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

State-based view of TS

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

↓ abstraction from actions

state graph $G_{\mathcal{T}}$

- set of nodes = state space S
- edges = transitions without action label

Act for modeling interactions/communication and specifying fairness assumptions

AP, L for specifying properties

State-based view of TS

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

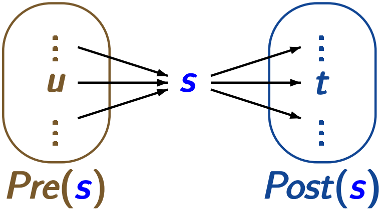
abstraction from actions

state graph $G_{\mathcal{T}}$

- set of nodes = state space S
- edges = transitions without action label

use standard notations for graphs, e.g.,

$$Post(s) = \{t \in S : s \to t\}$$
$$Pre(s) = \{u \in S : u \to s\}$$



Execution and path fragments

execution fragment: sequence of consecutive transitions

$s_0 \xrightarrow{\alpha_0} s_1 \xrightarrow{\alpha_1} \ldots$ infinite or

$s_0 \xrightarrow{\alpha_0} s_1 \xrightarrow{\alpha_1} \ldots \xrightarrow{\alpha_{n-1}} s_n$ finite

path fragment: sequence of states arising from the projection of an execution fragment to the states

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

such that $s_{i+1} \in Post(s_i)$ for all $i < |\pi|$

initial: if $s_0 \in S_0$ = set of initial states

maximal: if infinite or ending in a terminal state

Notations for paths

path fragment: sequence of states

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

s.t. $s_{i+1} \in Post(s_i)$ for all $i < |\pi|$

initial: if $s_0 \in S_0$ = set of initial states

maximal: if infinite or ending in terminal state

path of TS $\mathcal{T}$ $\widehat{=}$ initial, maximal path fragment

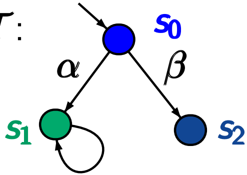
path of state s $\widehat{=}$ maximal path fragment starting in state s

$Paths(\mathcal{T})$ = set of all initial, maximal path fragments

$Paths(s)$ = set of all maximal path fragments starting in state s

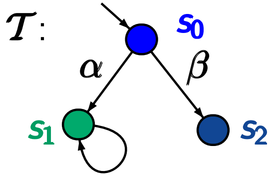
Paths of a TS

$\mathcal{T}$:



How many paths are there in $\mathcal{T}$?

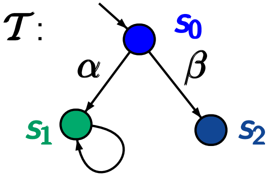
Paths of a TS



How many paths are there in $\mathcal{T}$?

answer: 2, namely $s_0\, s_1\, s_1\, s_1 \ldots$ and $s_0\, s_2$

Paths of a TS and its states



How many paths are there in $\mathcal{T}$?

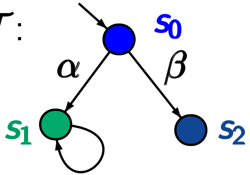
answer: 2, namely $s_0\, s_1\, s_1\, s_1 \ldots$ and $s_0\, s_2$

$Paths(s_1)$ = set of all maximal paths fragments starting in s_1

$= \{s_1^\omega\}$ where $s_1^\omega = s_1\, s_1\, s_1\, s_1\, \ldots$

Paths of a TS and its states

$\mathcal{T}$:



How many paths are there in $\mathcal{T}$?

answer: 2, namely $s_0\, s_1\, s_1\, s_1 \ldots$ and $s_0\, s_2$

$Paths(s_1)$ = set of all maximal paths fragments starting in s_1

$= \{s_1^\omega\}$ where $s_1^\omega = s_1\, s_1\, s_1\, s_1\, \ldots$

$Paths_{fin}(s_1)$ = set of all finite path fragments starting in s_1

$= \{s_1^n : n \in \mathbb{N}, n \geq 1\}$

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

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

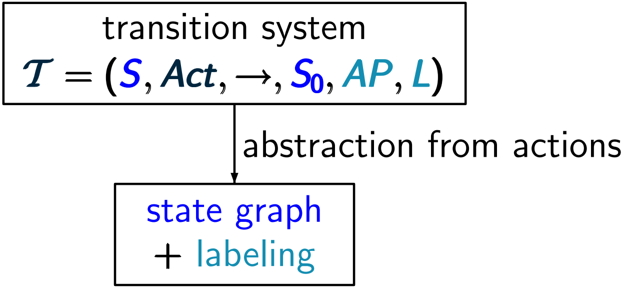
Linear-time vs branching-time

Linear-time vs branching-time

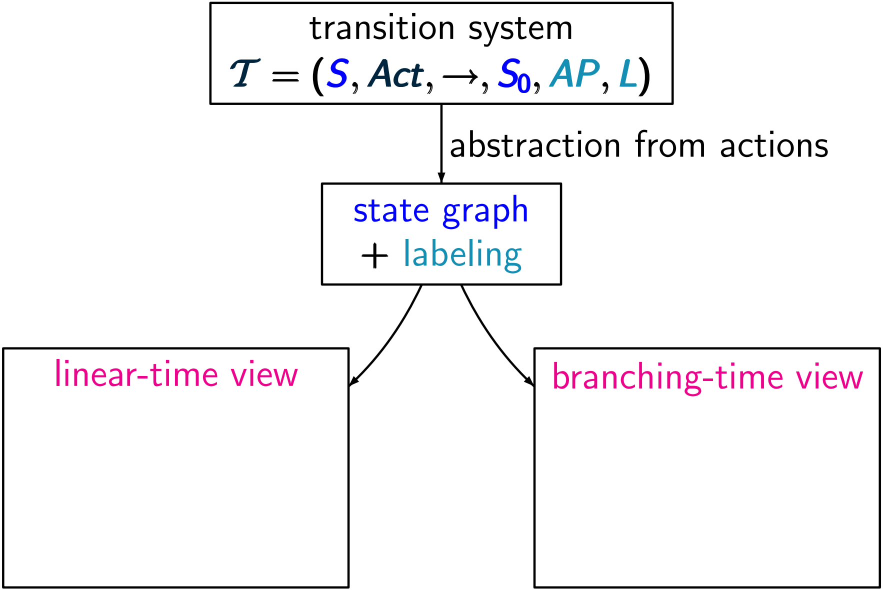
transition system

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

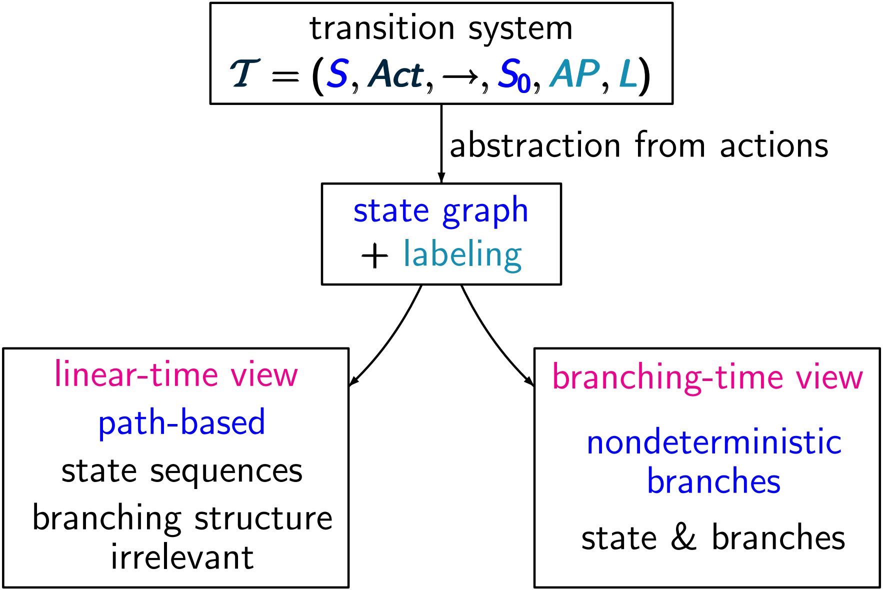
Linear-time vs branching-time



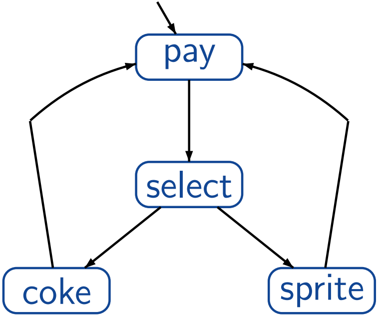
Linear-time vs branching-time



Linear-time vs branching-time

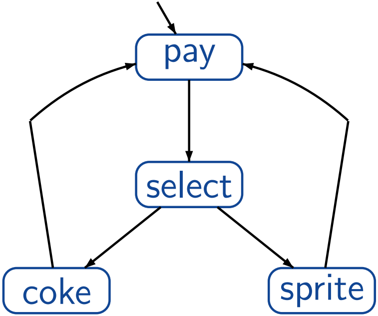


Example: vending machine



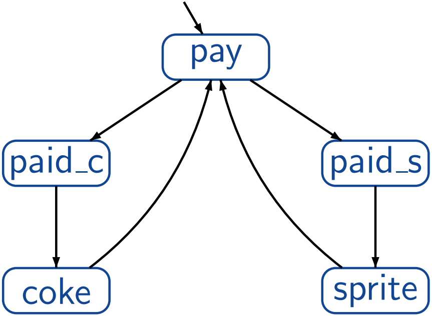
vending machine with
1 coin deposit
select drink after
having paid

Example: vending machine



vending machine with
1 coin deposit

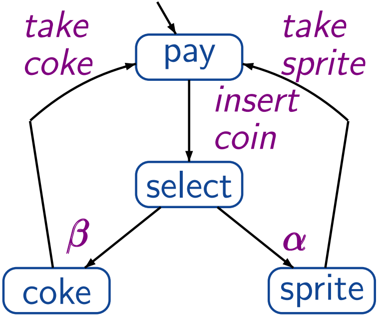
select drink after
having paid

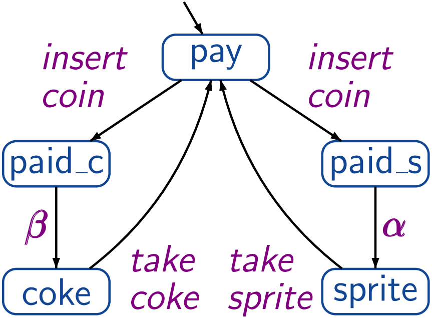


vending machine with
2 coin deposits

select drink by inserting
the coin

Example: vending machine

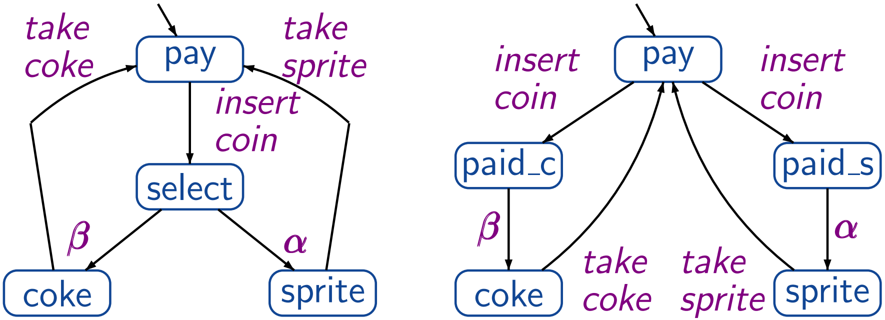




vending machine with
1 coin deposit
select drink after
having paid

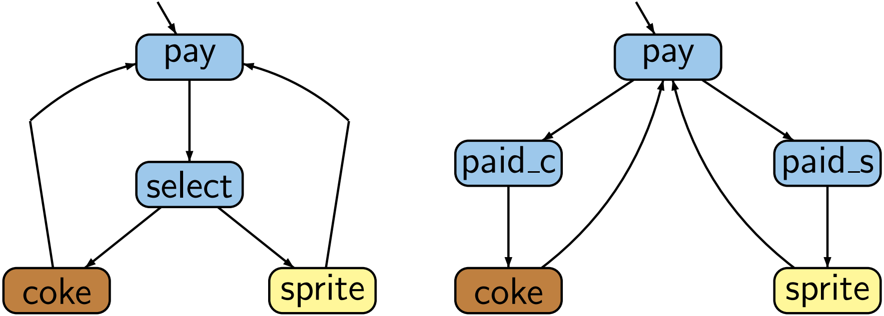
vending machine with
2 coin deposits
select drink by inserting
the coin

Example: vending machine



state based view: abstracts from actions and projects onto atomic propositions, e.g. $AP = \{coke, sprite\}$

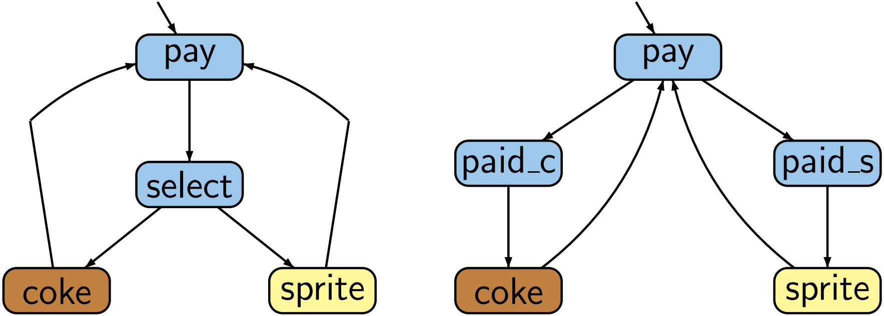
Example: vending machine



state based view: abstracts from actions and projects onto atomic propositions, e.g. $AP = \{coke, sprite\}$

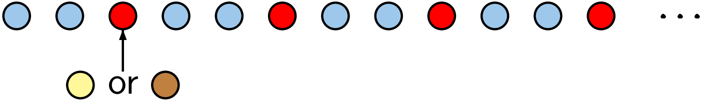
e.g., $L(\text{coke}) = \{coke\}$, $L(\text{pay}) = \varnothing$

