

Introduction

Modelling parallel systems

Linear Time Properties

Regular Properties

Linear Temporal Logic

Computation-Tree Logic

Equivalences and Abstraction

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

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

transition system $\mathcal{T} = (\mathcal{S}, \mathcal{Act}, \longrightarrow, \mathcal{S}_0, \mathcal{AP}, \mathcal{L})$

\mathcal{Act} for modeling interactions/communication

$\mathcal{AP}, \mathcal{L}$ for specifying properties

transition system $\mathcal{T} = (\mathcal{S}, \mathcal{Act}, \longrightarrow, \mathcal{S}_0, \mathcal{AP}, \mathcal{L})$

\mathcal{Act} for modeling interactions/communication
and specifying fairness assumptions

$\mathcal{AP}, \mathcal{L}$ for specifying properties

transition system $\mathcal{T} = (\mathcal{S}, \text{Act}, \longrightarrow, \mathcal{S}_0, \text{AP}, L)$



abstraction from actions

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

- set of nodes = state space \mathcal{S}
- edges = transitions without action label

Act for modeling interactions/communication
and specifying fairness assumptions

AP, L for specifying properties

transition system $\mathcal{T} = (\mathcal{S}, \text{Act}, \longrightarrow, \mathcal{S}_0, \text{AP}, L)$



abstraction from actions

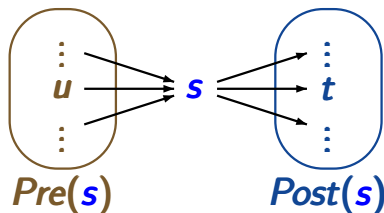
state graph $G_{\mathcal{T}}$

- set of nodes = state space \mathcal{S}
- edges = transitions without action label

use standard notations
for graphs, e.g.,

$$\text{Post}(s) = \{t \in \mathcal{S} : s \rightarrow t\}$$

$$\text{Pre}(s) = \{u \in \mathcal{S} : u \rightarrow s\}$$



execution fragment: sequence of consecutive transitions

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

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

execution fragment: sequence of consecutive transitions

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

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

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

execution fragment: sequence of consecutive transitions

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

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

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

$\pi = s_0 s_1 s_2 \dots$ infinite or $\pi = s_0 s_1 \dots s_n$ finite

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

execution fragment: sequence of consecutive transitions

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

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

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

$\pi = s_0 s_1 s_2 \dots$ infinite or $\pi = s_0 s_1 \dots s_n$ finite

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

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

execution fragment: sequence of consecutive transitions

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

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

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

$\pi = s_0 s_1 s_2 \dots$ infinite or $\pi = s_0 s_1 \dots s_n$ finite

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

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

maximal: if infinite or ending in a terminal state

path fragment: sequence of states

$\pi = s_0 s_1 s_2 \dots$ infinite or $\pi = s_0 s_1 \dots s_n$ finite

s.t. $s_{i+1} \in \text{Post}(s_i)$ 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 $\mathcal{T} \quad \hat{=}$ initial, maximal path fragment

path fragment: sequence of states

$\pi = s_0 s_1 s_2 \dots$ infinite or $\pi = s_0 s_1 \dots s_n$ finite

s.t. $s_{i+1} \in \text{Post}(s_i)$ 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} $\hat{=}$ initial, maximal path fragment

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

path fragment: sequence of states

$\pi = s_0 s_1 s_2 \dots$ infinite or $\pi = s_0 s_1 \dots s_n$ finite

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

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

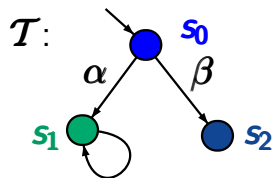
maximal: if infinite or ending in terminal state

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

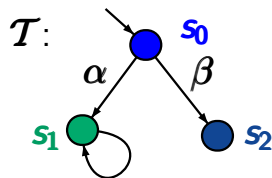
path of state $s \hat{=}$ maximal path fragment starting in state s

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

$\text{Paths}(s) =$ set of all maximal path fragments starting in state s

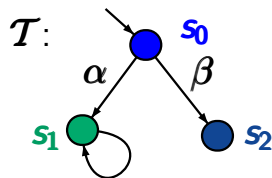


How many **paths** are there in \mathcal{T} ?



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

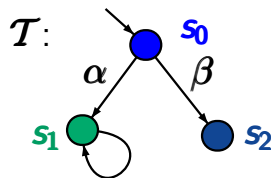
answer: 2, namely $s_0 s_1 s_1 s_1 \dots$ and $s_0 s_2$



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

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

$Paths(s_1)$ = set of all maximal paths fragments starting in s_1
= $\{s_1^\omega\}$ where $s_1^\omega = s_1 s_1 s_1 s_1 \dots$



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

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

$Paths(s_1)$ = set of all maximal paths fragments starting in s_1
= $\{s_1^\omega\}$ where $s_1^\omega = s_1 s_1 s_1 s_1 \dots$

$Paths_{fin}(s_1)$ = set of all finite path fragments starting in s_1
= $\{s_1^n : n \in \mathbb{N}, n \geq 1\}$

