclusion and trace equivalence:

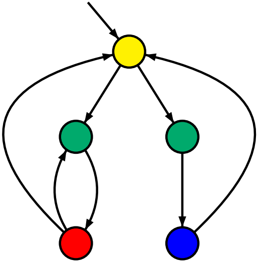
e.g., $Traces(\mathcal{T}_1) \subseteq Traces(\mathcal{T}_2)$

preserves all **LTL** properties

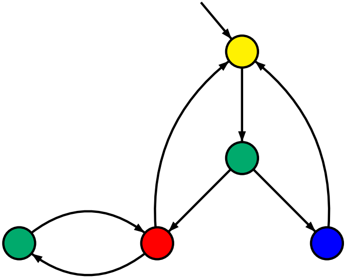
* none of the LT relations is compatible with **CTL**
* checking LT relations is computationally hard
* **minimization ???**

Minimization w.r.t. trace equivalence?

$\mathcal{T}_1$:

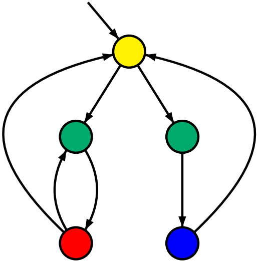


$\mathcal{T}_2$:

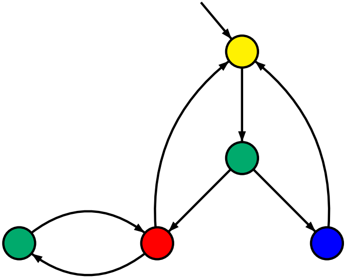


Minimization w.r.t. trace equivalence?

$\mathcal{T}_1$:



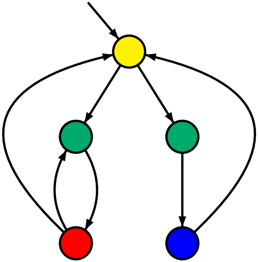
$\mathcal{T}_2$:



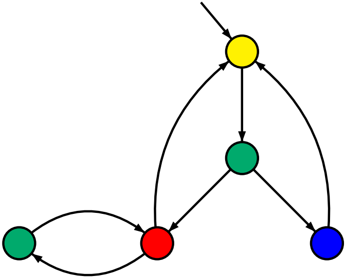
- $\mathit{Traces}(\mathcal{T}_1) = \mathit{Traces}(\mathcal{T}_2)$

Minimization w.r.t. trace equivalence?

$\mathcal{T}_1$:



$\mathcal{T}_2$:

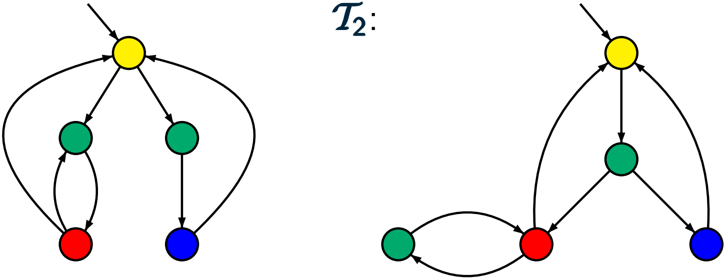


- $\mathit{Traces}(\mathcal{T}_1) = \mathit{Traces}(\mathcal{T}_2)$

  but $\mathcal{T}_1$ and $\mathcal{T}_2$ are not isomorphic

Minimization w.r.t. trace equivalence?

$\mathcal{T}_1$: $\mathcal{T}_2$:



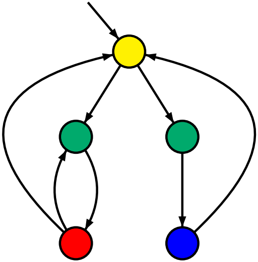
- $\textit{Traces}(\mathcal{T}_1) = \textit{Traces}(\mathcal{T}_2)$

  but $\mathcal{T}_1$ and $\mathcal{T}_2$ are not isomorphic

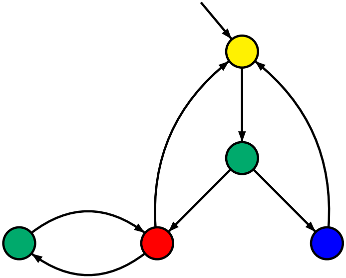
- $\mathcal{T}_1$, $\mathcal{T}_2$ have **5** states and **7** transitions each

Minimization w.r.t. trace equivalence?