Example: vending machine

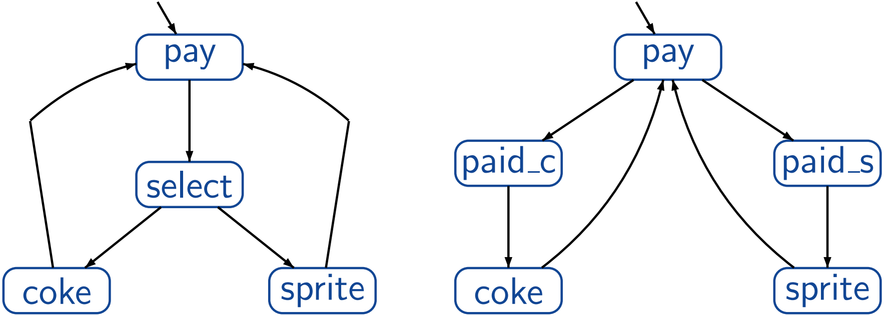


state based view: abstracts from actions and projects onto atomic propositions, e.g. ***AP*** = {***coke***, ***sprite***}

linear time: all observable behaviors are of the form

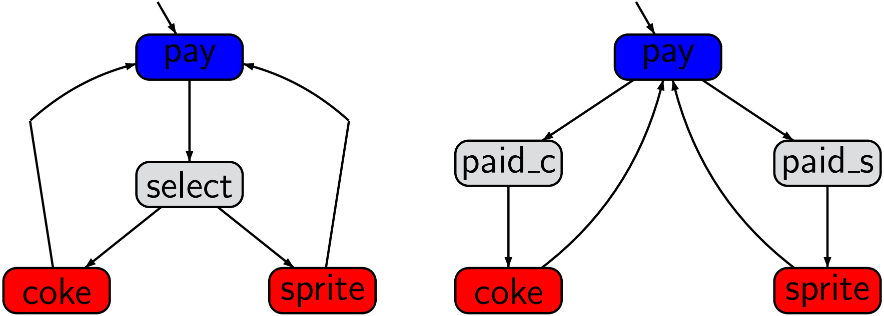


Example: vending machine



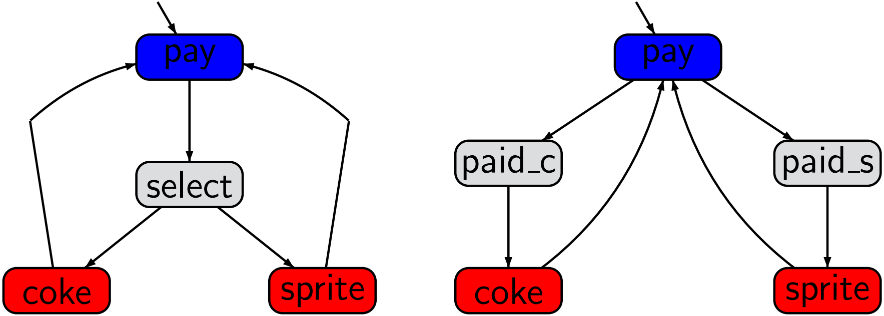
state based view: abstracts from actions and projects on atomc propositions, e.g., $AP = \{pay, drink\}$

Example: vending machine



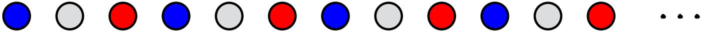
state based view: abstracts from actions and projects on atomc propositions, e.g., $AP = \{pay, drink\}$

Example: vending machine

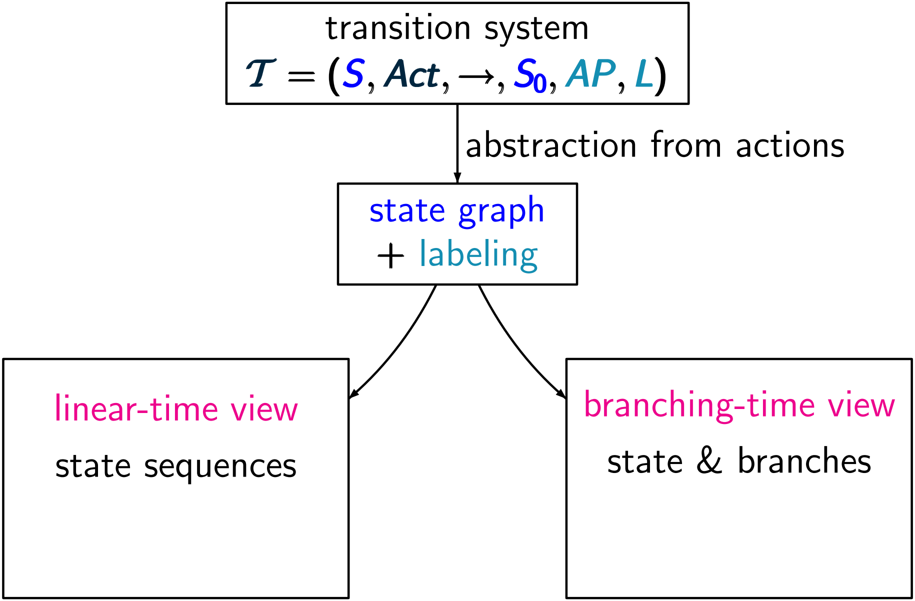


state based view: abstracts from actions and projects
on atomc propositions, e.g., $AP = \{pay, drink\}$

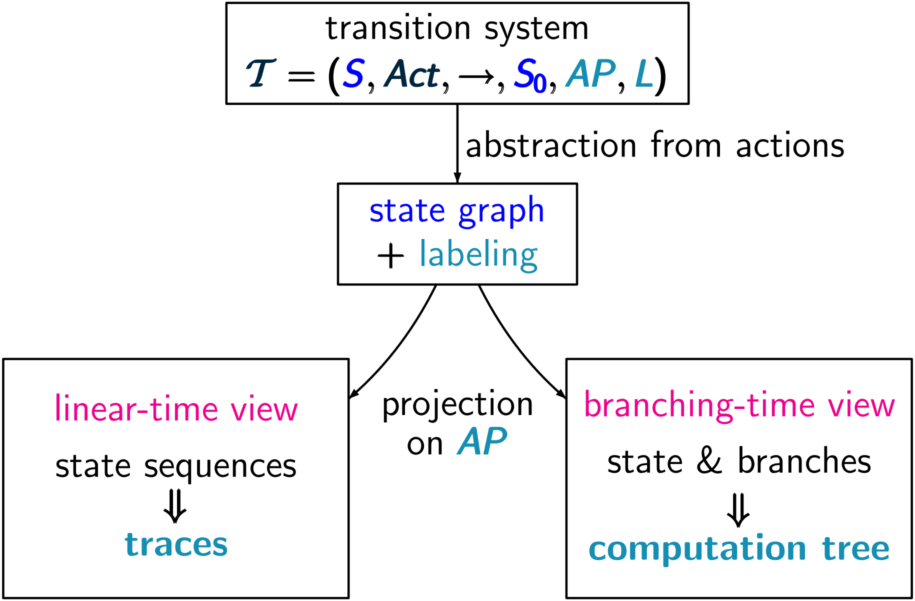
linear & branching time:
all observable behaviors have the form



Linear-time vs branching-time



Linear-time vs branching-time



Traces

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

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**

Treatment of terminal states

perform standard graph algorithms to compute the reachable fragment of the given TS

$$\mathit{Reach}(\mathcal{T}) = \left\{ \begin{array}{c} \text{set of states that are reachable} \\ \text{from some initial state} \end{array} \right.$$

Treatment of terminal states

perform standard graph algorithms to compute the reachable fragment of the given TS

$$\mathit{Reach}(\mathcal{T}) = \left\{ \begin{array}{c} \text{set of states that are reachable} \\ \text{from some initial state} \end{array} \right.$$

for each reachable terminal state s:

- if s stands for an intended halting configuration then add a transition from s to a trap state:

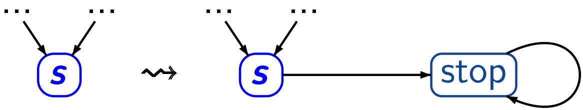
Treatment of terminal states

perform standard graph algorithms to compute the reachable fragment of the given TS

$$\mathit{Reach}(\mathcal{T}) = \left\{ \begin{array}{c} \text{set of states that are reachable} \\ \text{from some initial state} \end{array} \right.$$

for each reachable terminal state s:

- if s stands for an intended halting configuration then add a transition from s to a trap state:



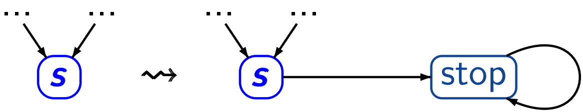
Treatment of terminal states

perform standard graph algorithms to compute the reachable fragment of the given TS

$$Reach(\mathcal{T}) = \left\{ \begin{array}{c} \text{set of states that are reachable} \\ \text{from some initial state} \end{array} \right.$$

for each reachable terminal state s:

- if s stands for an intended halting configuration then add a transition from s to a trap state:



- if s stands for system fault, e.g., deadlock then correct the design before checking further properties

Traces of a transition system

Let $\mathcal{T}$ be a TS

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

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

Traces of a transition system

Let $\mathcal{T}$ be a TS

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

↑ initial, maximal 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 $\longleftarrow$ *without* terminal states

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

π $\uparrow$ initial, infinite path fragment

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

$\widehat{\pi}$ $\uparrow$ 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

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

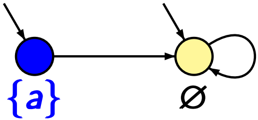
↑ initial, finite path fragment

Example: traces

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

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

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



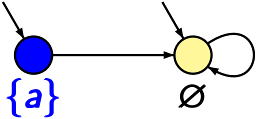
TS $\mathcal{T}$ with a single atomic proposition a

Example: traces

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

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

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

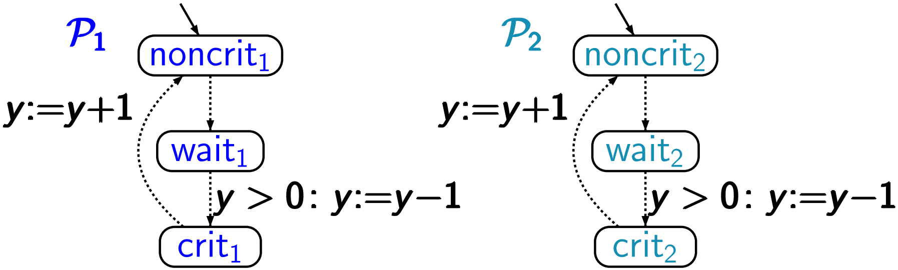


TS $\mathcal{T}$ with a single atomic proposition a

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

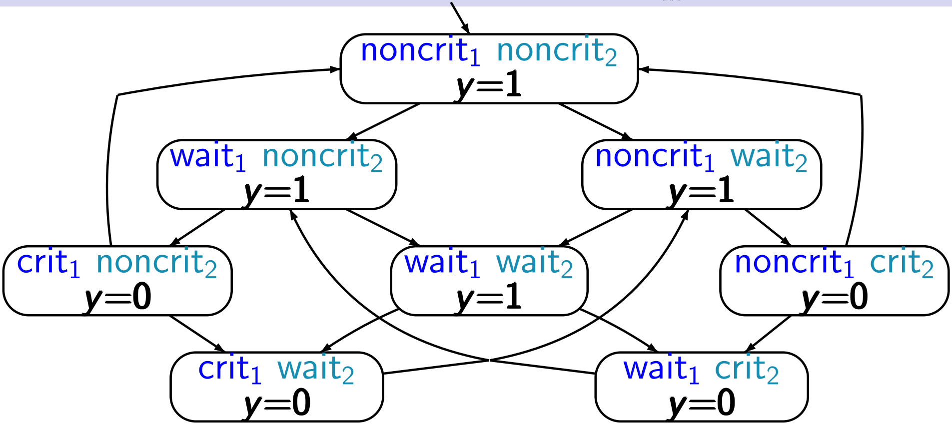
$$Traces_{fin}(\mathcal{T}) = \{\{a\}\varnothing^n : n \geq 0\} \cup \{\varnothing^m : m \geq 1\}$$

Mutual exclusion with semaphore



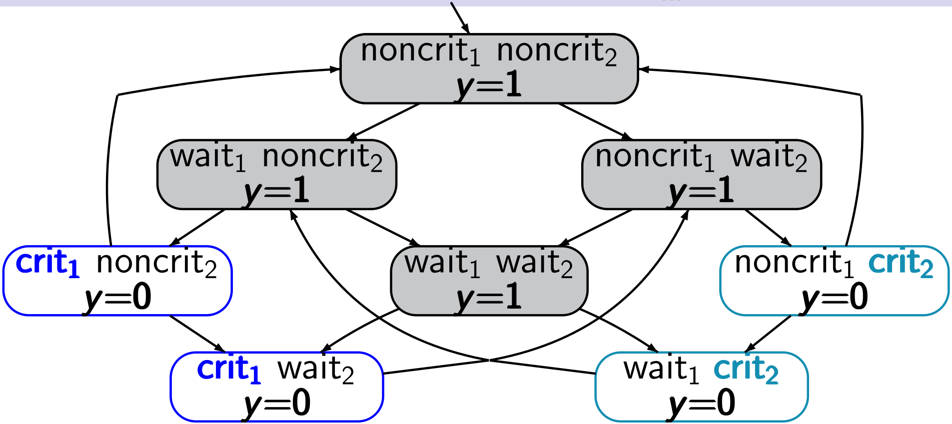
transition system $\mathcal{T}_{\mathcal{P}_1 ||| \mathcal{P}_2}$ arises by unfolding the composite program graph $\mathcal{P}_1 ||| \mathcal{P}_2$