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

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

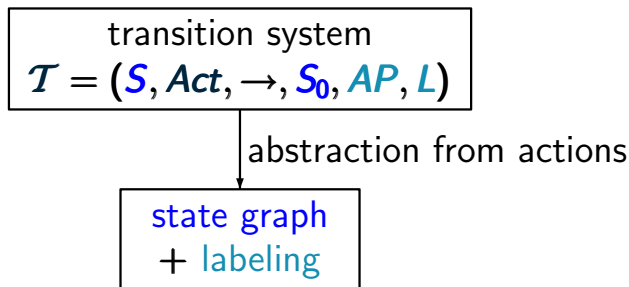
Linear Temporal Logic

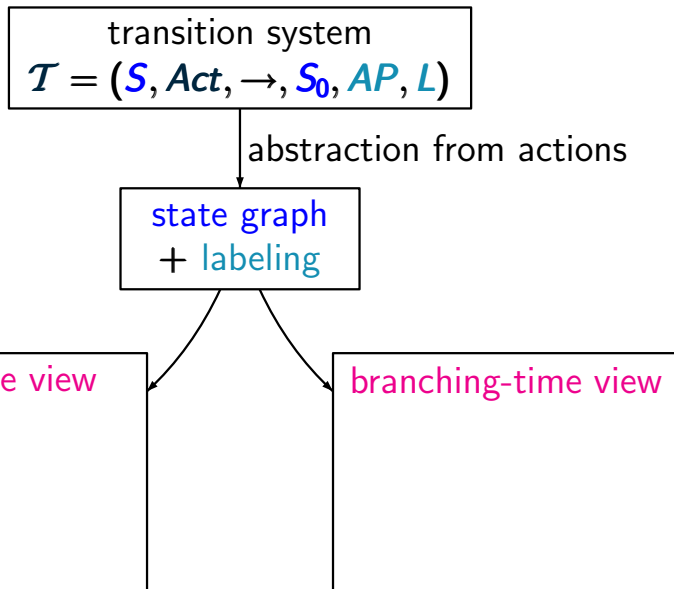
Computation-Tree Logic

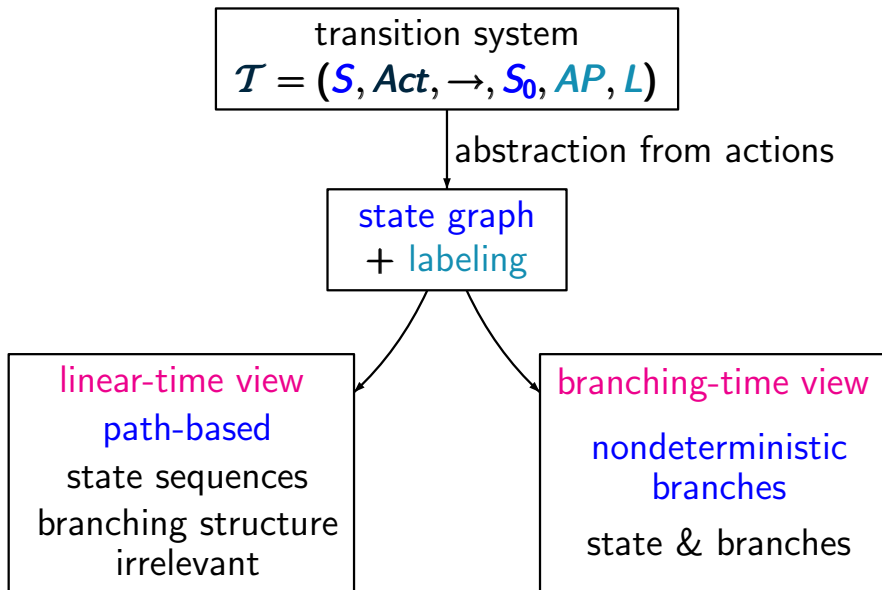
Equivalences and Abstraction

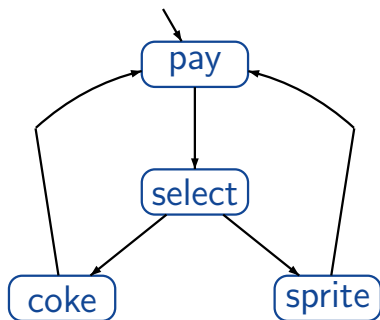
transition system

$$\mathcal{T} = (\mathcal{S}, Act, \rightarrow, \mathcal{S}_0, AP, L)$$









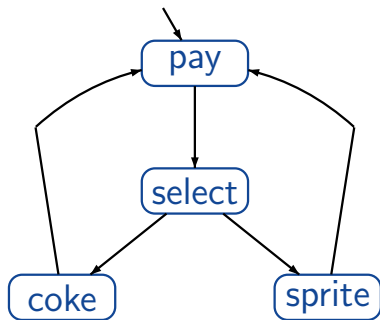
vending machine with

1 coin deposit

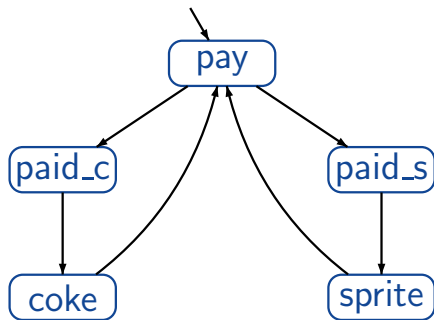
select drink after
having paid

Example: vending machine

LTB2.4-2



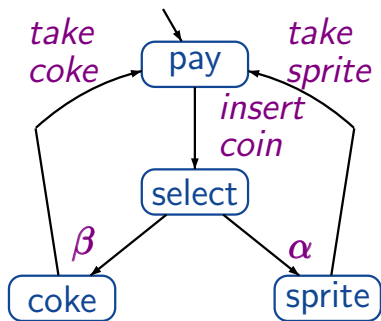
vending machine with
1 coin deposit
select drink after
having paid



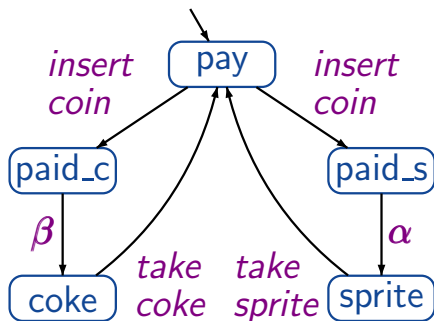
vending machine with
2 coin deposits
select drink by inserting
the coin

Example: vending machine

LTB2.4-2



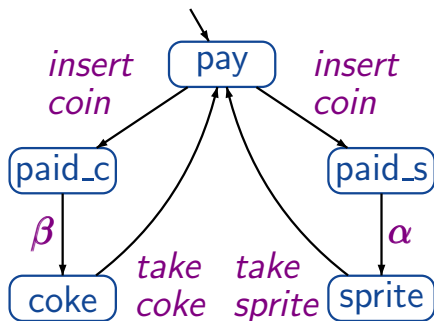
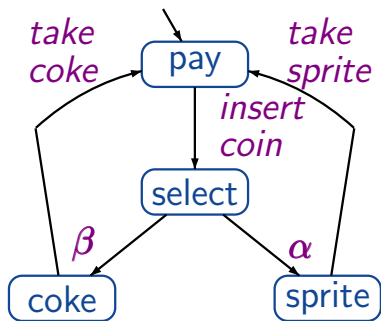
vending machine with
1 coin deposit
select drink after
having paid



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

Example: vending machine

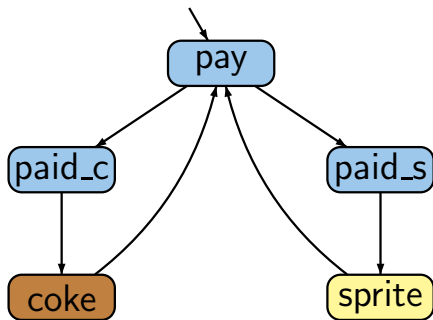
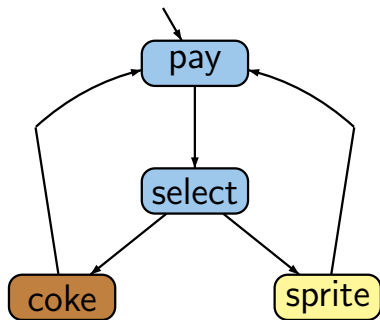
LTB2.4-2



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

Example: vending machine

LTB2.4-2

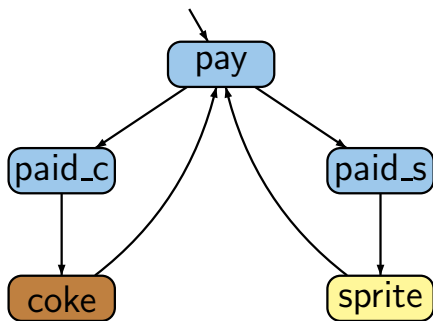
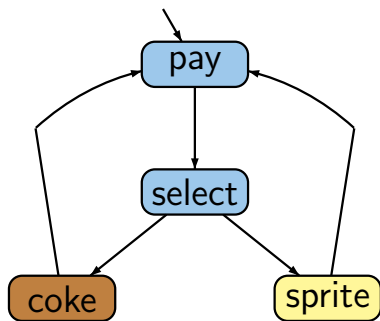


state based view: abstracts from actions and projects onto atomic propositions, e.g. $AP = \{\text{coke}, \text{sprite}\}$