$\mathcal{T}_1$:



$\mathcal{T}_2$:



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

  but $\mathcal{T}_1$ and $\mathcal{T}_2$ are not isomorphic
- $\mathcal{T}_1$, $\mathcal{T}_2$ have **5** states and **7** transitions each
- there is no smaller TS that is trace-equivalent to $\mathcal{T}_i$

Classification of implementation relations BSEQOR5.1-6

Classification of implementation relations

- linear vs. branching time
  - linear time: trace relations
  - branching time: (bi)simulation relations

Classification of implementation relations

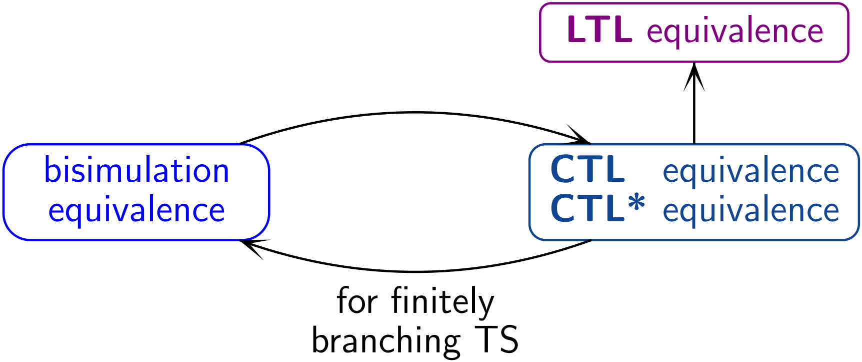
- linear vs. branching time
  * linear time: trace relations
  * branching time: (bi)simulation relations

- (nonsymmetric) preorders vs. equivalences:
  * preorders: trace inclusion, simulation
  * equivalences: trace equivalence, bisimulation

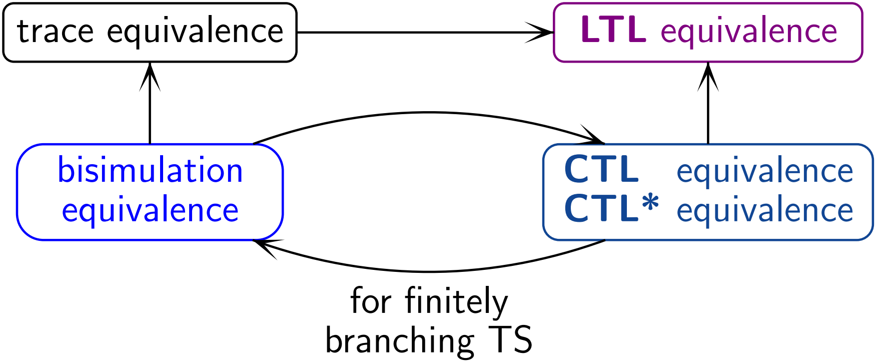
Classification of implementation relations

- linear vs. branching time
  - linear time: trace relations
  - branching time: (bi)simulation relations
- (nonsymmetric) preorders vs. equivalences:
  - preorders: trace inclusion, simulation
  - equivalences: trace equivalence, bisimulation
- strong vs. weak relations
  - strong: reasoning about all transitions
  - weak: abstraction from stutter steps

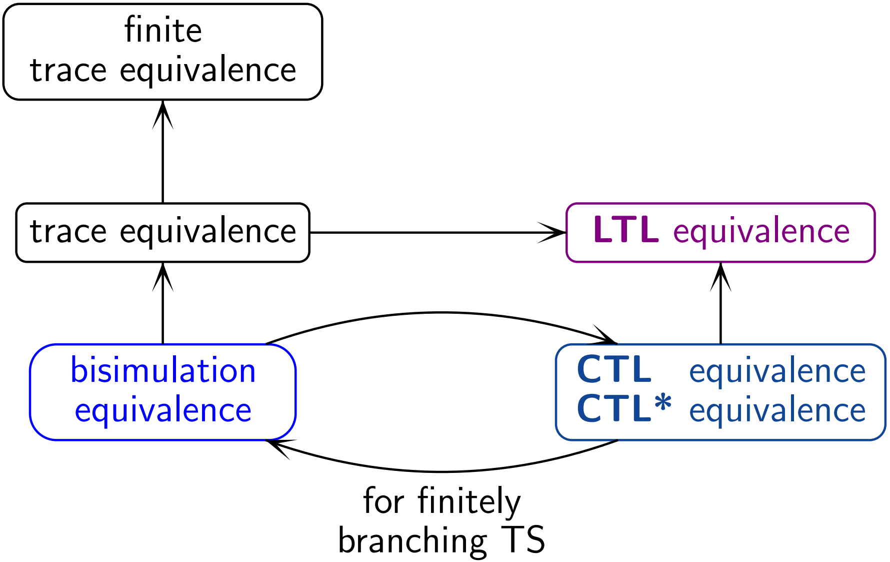
Summary: equivalences



Summary: equivalences



Summary: equivalences



Summary: equivalences