Mutual exclusion with semaphore $\mathcal{T}_{\mathcal{P}_1 ||| \mathcal{P}_2}$



set of atomic propositions $AP = \{\mathbf{crit_1}, \mathbf{crit_2}\}$

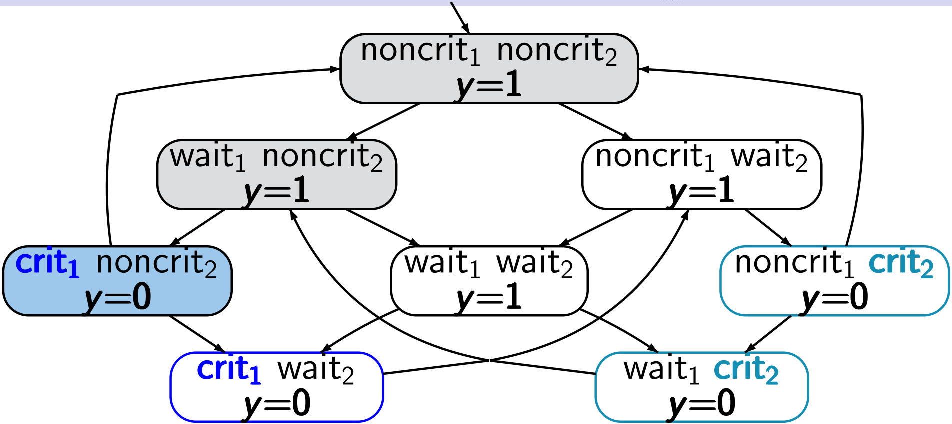
Mutual exclusion with semaphore $\mathcal{T}_{\mathcal{P}_1 ||| \mathcal{P}_2}$



set of atomic propositions $AP = \{crit_1, crit_2\}$

e.g., $L(\langle noncrit_1, noncrit_2, y{=}1\rangle) =$
$L(\langle wait_1, noncrit_2, y{=}1\rangle) = \varnothing$

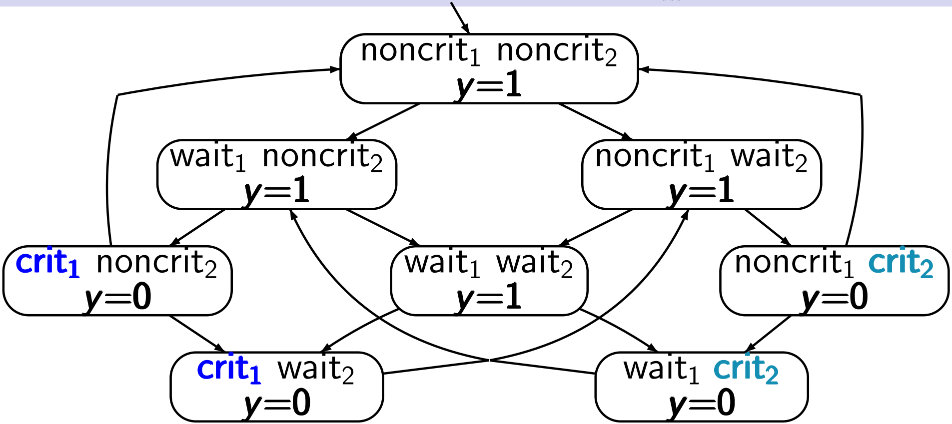
Mutual exclusion with semaphore $\mathcal{T}_{\mathcal{P}_1 ||| \mathcal{P}_2}$



set of atomic propositions $AP = \{crit_1, crit_2\}$

traces, e.g., $\varnothing\ \varnothing\ \{crit_1\}\ \varnothing\ \varnothing\ \{crit_1\}\ \varnothing\ \varnothing\ \{crit_1\} \ldots$

Mutual exclusion with semaphore $\mathcal{T}_{\mathcal{P}_1 ||| \mathcal{P}_2}$

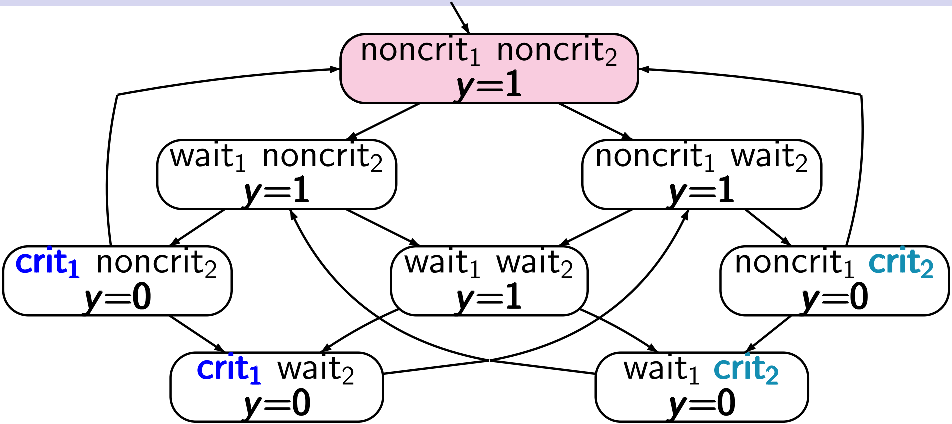


set of atomic propositions $AP = \{crit_1, crit_2\}$

traces, e.g., $\varnothing\ \varnothing\ \{crit_1\}\ \varnothing\ \varnothing\ \{crit_1\}\ \varnothing\ \varnothing\ \{crit_1\} \ldots$

$\varnothing\ \varnothing\ \varnothing\ \{crit_1\}\ \varnothing\ \{crit_2\}\ \{crit_2\}\ \varnothing \ldots$

Mutual exclusion with semaphore $\mathcal{T}_{\mathcal{P}_1 ||| \mathcal{P}_2}$

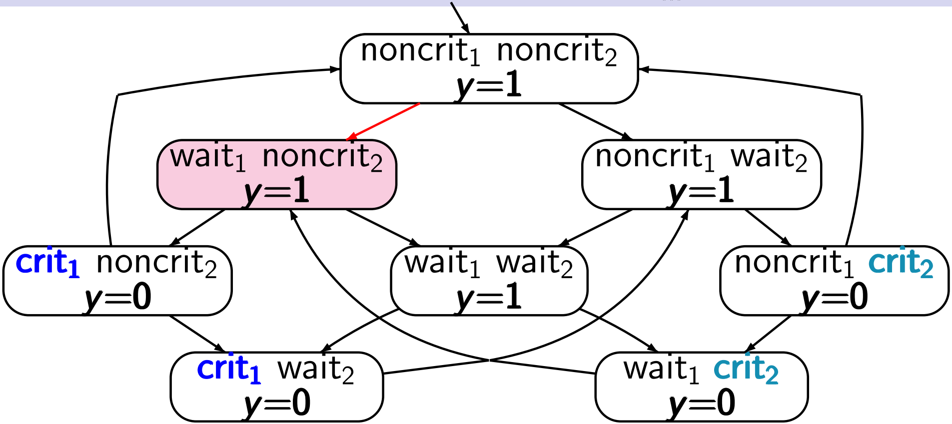


set of atomic propositions $AP = \{crit_1, crit_2\}$

traces, e.g., $\varnothing\ \varnothing\ \{crit_1\}\ \varnothing\ \varnothing\ \{crit_1\}\ \varnothing\ \varnothing\ \{crit_1\} \ldots$

$\varnothing\ \varnothing\ \varnothing\ \{crit_1\}\ \varnothing\ \{crit_2\}\ \{crit_2\}\ \varnothing \ldots$

Mutual exclusion with semaphore $\mathcal{T}_{\mathcal{P}_1 ||| \mathcal{P}_2}$

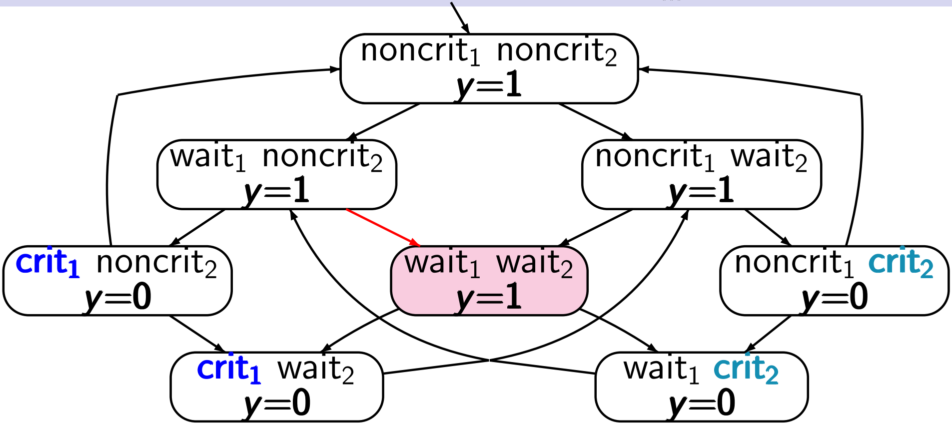


set of atomic propositions $AP = \{crit_1, crit_2\}$

traces, e.g., $\varnothing\ \varnothing\ \{crit_1\}\ \varnothing\ \varnothing\ \{crit_1\}\ \varnothing\ \varnothing\ \{crit_1\} \ldots$

$\varnothing\ \varnothing\ \varnothing\ \{crit_1\}\ \varnothing\ \{crit_2\}\ \{crit_2\}\ \varnothing \ldots$

Mutual exclusion with semaphore $\mathcal{T}_{\mathcal{P}_1 ||| \mathcal{P}_2}$

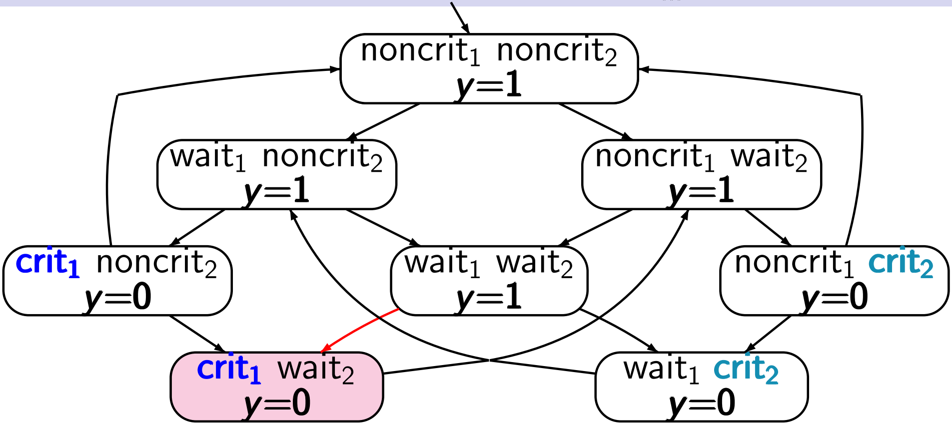


set of atomic propositions $AP = \{crit_1, crit_2\}$

traces, e.g., $\varnothing\ \varnothing\ \{crit_1\}\ \varnothing\ \varnothing\ \{crit_1\}\ \varnothing\ \varnothing\ \{crit_1\} \ldots$

$\varnothing\ \varnothing\ \varnothing\ \{crit_1\}\ \varnothing\ \{crit_2\}\ \{crit_2\}\ \varnothing \ldots$

Mutual exclusion with semaphore $\mathcal{T}_{\mathcal{P}_1 ||| \mathcal{P}_2}$

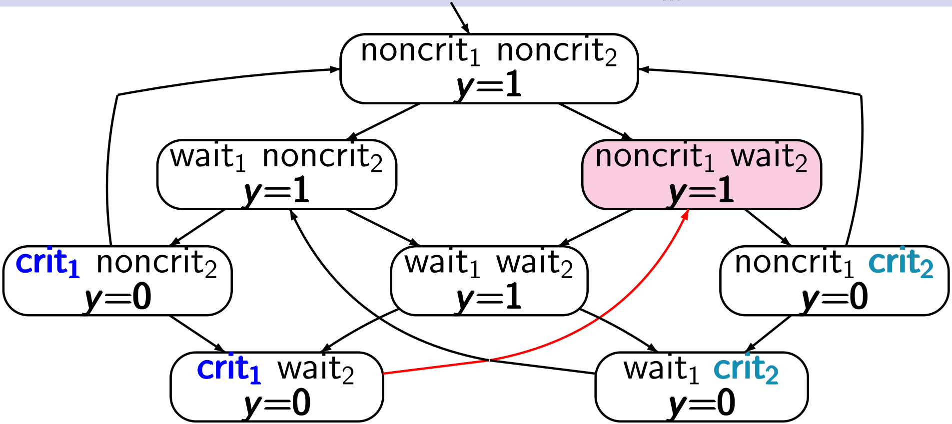


set of atomic propositions $AP = \{crit_1, crit_2\}$

traces, e.g., $\varnothing\ \varnothing\ \{crit_1\}\ \varnothing\ \varnothing\ \{crit_1\}\ \varnothing\ \varnothing\ \{crit_1\} \ldots$

$\varnothing\ \varnothing\ \varnothing\ \{crit_1\}\ \varnothing\ \{crit_2\}\ \{crit_2\}\ \varnothing \ldots$

Mutual exclusion with semaphore $\mathcal{T}_{\mathcal{P}_1 ||| \mathcal{P}_2}$

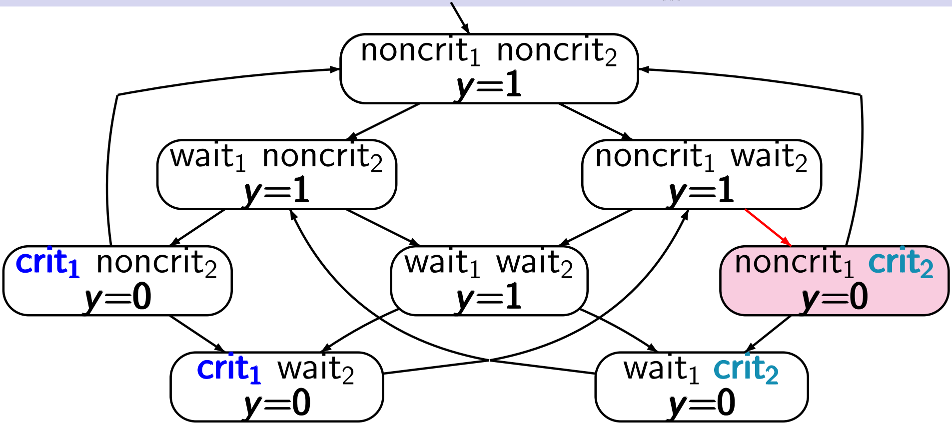


set of atomic propositions $AP = \{crit_1, crit_2\}$

traces, e.g., $\varnothing\ \varnothing\ \{crit_1\}\ \varnothing\ \varnothing\ \{crit_1\}\ \varnothing\ \varnothing\ \{crit_1\} \ldots$