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

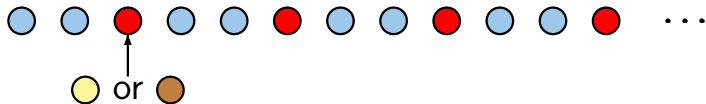
Example: vending machine

LTB2.4-2



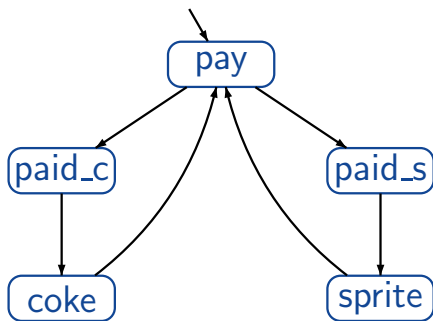
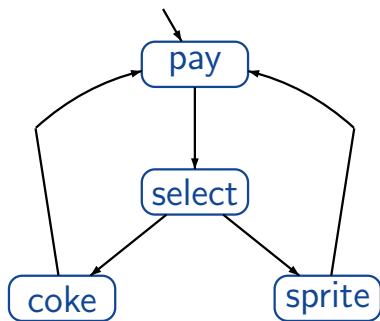
state based view: abstracts from actions and projects onto atomic propositions, e.g. $AP = \{\text{coke}, \text{sprite}\}$

linear time: all observable behaviors are of the form



Example: vending machine

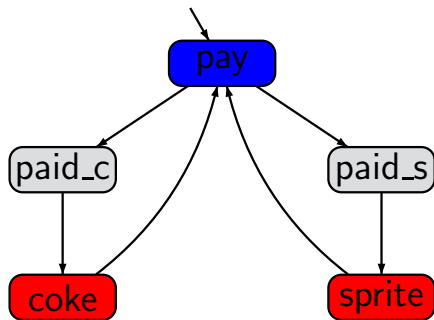
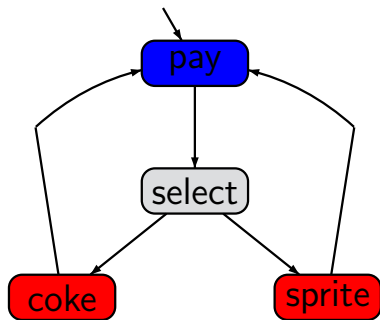
LTB2.4-3



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

Example: vending machine

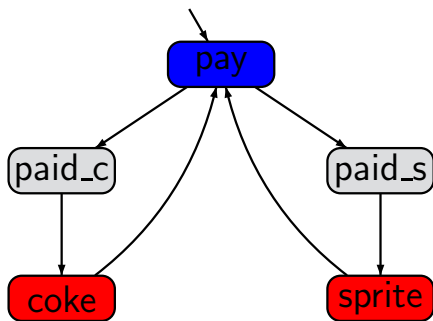
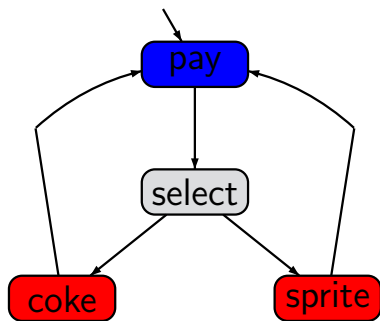
LTB2.4-3



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

Example: vending machine

LTB2.4-3

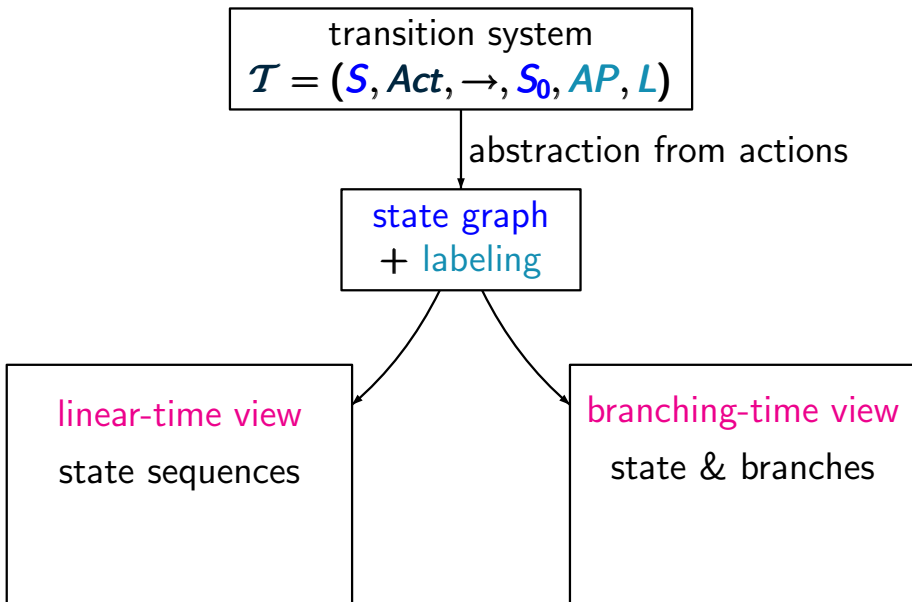


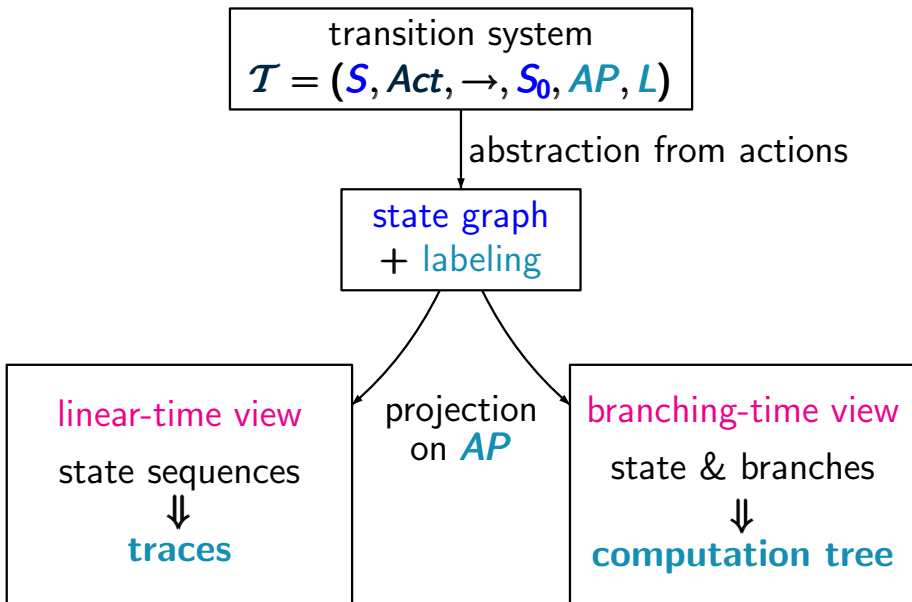
state based view: abstracts from actions and projects on atomic propositions, e.g., $AP = \{\text{pay}, \text{drink}\}$

linear & branching time:

all observable behaviors have the form







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

execution: states + actions

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



paths: sequences of states

$s_0 s_1 s_2 \dots$ infinite or $s_0 s_1 \dots s_n$ finite

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

execution: states + actions

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



paths: sequences of states

$s_0 s_1 s_2 \dots$ infinite or $s_0 s_1 \dots s_n$ finite



traces: sequences of sets of atomic propositions

$L(s_0) L(s_1) L(s_2) \dots$

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

execution: states + actions

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



paths: sequences of states

$s_0 s_1 s_2 \dots$ infinite or $s_0 s_1 \dots s_n$ finite



traces: sequences of sets of atomic propositions

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

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

execution: states + actions

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

paths: sequences of states

$s_0 s_1 s_2 \dots$ infinite or $s_0 s_1 \dots s_n$ finite

traces: sequences of sets of atomic propositions

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

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

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

execution: states + actions

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

paths: sequences of states

$s_0 s_1 s_2 \dots$ infinite or ~~$s_0 s_1 \dots s_n$ finite~~

traces: sequences of sets of atomic propositions

$L(s_0) L(s_1) L(s_2) \dots \in (2^{AP})^\omega \cup \del{(2^{AP})^+}$

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

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

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

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

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

for each reachable terminal state s :

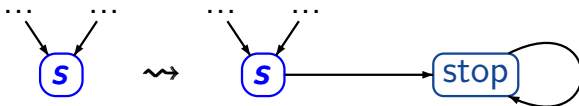
- if s stands for an intended halting configuration then add a transition from s to a trap state:

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

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

for each reachable terminal state s :

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

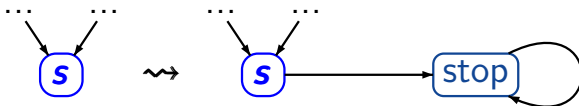


perform standard graph algorithms to compute the reachable fragment of the given TS

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

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

Let \mathcal{T} be a TS

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

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

Let \mathcal{T} be a TS

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

initial, maximal \uparrow path fragment

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

initial, finite \uparrow path fragment

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

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

↑
initial, *infinite* path fragment

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

↑
initial, *finite* path fragment

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

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

↑
initial, *infinite* path fragment

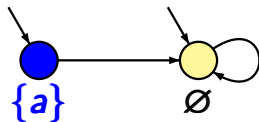
$$\text{Traces}_{fin}(\mathcal{T}) \stackrel{\text{def}}{=} \{ \text{trace}(\hat{\pi}) : \hat{\pi} \in \text{Paths}_{fin}(\mathcal{T}) \} \subseteq (2^{AP})^*$$

↑
initial, *finite* path fragment

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

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

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

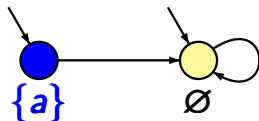


TS \mathcal{T} with a single atomic proposition a

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

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

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



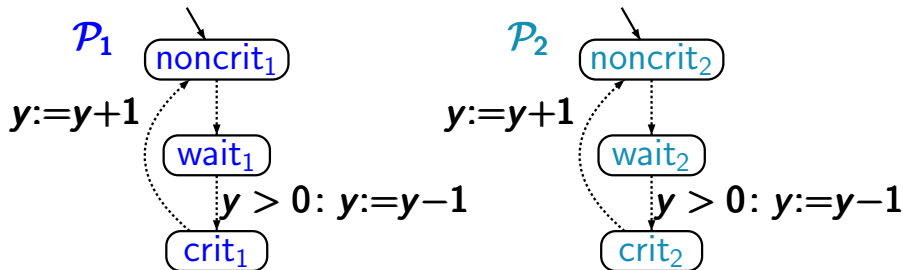
TS \mathcal{T} with a single atomic proposition a

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

$$\text{Traces}_{fin}(\mathcal{T}) = \{ \{a\}\emptyset^n : n \geq 0 \} \cup \{ \emptyset^m : m \geq 1 \}$$

Mutual exclusion with semaphore

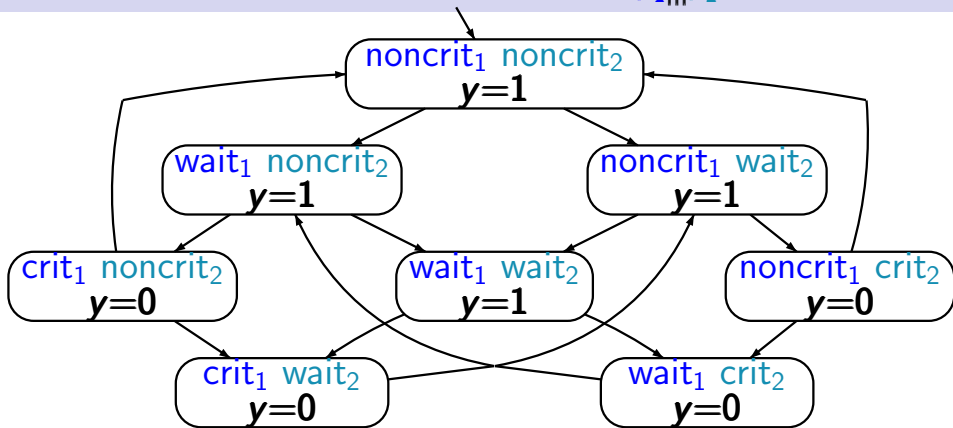
LTB2.4-8



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 $\mathcal{T}_{P_1 ||| P_2}$

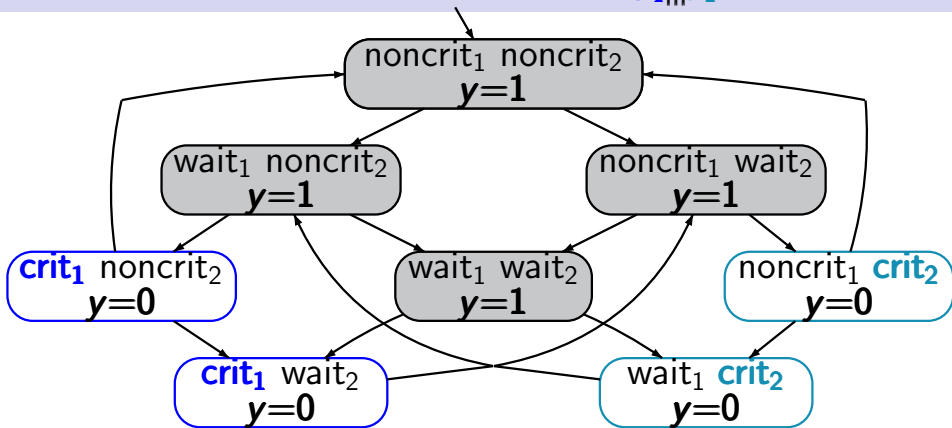
LTB2.4-8



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

Mutual exclusion with semaphore $\mathcal{T}_{P_1 || P_2}$

LITB2.4-8



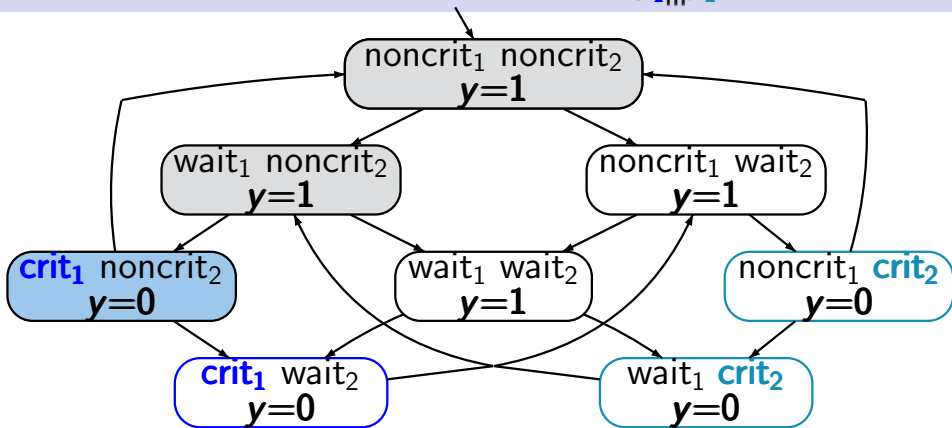
set of atomic propositions $AP = \{\text{crit}_1, \text{crit}_2\}$