$\varnothing\ \varnothing\ \varnothing\ \{crit_1\}\ \varnothing\ \{crit_2\}\ \{crit_2\}\ \varnothing \ldots$

Mutual exclusion with semaphore $\mathcal{T}_{\mathcal{P}_1 ||| \mathcal{P}_2}$

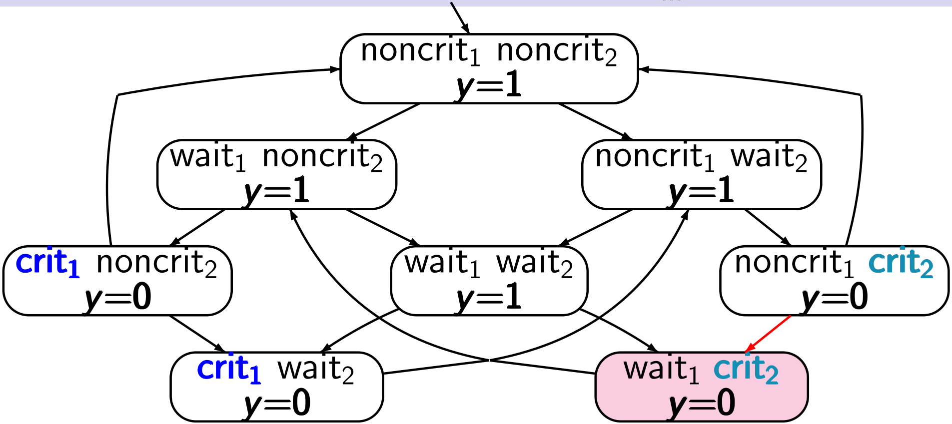


set of atomic propositions $AP = \{crit_1, crit_2\}$

traces, e.g., $\varnothing\,\varnothing\,\{crit_1\}\,\varnothing\,\varnothing\,\{crit_1\}\,\varnothing\,\varnothing\,\{crit_1\}\ldots$

$\varnothing\,\varnothing\,\varnothing\,\{crit_1\}\,\varnothing\,\{crit_2\}\,\{crit_2\}\,\varnothing\ldots$

Mutual exclusion with semaphore $\mathcal{T}_{\mathcal{P}_1 ||| \mathcal{P}_2}$

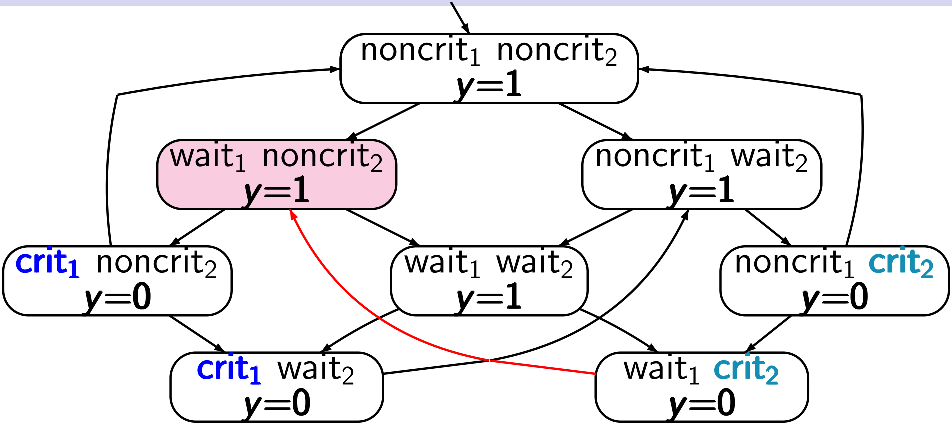


set of atomic propositions $AP = \{crit_1, crit_2\}$

traces, e.g., $\varnothing\ \varnothing\ \{crit_1\}\ \varnothing\ \varnothing\ \{crit_1\}\ \varnothing\ \varnothing\ \{crit_1\} \ldots$

$\varnothing\ \varnothing\ \varnothing\ \{crit_1\}\ \varnothing\ \{crit_2\}\ \{crit_2\}\ \varnothing \ldots$

Mutual exclusion with semaphore $\mathcal{T}_{\mathcal{P}_1 ||| \mathcal{P}_2}$

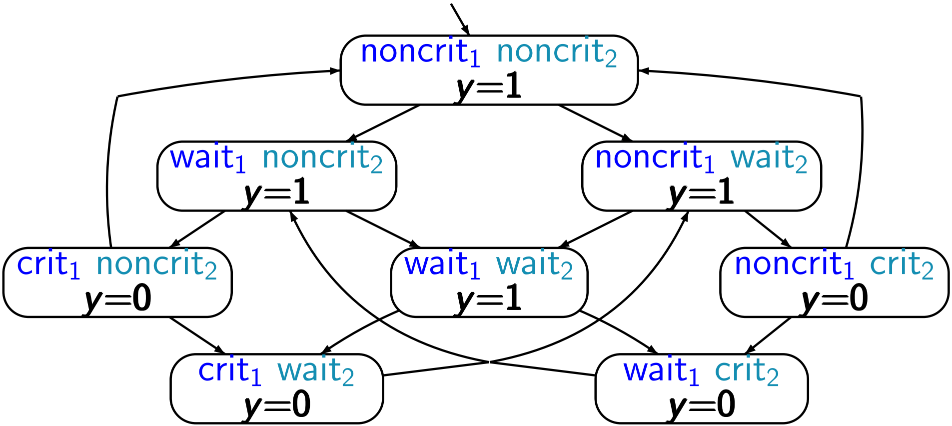


set of atomic propositions $AP = \{crit_1, crit_2\}$

traces, e.g., $\varnothing\ \varnothing\ \{crit_1\}\ \varnothing\ \varnothing\ \{crit_1\}\ \varnothing\ \varnothing\ \{crit_1\} \ldots$

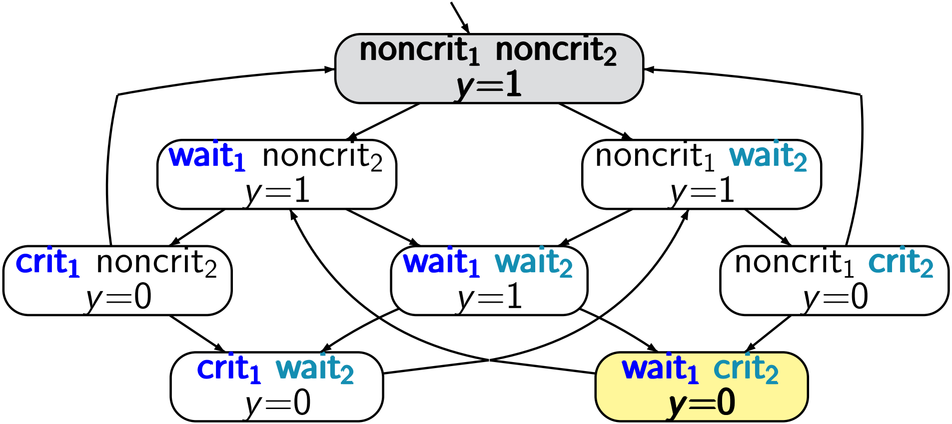
$\varnothing\ \varnothing\ \varnothing\ \{crit_1\}\ \varnothing\ \{crit_2\}\ \{crit_2\}\ \varnothing \ldots$

Mutual exclusion with semaphor $\mathcal{T}_{\mathcal{P}_1 ||| \mathcal{P}_2}$



set of propositions $AP = \{wait_1, crit_1, wait_2, crit_2\}$

Mutual exclusion with semaphor $\mathcal{T}_{\mathcal{P}_1 ||| \mathcal{P}_2}$

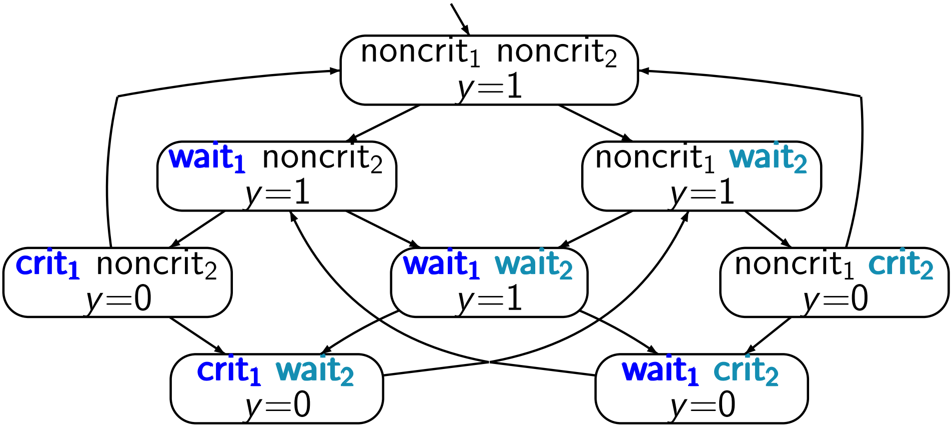


set of propositions $AP = \{wait_1, crit_1, wait_2, crit_2\}$

e.g., $L(\langle noncrit_1, noncrit_2, y{=}1\rangle) = \varnothing$

$L(\langle wait_1, crit_2, y{=}1\rangle) = \{wait_1, crit_2\}$

Mutual exclusion with semaphor $\mathcal{T}_{\mathcal{P}_1 ||| \mathcal{P}_2}$

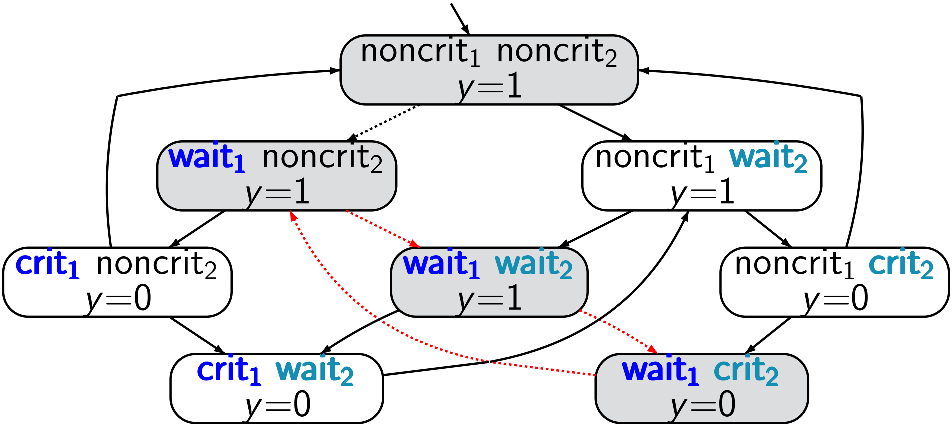


set of propositions $AP = \{\text{wait}_1, \text{crit}_1, \text{wait}_2, \text{crit}_2\}$

traces, e.g.,

$$\varnothing \left(\{\text{wait}_1\}\,\{\text{wait}_1, \text{wait}_2\}\,\{\text{wait}_1, \text{crit}_2\}\right)^{\omega}$$

Mutual exclusion with semaphor $\mathcal{T}_{\mathcal{P}_1 ||| \mathcal{P}_2}$

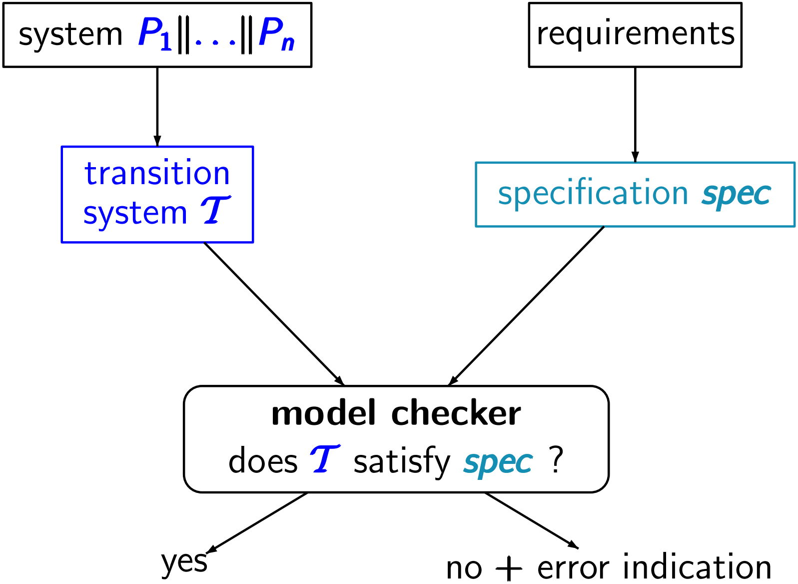


set of propositions $AP = \{wait_1, crit_1, wait_2, crit_2\}$

traces, e.g.,

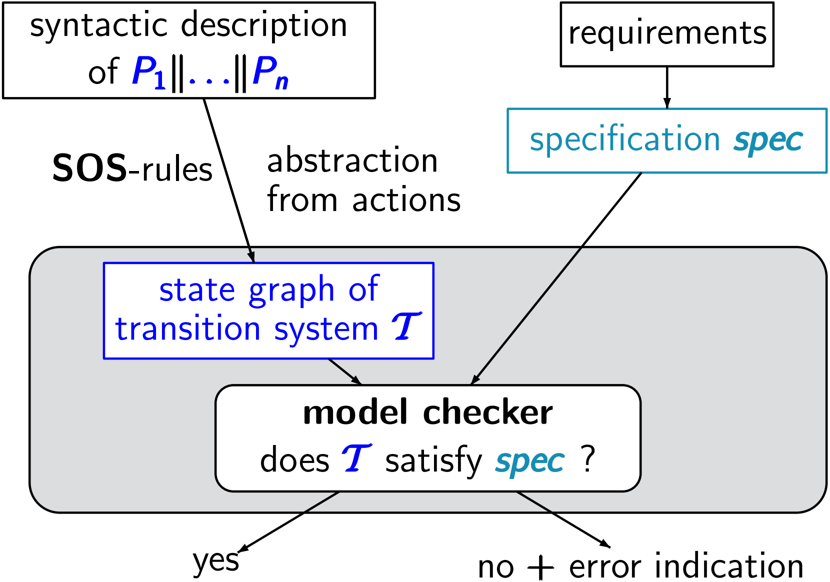
$$\varnothing\ (\{wait_1\}\ \{wait_1, wait_2\}\ \{wait_1, crit_2\})^\omega$$