e.g., $L(\langle \text{noncrit}_1, \text{noncrit}_2, y=1 \rangle) =$

$L(\langle \text{wait}_1, \text{noncrit}_2, y=1 \rangle) = \emptyset$

Mutual exclusion with semaphore $\mathcal{T}_{P_1 || P_2}$

LITB2.4-8

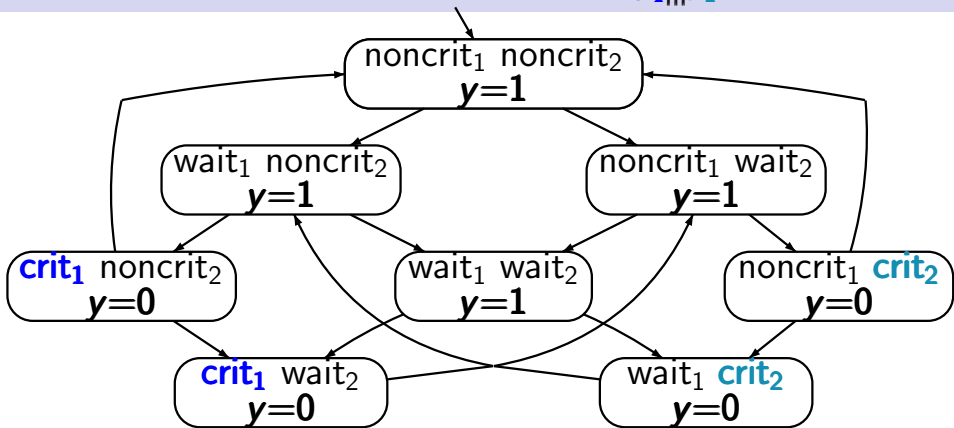


set of atomic propositions $AP = \{\text{crit}_1, \text{crit}_2\}$

traces, e.g., $\emptyset \emptyset \{\text{crit}_1\} \emptyset \emptyset \{\text{crit}_1\} \emptyset \emptyset \{\text{crit}_1\} \dots$

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

LITB2.4-8



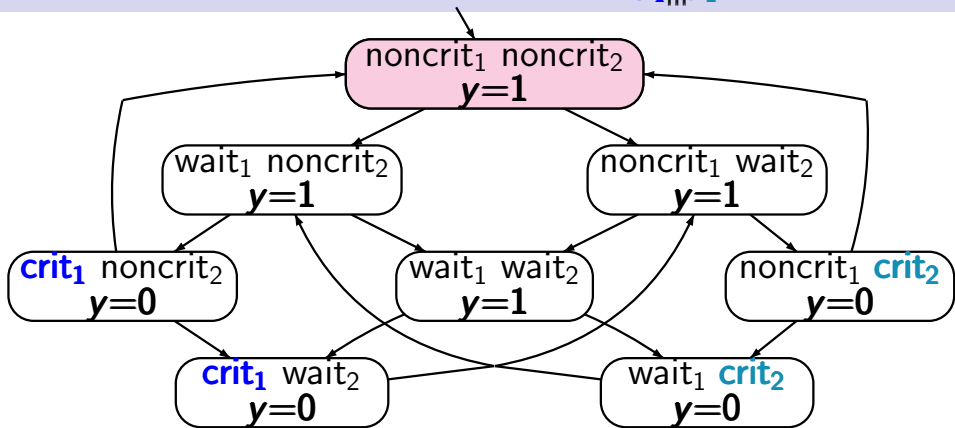
set of atomic propositions $AP = \{\text{crit}_1, \text{crit}_2\}$

traces, e.g., $\emptyset \emptyset \{\text{crit}_1\} \emptyset \emptyset \{\text{crit}_1\} \emptyset \emptyset \{\text{crit}_1\} \dots$

$\emptyset \emptyset \emptyset \{\text{crit}_1\} \emptyset \{\text{crit}_2\} \{\text{crit}_2\} \emptyset \dots$

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

LITB2.4-8



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

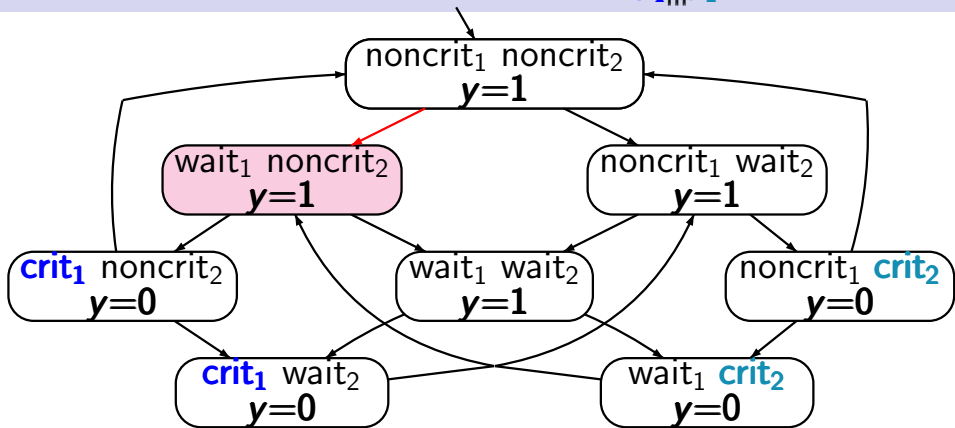
traces, e.g., $\emptyset \emptyset \{\text{crit}_1\} \emptyset \emptyset \{\text{crit}_1\} \emptyset \emptyset \{\text{crit}_1\} \dots$

$\emptyset \emptyset \emptyset \{\text{crit}_1\} \emptyset \{\text{crit}_2\} \{\text{crit}_2\} \emptyset \dots$



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

LITB2.4-8



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

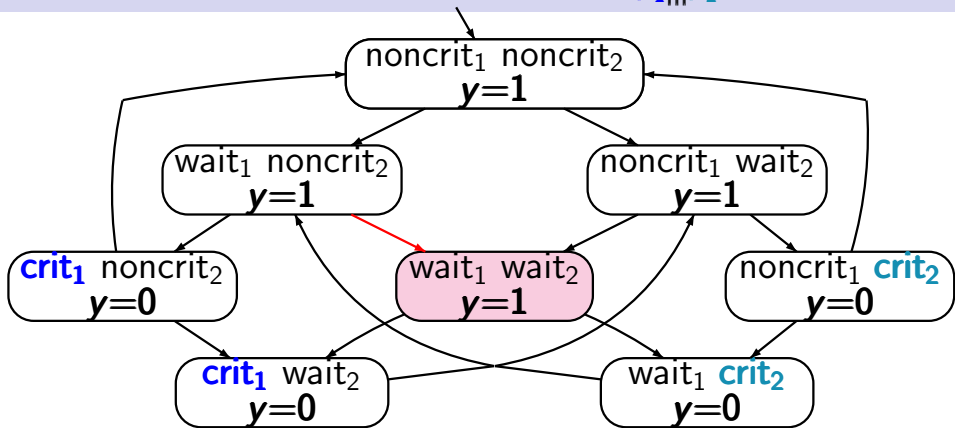
traces, e.g., $\emptyset \emptyset \{\text{crit}_1\} \emptyset \emptyset \{\text{crit}_1\} \emptyset \emptyset \{\text{crit}_1\} \dots$

$\emptyset \emptyset \emptyset \{\text{crit}_1\} \emptyset \{\text{crit}_2\} \{\text{crit}_2\} \emptyset \dots$