- 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

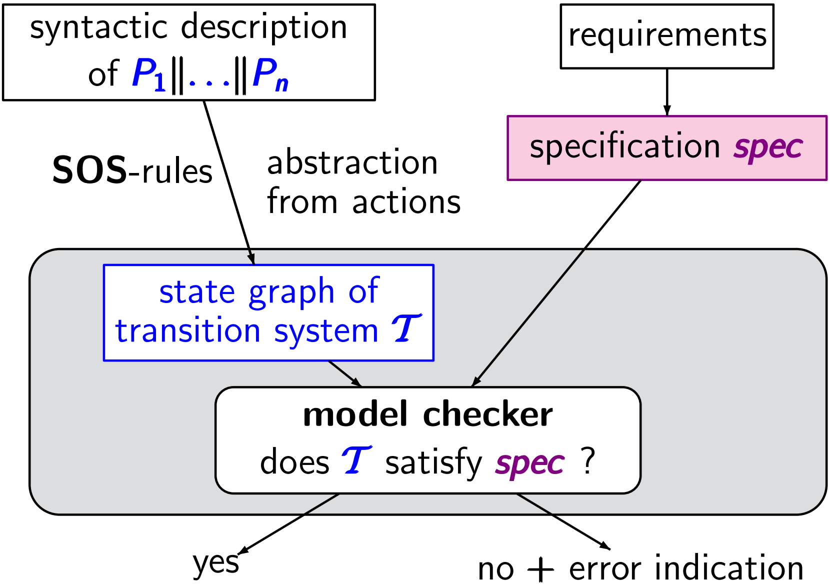


system $P_1 \| \ldots \| P_n$
requirements
transition system $\mathcal{T}$
specification *spec*
model checker
does $\mathcal{T}$ satisfy *spec* ?
yes
no + error indication

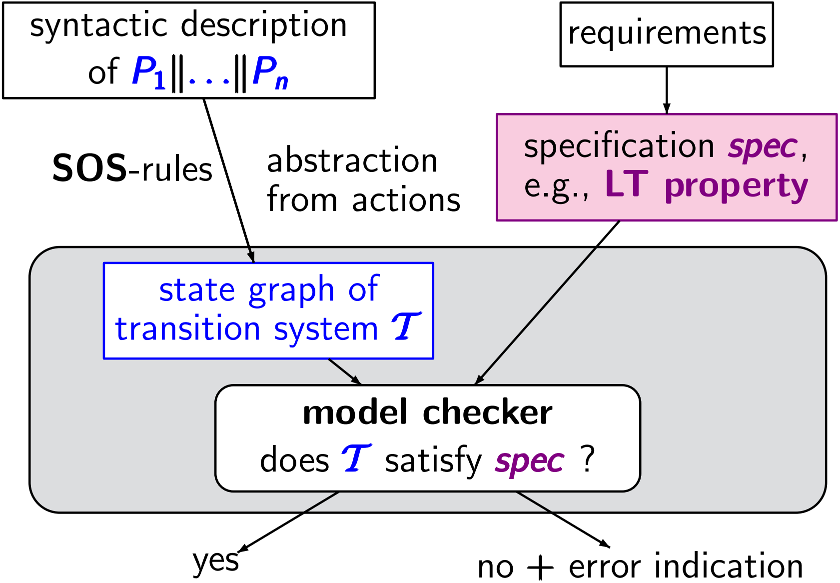
Model checking



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 $\Sigma = 2^{AP}$, 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:

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

LT properties for mutual exclusion protocols

$AP = \{wait_1, crit_1, wait_2, crit_2\}$

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 $crit_2 \notin A_i$

$\varnothing\ \{wait_1\}\ \{crit_1\}\ \varnothing\ \{wait_1\}\ \{crit_1\}\ \ldots \qquad \in MUTEX$

LT properties for mutual exclusion protocols

$AP = \{wait_1, crit_1, wait_2, crit_2\}$

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 $crit_2 \notin A_i$

$\varnothing \{wait_1\} \{crit_1\} \varnothing \{wait_1\} \{crit_1\} \ldots \quad \in MUTEX$

$\varnothing \{wait_1\} \{crit_1\} \{crit_1, wait_2\} \{crit_1, crit_2\} \ldots \notin MUTEX$

$AP = \{wait_1, crit_1, wait_2, crit_2\}$

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 $crit_2 \notin A_i$

$\varnothing \; \{wait_1\} \, \{crit_1\} \, \varnothing \; \{wait_1\} \, \{crit_1\} \; \ldots \quad \in MUTEX$

$\varnothing \, \{wait_1\} \, \{crit_1\} \, \{crit_1, wait_2\} \, \{crit_1, crit_2\} \ldots \notin MUTEX$

$\varnothing \, \varnothing \, \{wait_1, crit_1, crit_2\} \ldots \quad \notin MUTEX$

LT properties for mutual exclusion protocols

$AP = \{wait_1, crit_1, wait_2, crit_2\}$

safety:

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

$$crit_1 \notin A_i \quad \text{or} \quad crit_2 \notin A_i$$

liveness (starvation freedom):

$LIVE$ = set of all infinite words $A_0\, A_1\, A_2 \ldots$ s.t.

$$\overset{\infty}{\exists}\, i \in \mathbb{N}.wait_1 \in A_i \implies \overset{\infty}{\exists}\, i \in \mathbb{N}.crit_1 \in A_i$$

$$\wedge \quad \overset{\infty}{\exists}\, i \in \mathbb{N}.wait_2 \in A_i \implies \overset{\infty}{\exists}\, i \in \mathbb{N}.crit_2 \in A_i$$

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

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:

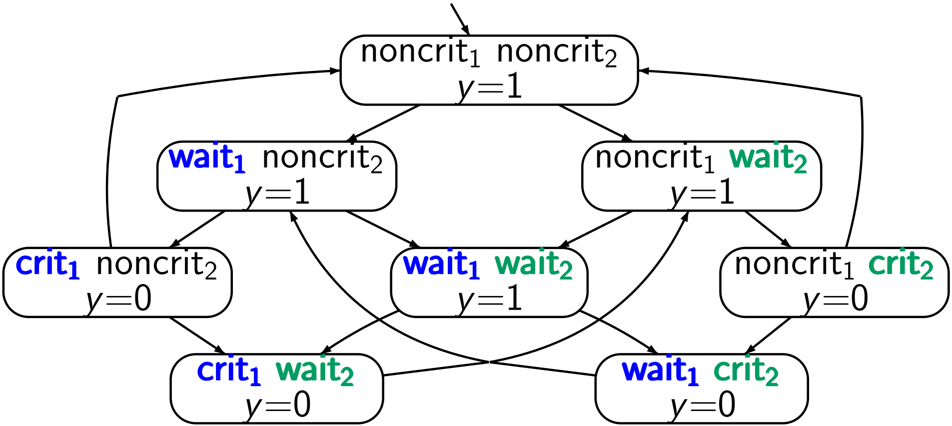
If $\mathcal{T}$ is a TS (without terminal states) over AP and E an LT property over AP then

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

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

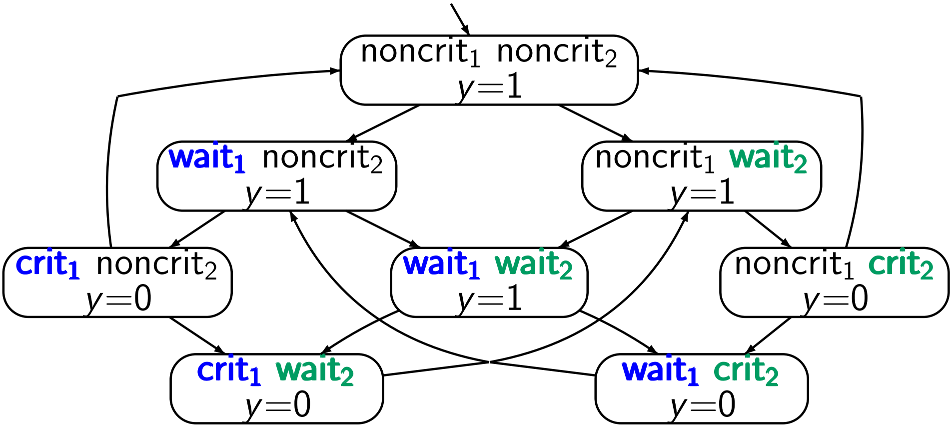
$$s \models E \quad \text{iff} \quad Traces(s) \subseteq E$$

Mutual exclusion with semaphore



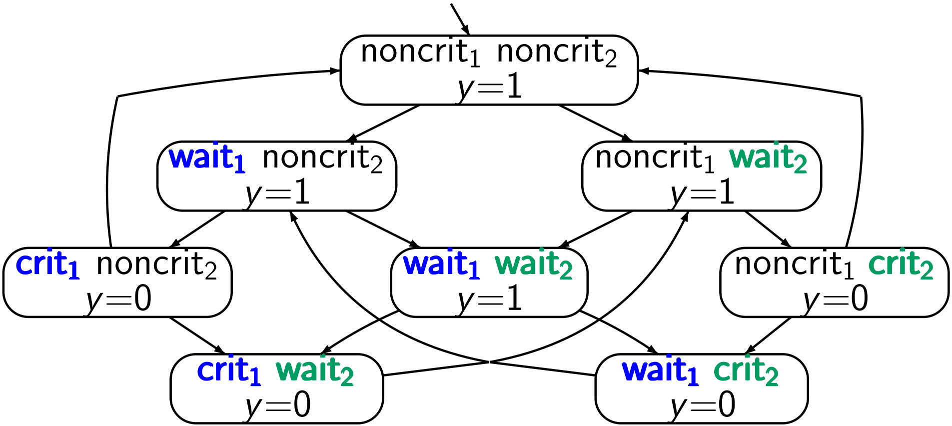
$\mathcal{T}_{Sem} \models MUTEX$

Mutual exclusion with semaphore



$\mathcal{T}_{Sem} \models MUTEX, \quad \mathcal{T}_{Sem} \models LIVE$?

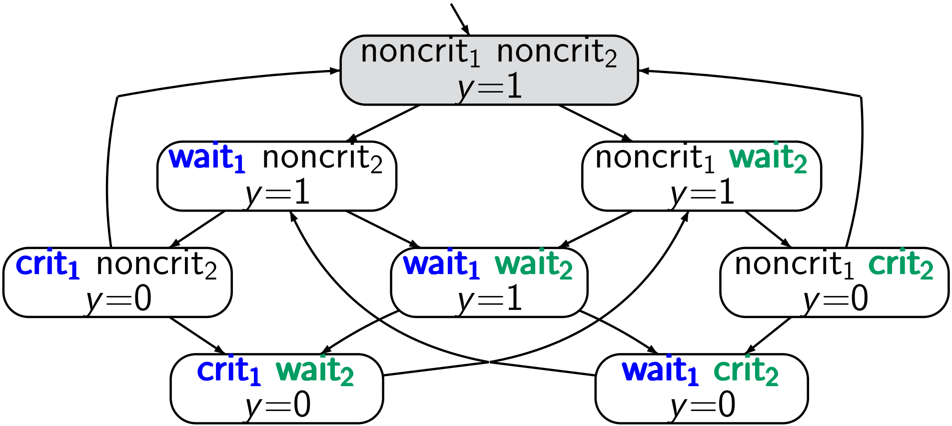
Mutual exclusion with semaphore



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

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

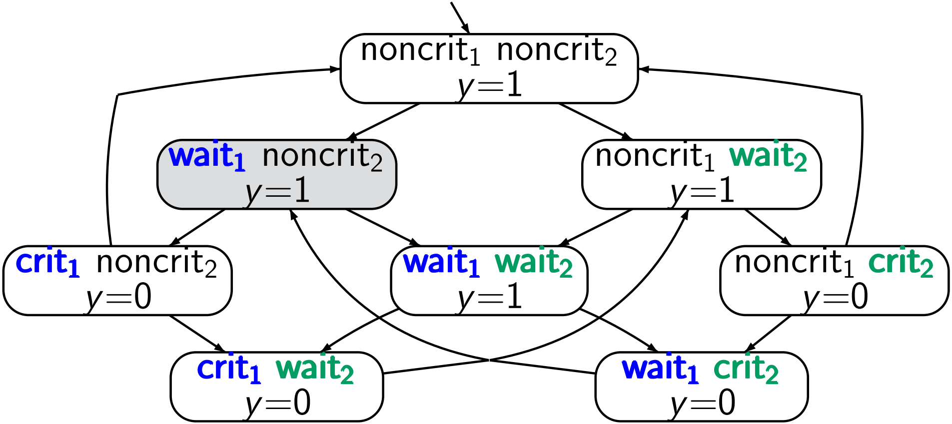
Mutual exclusion with semaphore



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

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

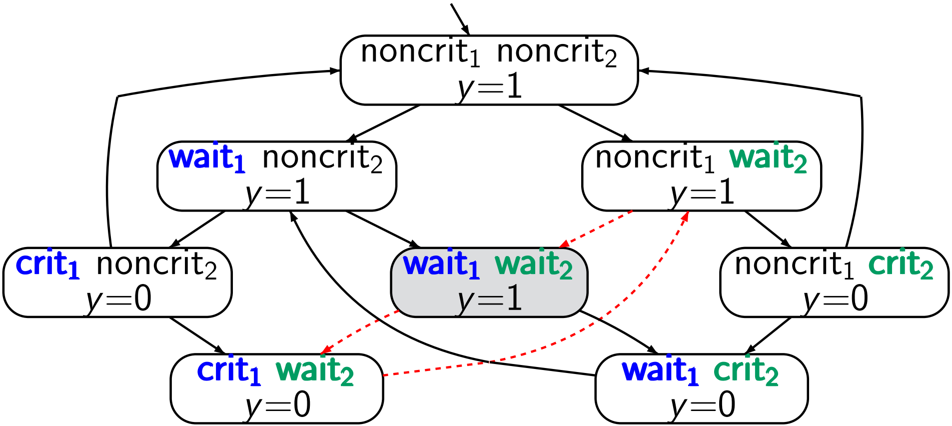
Mutual exclusion with semaphore



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

$$\varnothing \, \{\text{wait}_1\} \left(\{\text{wait}_1, \text{wait}_2\} \{\text{crit}_1, \text{wait}_2\} \{\text{wait}_2\} \right)^{\omega} \notin LIVE$$

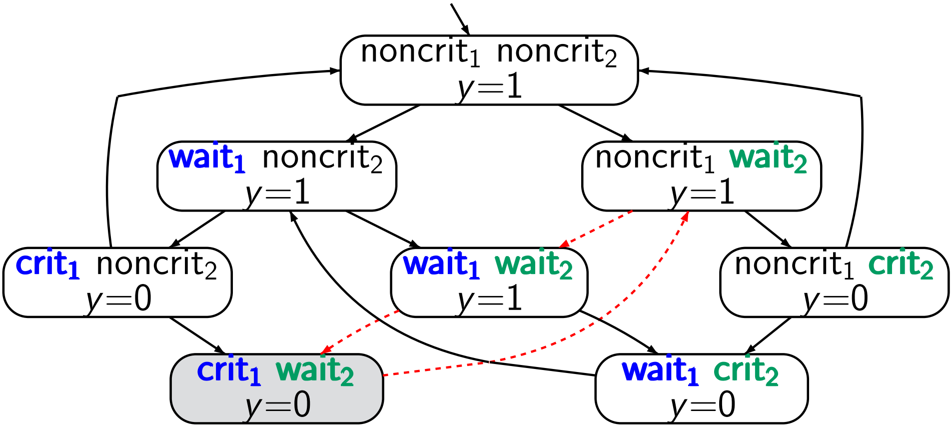
Mutual exclusion with semaphore



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

$$\varnothing \, \{wait_1\} \left(\{wait_1, wait_2\} \, \{crit_1, wait_2\} \{wait_2\} \right)^{\omega} \notin \mathit{LIVE}$$

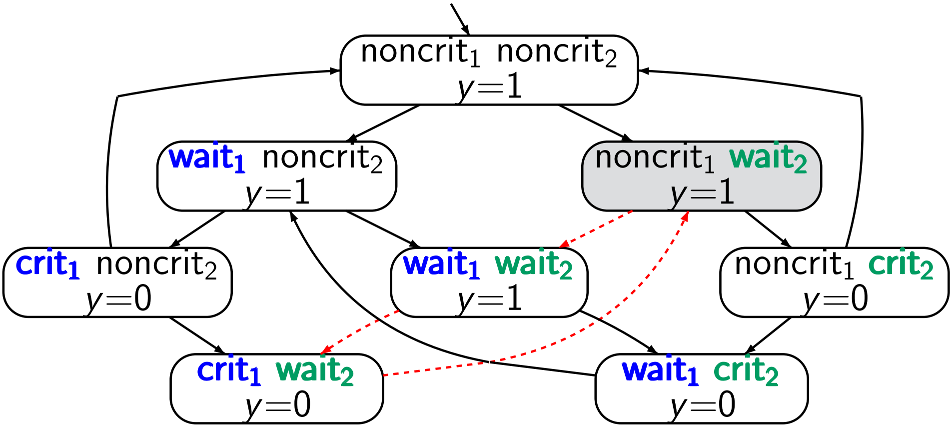
Mutual exclusion with semaphore



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

$$\varnothing \, \{\mathsf{wait}_1\} \left(\{\mathsf{wait}_1, \mathsf{wait}_2\} \, \{\mathsf{crit}_1, \mathsf{wait}_2\} \{\mathsf{wait}_2\} \right)^\omega \notin LIVE$$

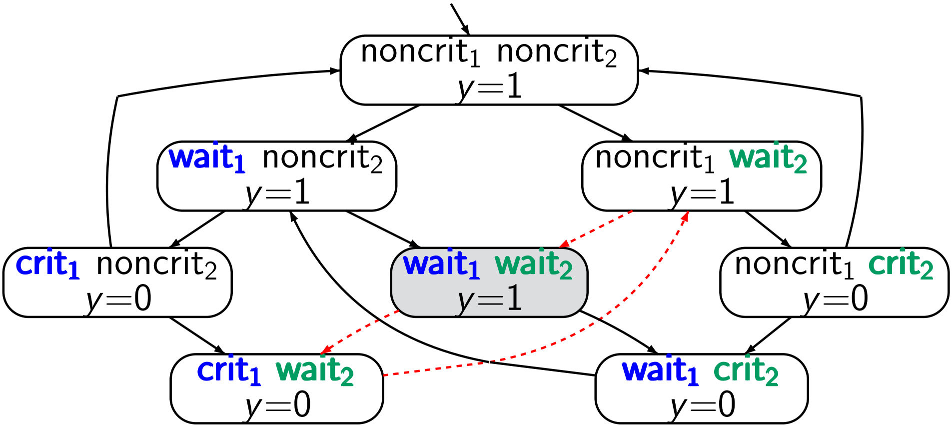
Mutual exclusion with semaphore



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

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

Mutual exclusion with semaphore



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

$$\varnothing\,\{wait_1\}\,\big(\,\{wait_1, wait_2\}\,\{crit_1, wait_2\}\{wait_2\}\,\big)^{\omega} \notin LIVE$$

Peterson's mutual exclusion algorithm

LTB2.4-17

Peterson's mutual exclusion algorithm

for competing processes $\mathcal{P}_1$ and $\mathcal{P}_2$,

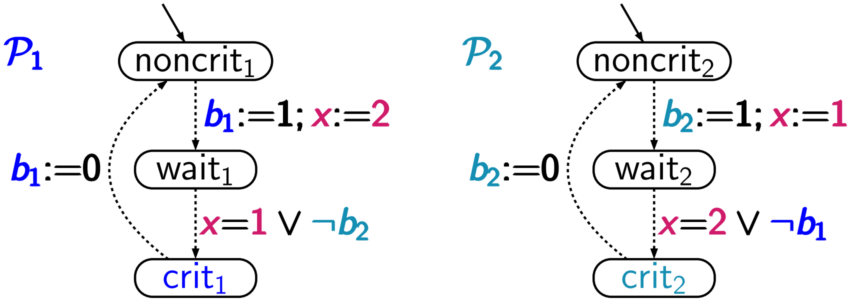
using three additional shared variables

$b_1, b_2 \in \{0, 1\}$, $x \in \{1, 2\}$

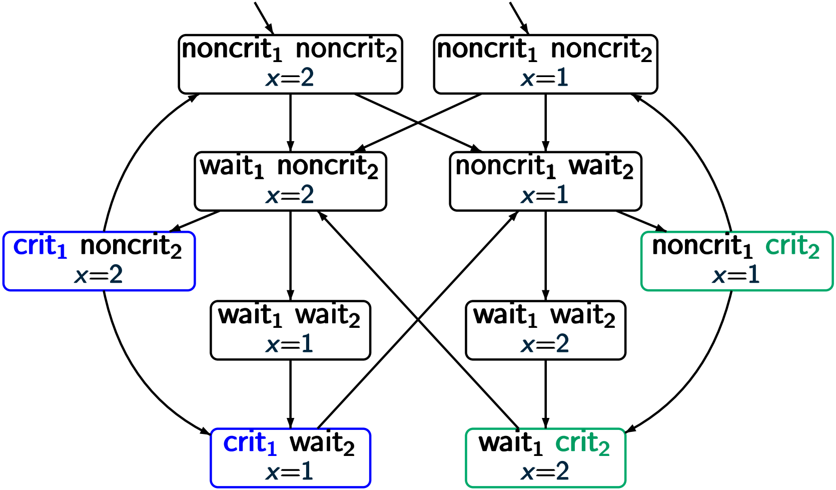
Peterson's mutual exclusion algorithm

for competing processes $\mathcal{P}_1$ and $\mathcal{P}_2$,

using three additional shared variables
$b_1, b_2 \in \{0,1\}$, $x \in \{1,2\}$

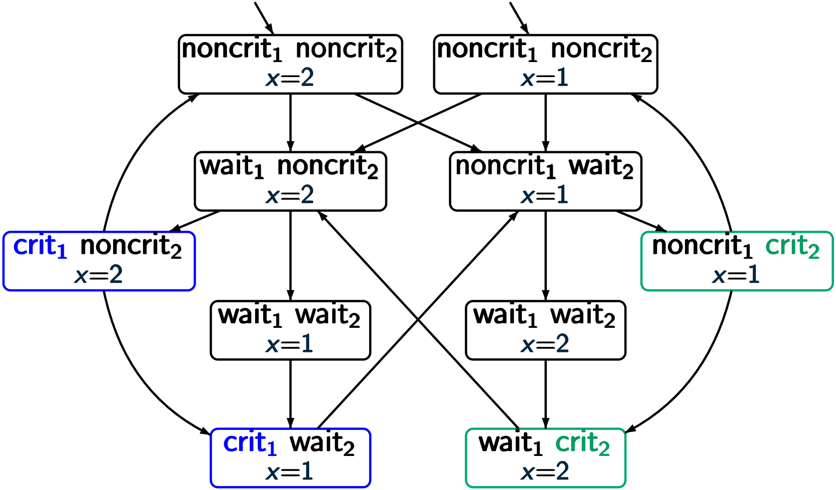


Peterson's mutual exclusion algorithm



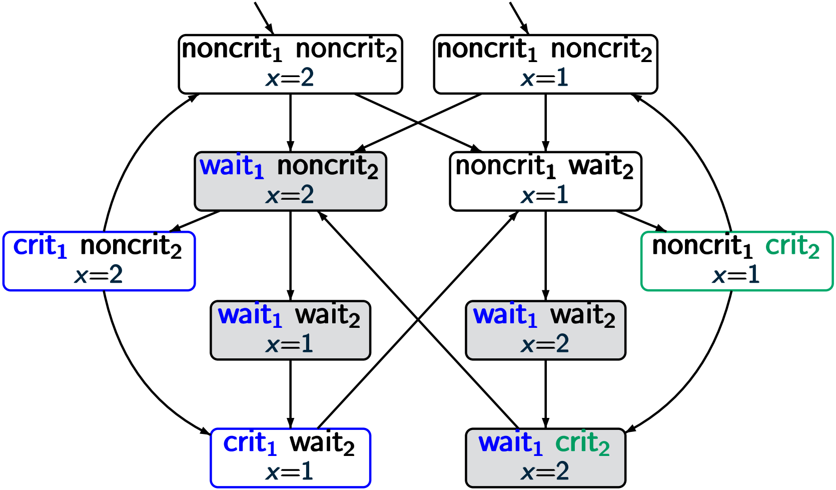
$\mathcal{T}_{Pet} \models MUTEX$

Peterson's mutual exclusion algorithm



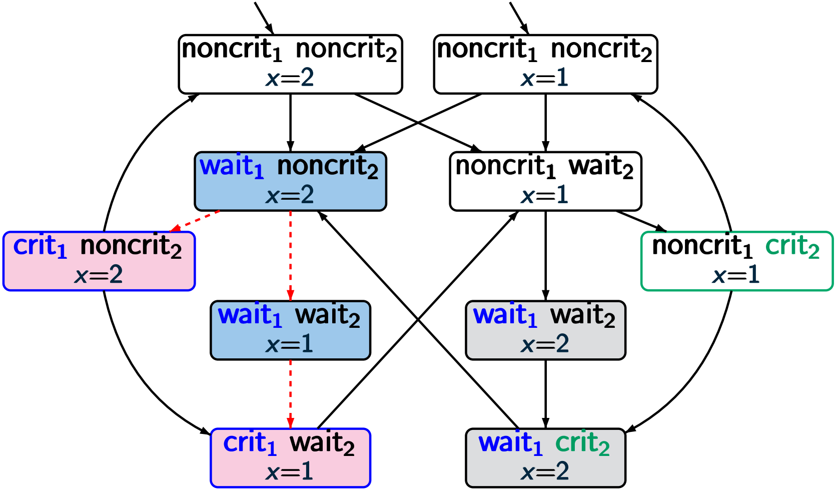
$\mathcal{T}_{Pet} \models$ *MUTEX* and $\mathcal{T}_{Pet} \models$ *LIVE*

Peterson's mutual exclusion algorithm



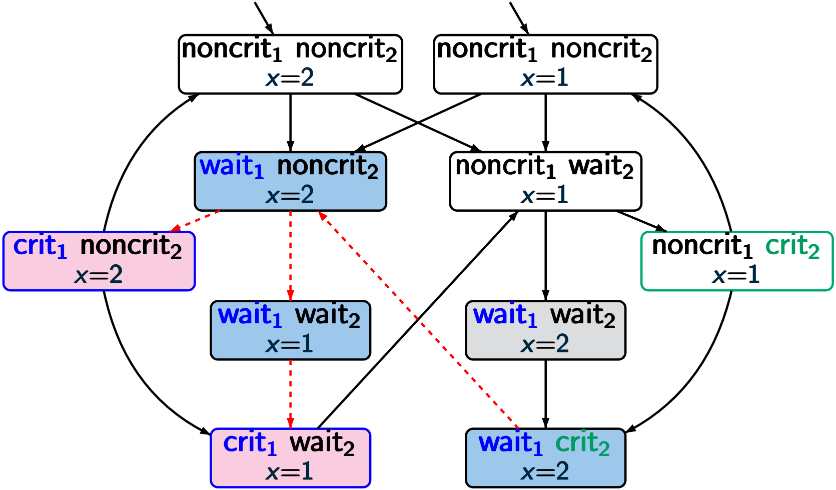
$\mathcal{T}_{Pet} \models \mathit{MUTEX}$ and $\mathcal{T}_{Pet} \models \mathit{LIVE}$

Peterson's mutual exclusion algorithm



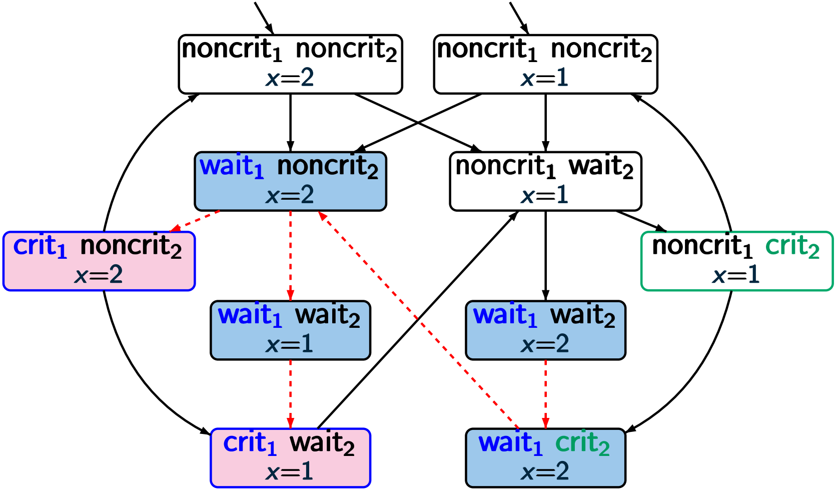
$\mathcal{T}_{Pet} \models MUTEX$ and $\mathcal{T}_{Pet} \models LIVE$