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

LITB2.4-8



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

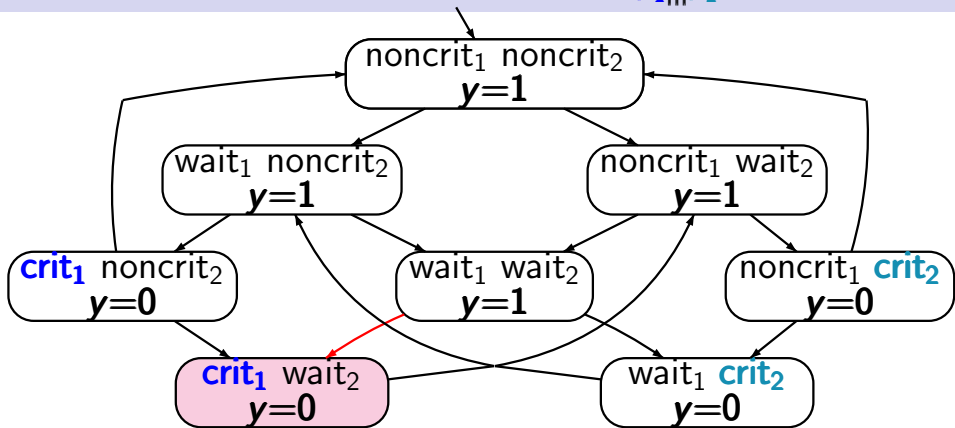
traces, e.g., $\emptyset \emptyset \{\text{crit}_1\} \emptyset \emptyset \{\text{crit}_1\} \emptyset \emptyset \{\text{crit}_1\} \dots$

$\emptyset \emptyset \emptyset \{\text{crit}_1\} \emptyset \{\text{crit}_2\} \{\text{crit}_2\} \emptyset \dots$



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

LITB2.4-8



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

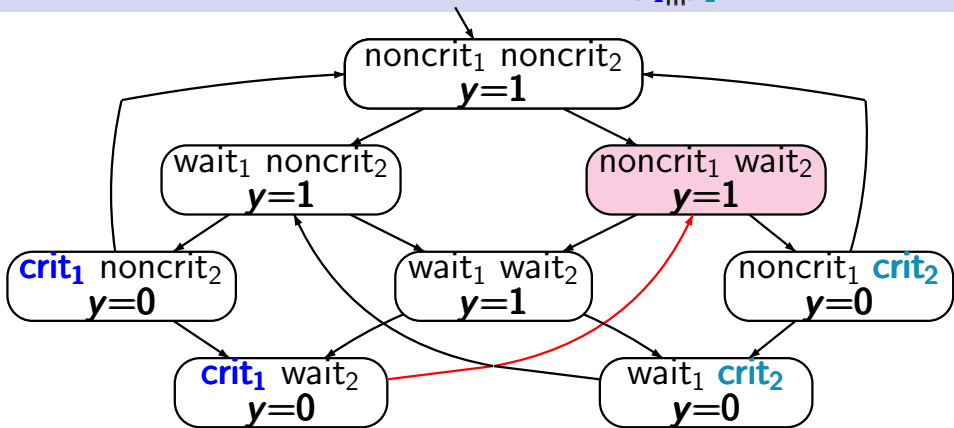
traces, e.g., $\emptyset \emptyset \{\text{crit}_1\} \emptyset \emptyset \{\text{crit}_1\} \emptyset \emptyset \{\text{crit}_1\} \dots$

$\emptyset \emptyset \emptyset \{\text{crit}_1\} \emptyset \{\text{crit}_2\} \{\text{crit}_2\} \emptyset \dots$



Mutual exclusion with semaphore $\mathcal{T}_{P_1 || P_2}$

LITB2.4-8



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

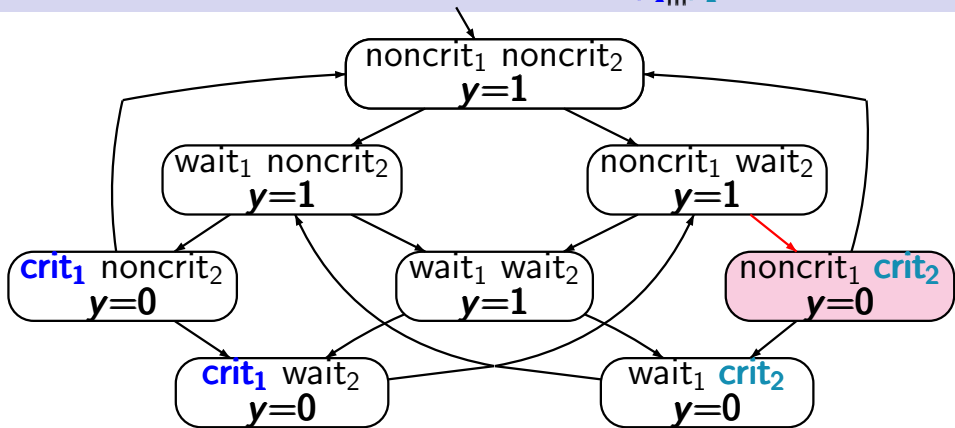
traces, e.g., $\emptyset \emptyset \{\text{crit}_1\} \emptyset \emptyset \{\text{crit}_1\} \emptyset \emptyset \{\text{crit}_1\} \dots$

$\emptyset \emptyset \emptyset \{\text{crit}_1\} \emptyset \{\text{crit}_2\} \{\text{crit}_2\} \emptyset \dots$



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

LITB2.4-8



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

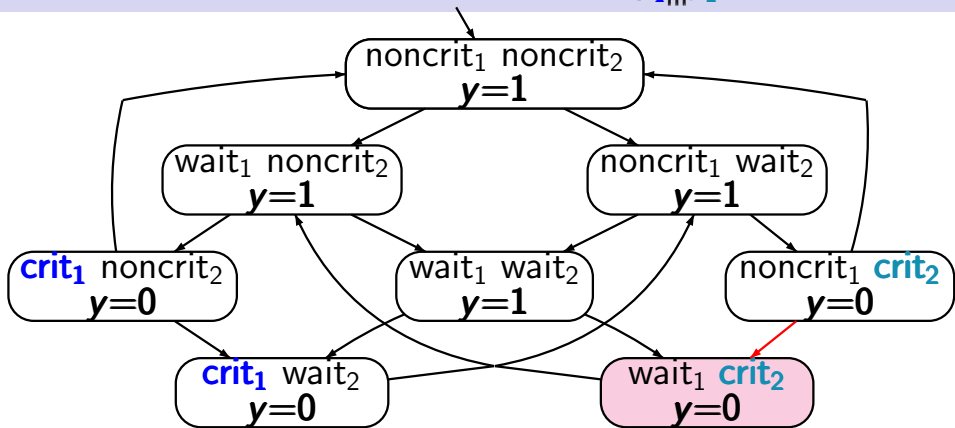
traces, e.g., $\emptyset \emptyset \{\text{crit}_1\} \emptyset \emptyset \{\text{crit}_1\} \emptyset \emptyset \{\text{crit}_1\} \dots$

$\emptyset \emptyset \emptyset \{\text{crit}_1\} \emptyset \{\text{crit}_2\} \{\text{crit}_2\} \emptyset \dots$