Peterson's mutual exclusion algorithm



$\mathcal{T}_{Pet} \models \mathit{MUTEX}$ and $\mathcal{T}_{Pet} \models \mathit{LIVE}$

Peterson's mutual exclusion algorithm



$\mathcal{T}_{Pet} \models \mathit{MUTEX}$ and $\mathcal{T}_{Pet} \models \mathit{LIVE}$

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:

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

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

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

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 $\mathcal{T}_1 \models E$

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

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

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

> 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 $\mathcal{T}_1 \models E$

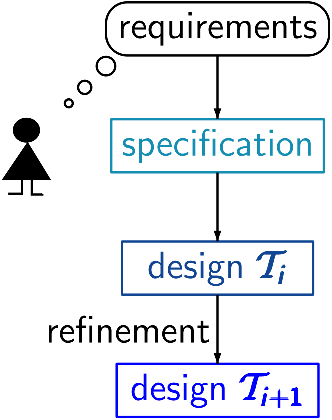
$(2) \Longrightarrow (1)$: consider $E = Traces(\mathcal{T}_2)$

Relevance of trace inclusion

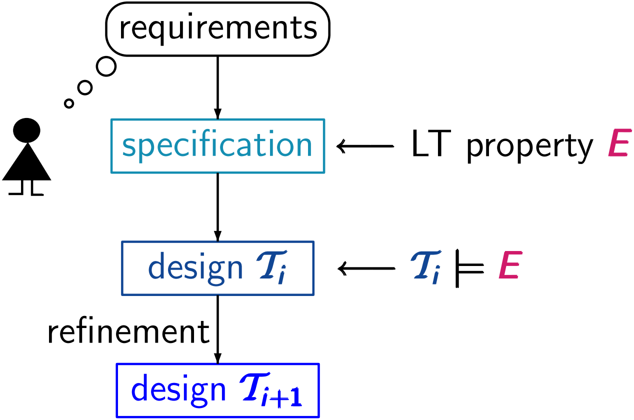
Trace inclusion appears naturally

- as an implementation/refinement relation
- when resolving nondeterminism
- in the context of abstractions

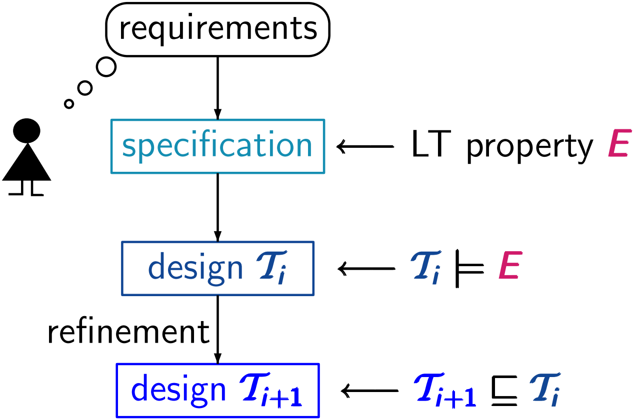
Software design cycle



Software design cycle



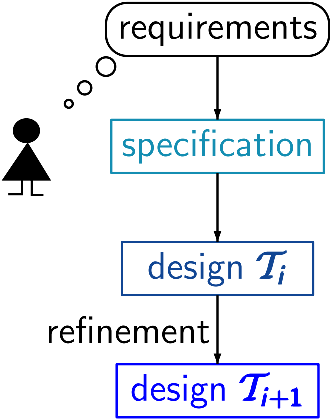
Software design cycle



implementation/refinement relation $\sqsubseteq$:

$\mathcal{T}_{i+1} \sqsubseteq \mathcal{T}_i$ iff "$\mathcal{T}_{i+1}$ correctly implements $\mathcal{T}_i$"

Trace inclusion as an implementation relation



trace inclusion

$\mathcal{T}_{i+1} \sqsubseteq \mathcal{T}_i$ iff

$\mathit{Traces}(\mathcal{T}_{i+1}) \subseteq \mathit{Traces}(\mathcal{T}_i)$

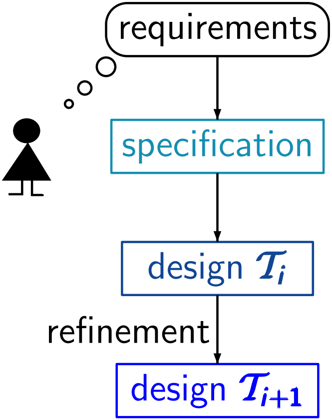
design $\mathcal{T}_i$ $\longleftarrow$ $\mathcal{T}_i \models E$

design $\mathcal{T}_{i+1}$ $\longleftarrow$ $\mathcal{T}_{i+1} \sqsubseteq \mathcal{T}_i$

implementation/refinement relation $\sqsubseteq$:

$\mathcal{T}_{i+1} \sqsubseteq \mathcal{T}_i$ iff "$\mathcal{T}_{i+1}$ correctly implements $\mathcal{T}_i$"

Trace inclusion as an implementation relation



trace inclusion

$\mathcal{T}_{i+1} \sqsubseteq \mathcal{T}_i$ iff

$\mathit{Traces}(\mathcal{T}_{i+1}) \subseteq \mathit{Traces}(\mathcal{T}_i)$

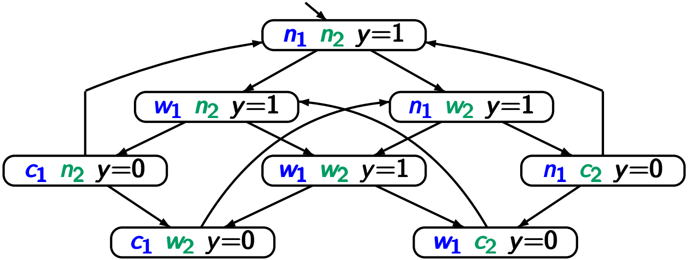
$\longleftarrow \mathcal{T}_i \models E$

$\longleftarrow \mathcal{T}_{i+1} \sqsubseteq \mathcal{T}_i$ implies $\mathcal{T}_{i+1} \models E$

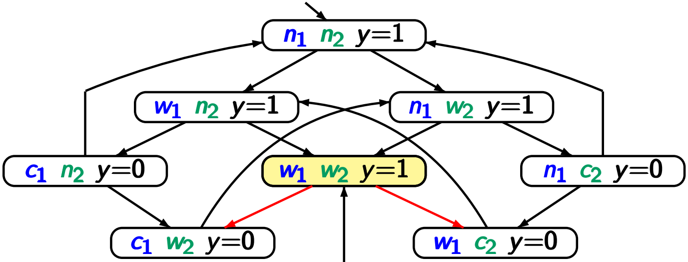
implementation/refinement relation $\sqsubseteq$:

$\mathcal{T}_{i+1} \sqsubseteq \mathcal{T}_i$ iff "$\mathcal{T}_{i+1}$ correctly implements $\mathcal{T}_i$"

Mutual exclusion with semaphore

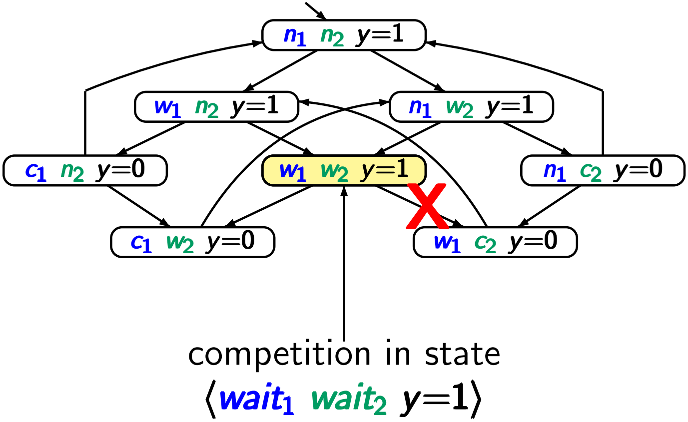


Mutual exclusion with semaphore



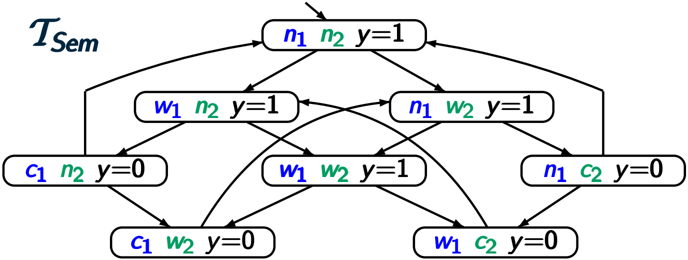
competition in state
$\langle wait_1\ wait_2\ y{=}1 \rangle$

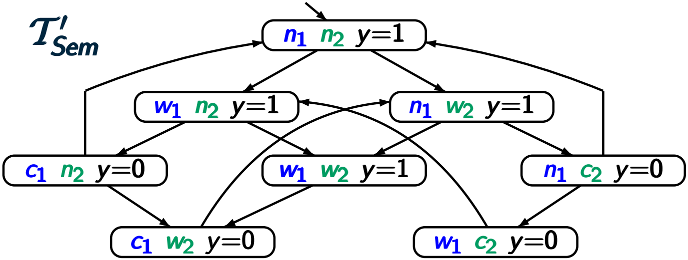
Mutual exclusion with semaphore



resolve the nondeterminism by giving priority to process P_1

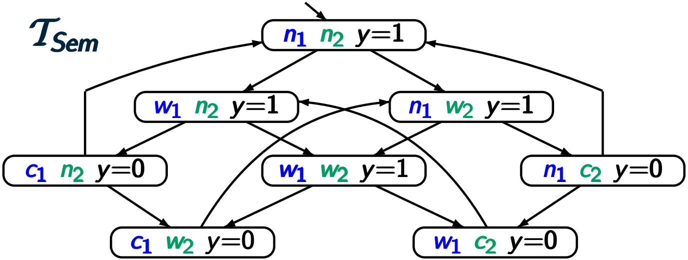
Mutual exclusion with semaphore



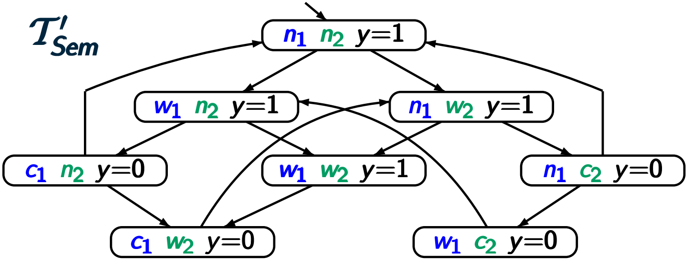


Mutual exclusion with semaphore

$\mathcal{T}_{Sem}$



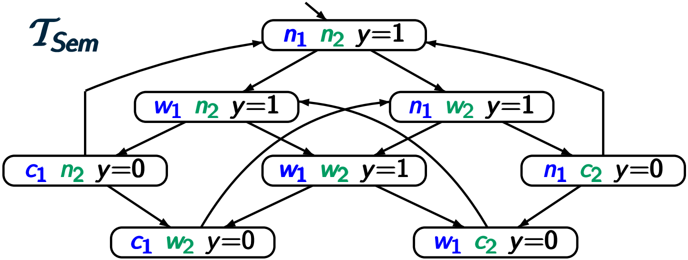
$\mathcal{T}'_{Sem}$



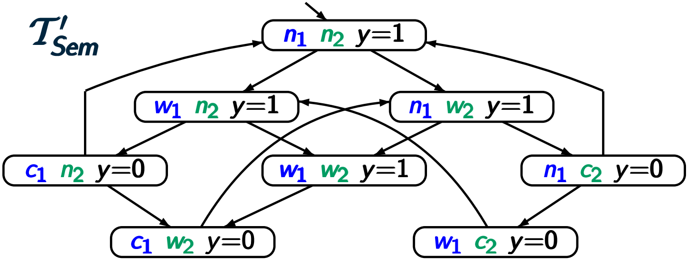
$$Paths(\mathcal{T}'_{Sem}) \subseteq Paths(\mathcal{T}_{Sem})$$

Mutual exclusion with semaphore

$\mathcal{T}_{Sem}$

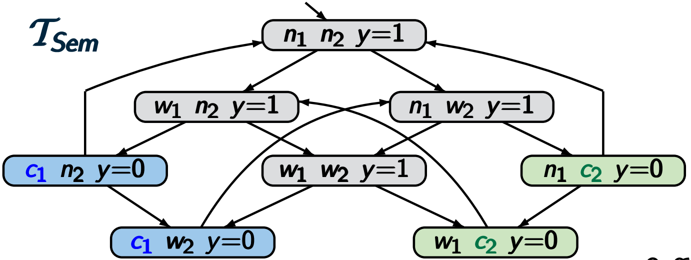


$\mathcal{T}'_{Sem}$

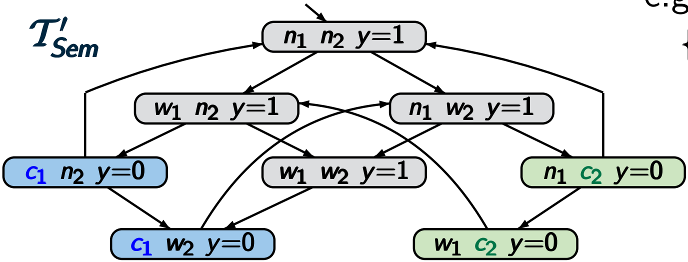


$Traces(\mathcal{T}'_{Sem}) \subseteq Traces(\mathcal{T}_{Sem})$ for any AP

Mutual exclusion with semaphore



e.g., for $AP = \{crit_1, crit_2\}$



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

Relevance of trace inclusion

Trace inclusion appears naturally

- as an implementation/refinement relation
- when resolving nondeterminism $\longleftarrow$

  e.g., $\mathit{Traces}(\mathcal{T}'_{\mathit{Sem}}) \subseteq \mathit{Traces}(\mathcal{T}_{\mathit{Sem}})$
- in the context of abstractions

Relevance of trace inclusion

Trace inclusion appears naturally

- as an implementation/refinement relation
- when resolving nondeterminism

whenever $\mathcal{T}'$ results from $\mathcal{T}$ by a scheduling policy for resolving nondeterministic choices in $\mathcal{T}$ then

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

- in the context of abstractions

Relevance of trace inclusion

Trace inclusion appears naturally

- as an implementation/refinement relation
- when resolving nondeterminism
- in the context of abstractions $\longleftarrow$

Trace inclusion and data abstraction

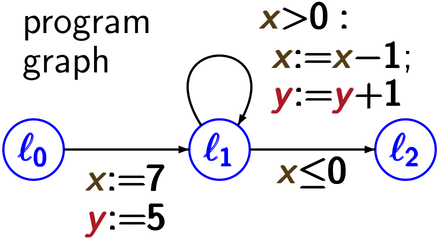

```
  ⋮
x:=7;  y:=5;
WHILE x>0 DO
     x:=x−1;
     y:=y+1
OD
  ⋮
```

```
        ⋮
ℓ0   x:=7; y:=5;
ℓ1   WHILE x>0 DO
            x:=x−1;
            y:=y+1
     OD
ℓ2      ⋮
```

does $\ell_2 \wedge odd(y)$
never hold ?

Trace inclusion and data abstraction

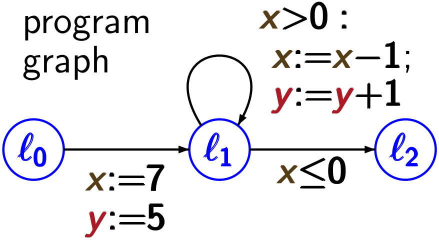
```
        ⋮
ℓ0   x:=7;  y:=5;
ℓ1   WHILE x>0 DO
            x:=x−1;
            y:=y+1
     OD
ℓ2      ⋮
```



does $\ell_2 \wedge odd(y)$
never hold ?

Trace inclusion and data abstraction

```
      ⋮
ℓ0   x:=7;  y:=5;
ℓ1   WHILE x>0 DO
            x:=x−1;
            y:=y+1
     OD
ℓ2    ⋮
```



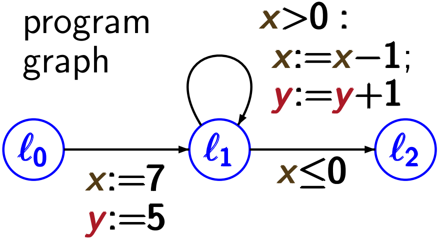
let $\mathcal{T}$ be the associated TS

does $\ell_2 \wedge odd(y)$ never hold ? $\longleftarrow$ $\mathcal{T} \models$ "never $\ell_2 \wedge odd(y)$" ?

Trace inclusion and data abstraction

```
     ⋮
ℓ0   x:=7;  y:=5;
ℓ1   WHILE x>0 DO
            x:=x−1;
            y:=y+1
     OD
ℓ2   ⋮
```

program graph



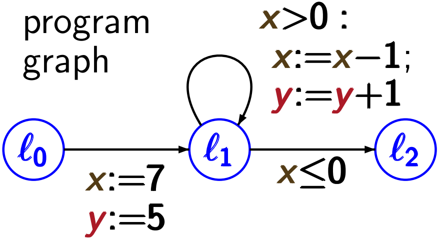
let $\mathcal{T}$ be the associated TS

does $\ell_2 \wedge odd(y)$ never hold ? $\longleftarrow$ $\mathcal{T} \models$ "never $\ell_2 \wedge odd(y)$" ?

data abstraction w.r.t. the predicates

$x>0$, $x=0$, $x \equiv_2 y$

```
      ⋮
ℓ0   x:=7; y:=5;
ℓ1   WHILE x>0 DO
           x:=x−1;
           y:=y+1
     OD
ℓ2    ⋮
```



let $\mathcal{T}$ be the associated TS

does $\ell_2 \wedge odd(y)$ never hold **?** $\longleftarrow$ $\mathcal{T} \models$ "never $\ell_2 \wedge odd(y)$" **?**

data abstraction w.r.t. the predicates

$x>0$, $x=0$, $x \equiv_2 y$ $\longleftarrow$ i.e., $x-y$ is even

Trace inclusion and data abstraction

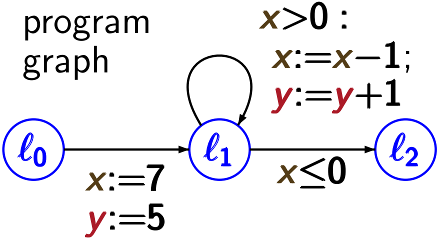

```
        ⋮
ℓ0   x:=7;  y:=5;
ℓ1   WHILE x>0 DO
          x:=x−1;
          y:=y+1
     OD
ℓ2      ⋮
```

does $\ell_2 \wedge \mathit{odd}(y)$ never hold ?

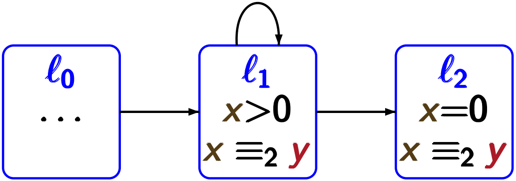
data abstraction w.r.t. the predicates

$x>0$, $x=0$, $x \equiv_2 y$

program graph



let $\mathcal{T}$ be the associated TS

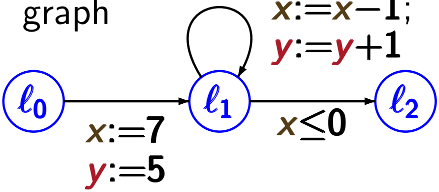


abstract transition system $\mathcal{T}'$

Trace inclusion and data abstraction

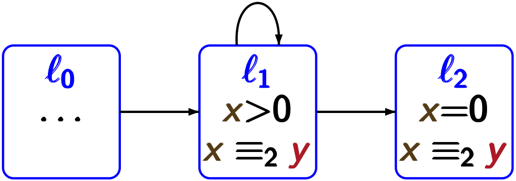

```
      ⋮
ℓ0   x:=7;  y:=5;
ℓ1   WHILE x>0 DO
          x:=x−1;
          y:=y+1
     OD
ℓ2    ⋮
```

program graph



let $\mathcal{T}$ be the associated TS

does $\ell_2 \wedge odd(y)$ never hold ?



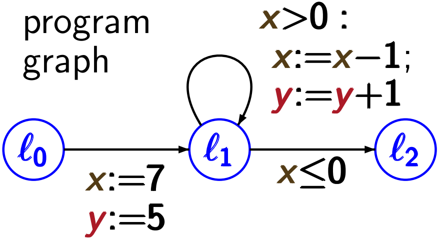
data abstraction w.r.t. the predicates

$x>0$, $x=0$, $x \equiv_2 y$

$\mathcal{T}' \models$ "never $\ell_2 \wedge odd(y)$"

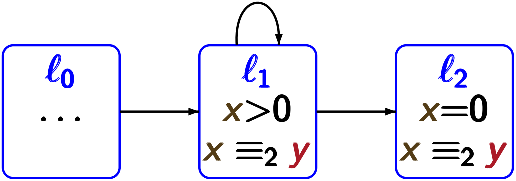
$\vdots$
ℓ_0 x:=7; y:=5;
ℓ_1 WHILE $x>0$ DO
    x:=$x-1$;
    y:=$y+1$
  OD
ℓ_2 $\vdots$

program graph



let $\mathcal{T}$ be the associated TS

does $\ell_2 \wedge odd(y)$ never hold ?



data abstraction w.r.t. the predicates

$x>0$, $x=0$, $x \equiv_2 y$

$\mathcal{T}' \models$ "never $\ell_2 \wedge odd(y)$"

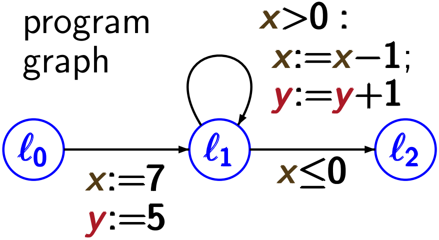
$Traces(\mathcal{T}) \subseteq Traces(\mathcal{T}')$

Trace inclusion and data abstraction

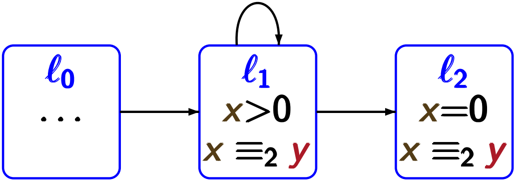

```
        ⋮
ℓ0   x:=7;  y:=5;
ℓ1   WHILE x>0 DO
          x:=x−1;
          y:=y+1
     OD
ℓ2      ⋮
```

does $\ell_2 \wedge odd(y)$
never hold ?

program graph



let $\mathcal{T}$ be the associated TS



$\mathcal{T} \models$ "never $\ell_2 \wedge odd(y)$" $\left\{ \begin{array}{l} \mathcal{T}' \not\models \text{"never } \ell_2 \wedge odd(y)\text{"} \\ Traces(\mathcal{T}) \subseteq Traces(\mathcal{T}') \end{array} \right.$

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

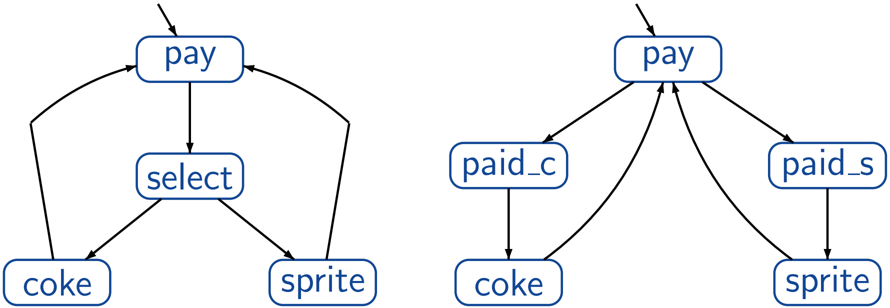
(2) for all LT-properties E: $\mathcal{T}_2 \models E \Longrightarrow \mathcal{T}_1 \models E$

The following statements are equivalent:

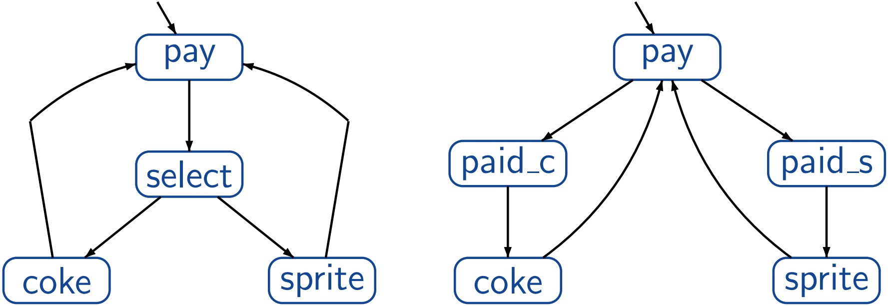
(1) $Traces(\mathcal{T}_1) = Traces(\mathcal{T}_2)$

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

Trace equivalent beverage machines

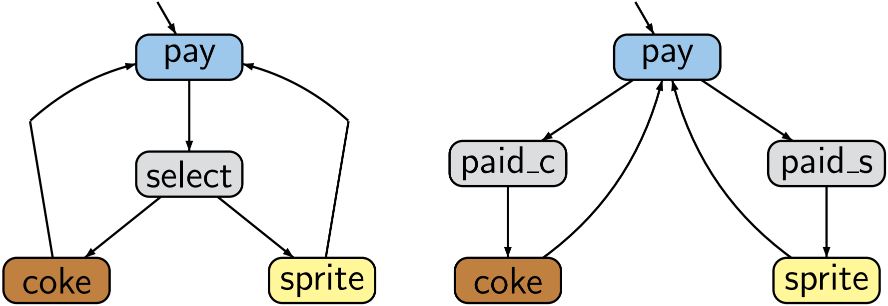


Trace equivalent beverage machines



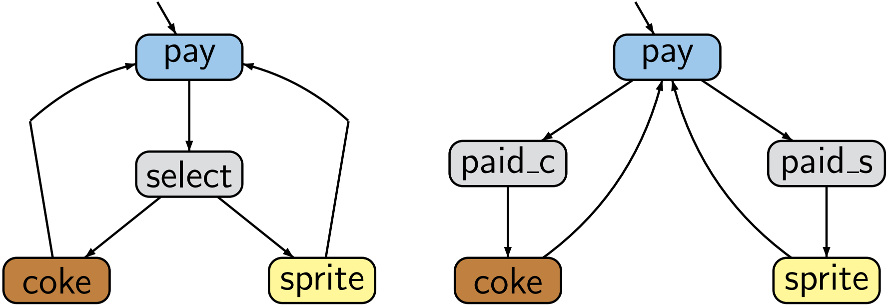
set of atomic propositions $AP = \{pay, coke, sprite\}$

Trace equivalent beverage machines



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

Trace equivalent beverage machines



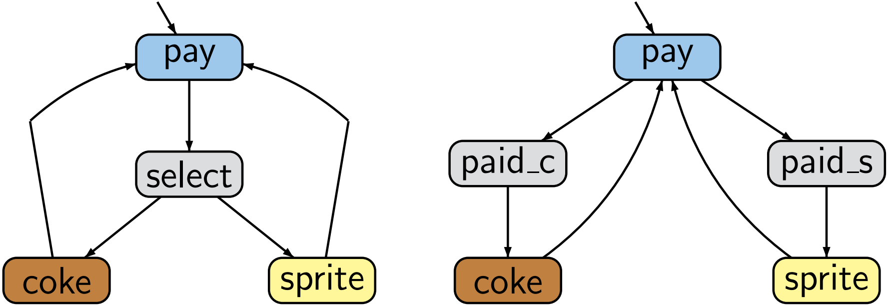
set of atomic propositions $AP = \{pay, coke, sprite\}$

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

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

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

Trace equivalent beverage machines



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

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

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

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