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

LITB2.4-8



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

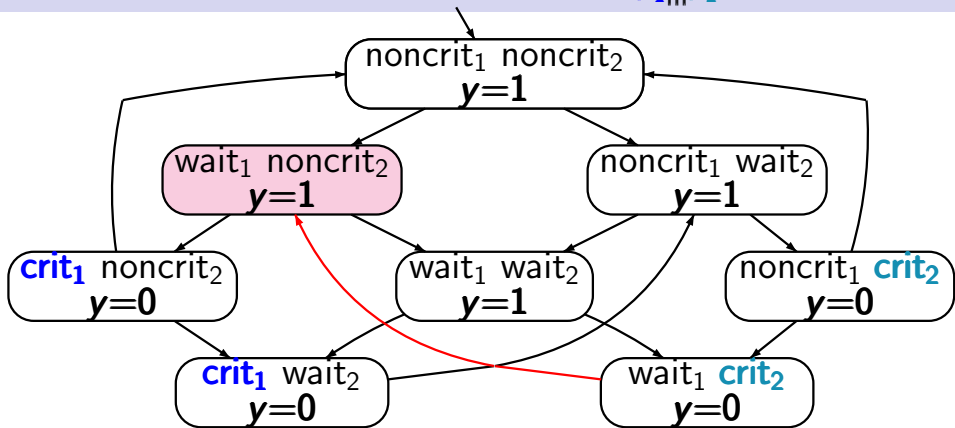
traces, e.g., $\emptyset \emptyset \{\text{crit}_1\} \emptyset \emptyset \{\text{crit}_1\} \emptyset \emptyset \{\text{crit}_1\} \dots$

$\emptyset \emptyset \emptyset \{\text{crit}_1\} \emptyset \{\text{crit}_2\} \{\text{crit}_2\} \emptyset \dots$



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

LITB2.4-8

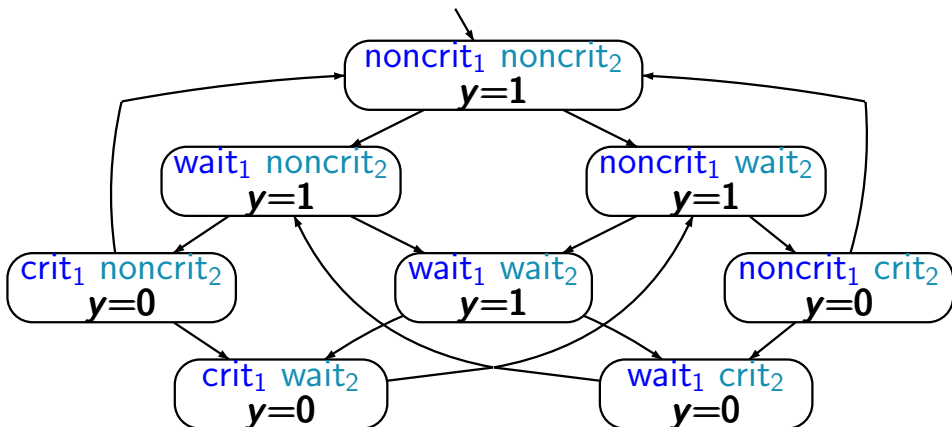


set of atomic propositions $AP = \{\text{crit}_1, \text{crit}_2\}$

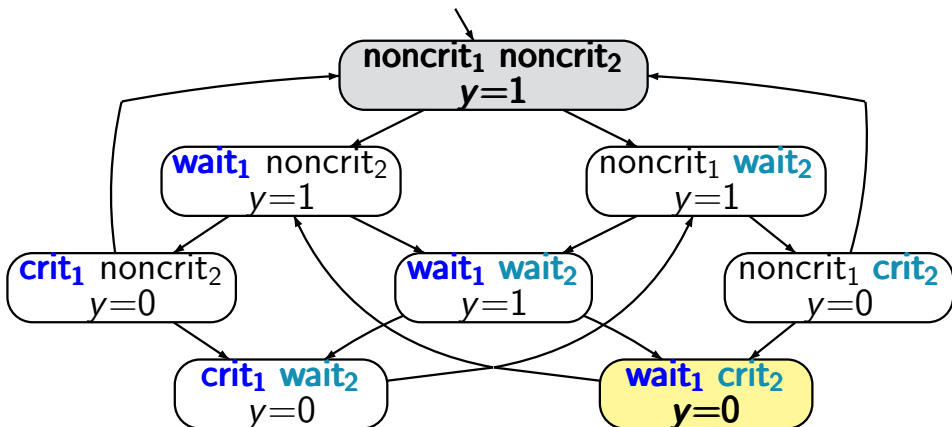
traces, e.g., $\emptyset \emptyset \{\text{crit}_1\} \emptyset \emptyset \{\text{crit}_1\} \emptyset \emptyset \{\text{crit}_1\} \dots$

$\emptyset \emptyset \emptyset \{\text{crit}_1\} \emptyset \{\text{crit}_2\} \{\text{crit}_2\} \emptyset \dots$





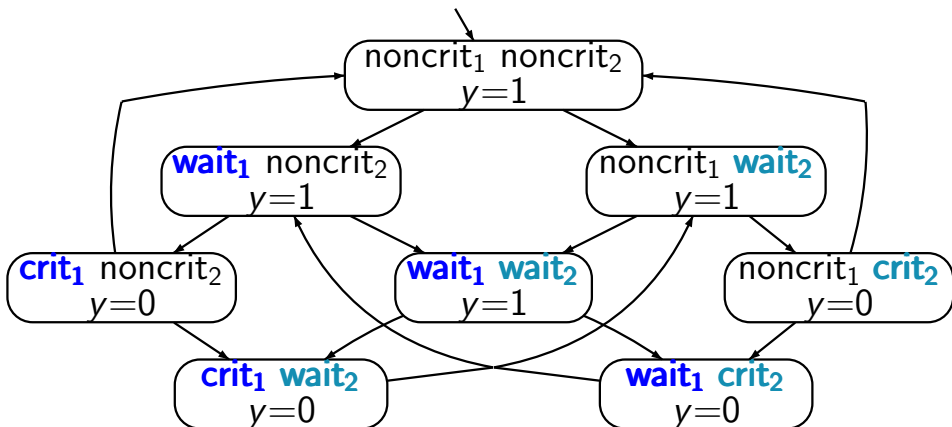
set of propositions $AP = \{\text{wait}_1, \text{crit}_1, \text{wait}_2, \text{crit}_2\}$



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

e.g., $L(\langle \text{noncrit}_1, \text{noncrit}_2, y=1 \rangle) = \emptyset$

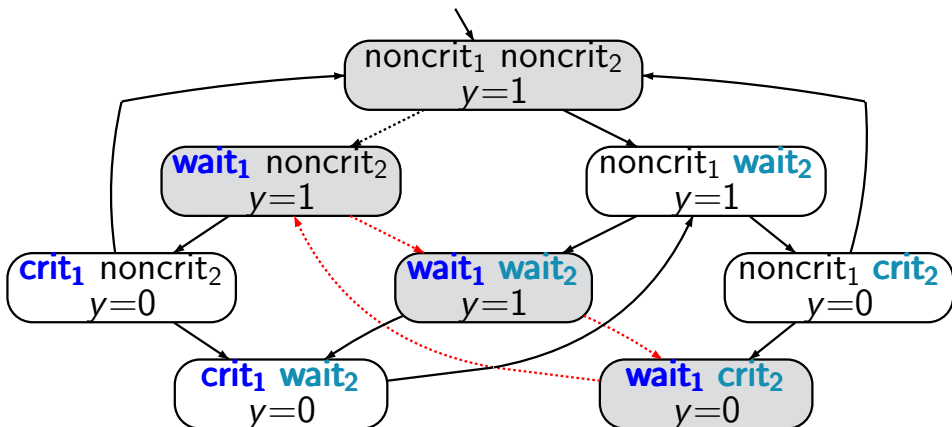
$L(\langle \text{wait}_1, \text{crit}_2, y=1 \rangle) = \{\text{wait}_1, \text{crit}_2\}$



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

traces, e.g.,

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



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

traces, e.g.,

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

Introduction

Modelling parallel systems

Linear Time Properties

state-based and linear time view

definition of linear time properties ←

invariants and safety

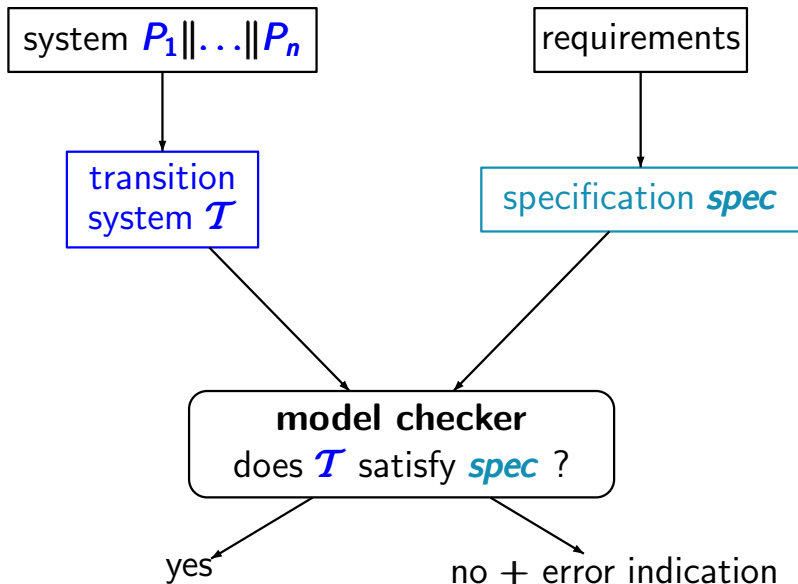
liveness and fairness

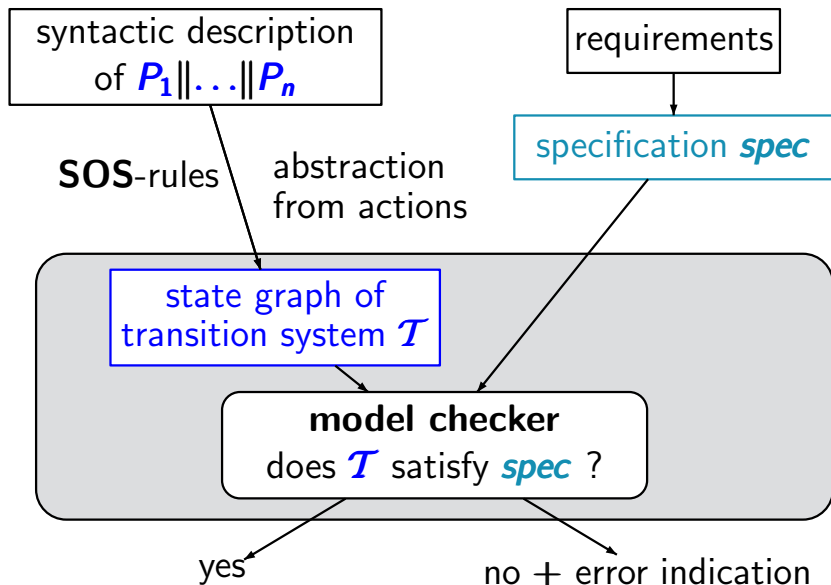
Regular Properties

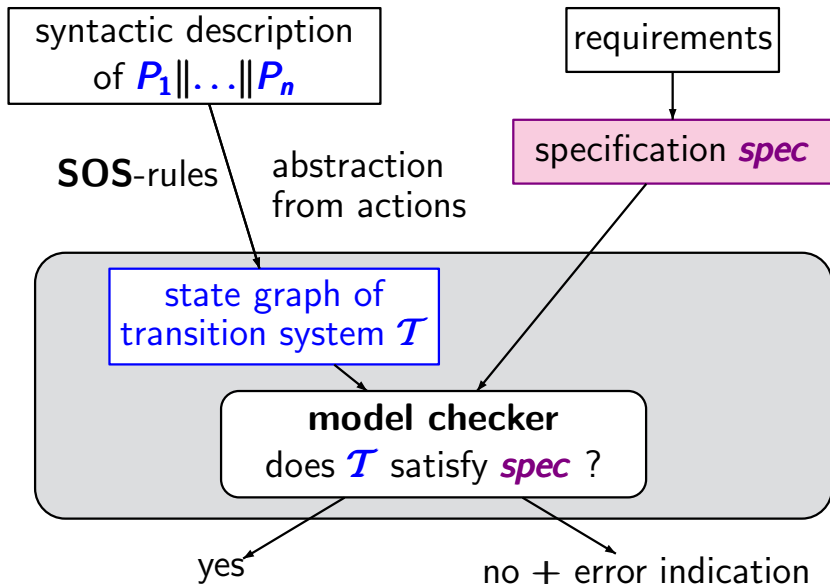
Linear Temporal Logic

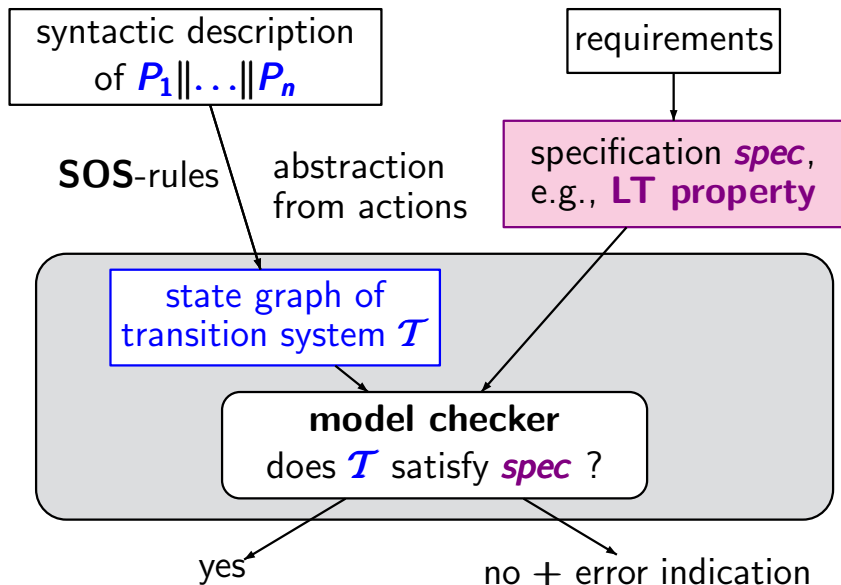
Computation-Tree Logic

Equivalences and Abstraction









for TS over AP without terminal states

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

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

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

E.g., for mutual exclusion problems and
 $AP = \{\text{crit}_1, \text{crit}_2, \dots\}$

safety:

$MUTEX =$ set of all infinite words $A_0 A_1 A_2 \dots$
over 2^{AP} such that for all $i \in \mathbb{N}$:
 $\text{crit}_1 \notin A_i$ or $\text{crit}_2 \notin A_i$

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

safety:

set of all infinite words $A_0 A_1 A_2 \dots$
 $MUTEX =$ over 2^{AP} such that for all $i \in \mathbb{N}$:
 $\text{crit}_1 \notin A_i$ or $\text{crit}_2 \notin A_i$

$$\emptyset \{\text{wait}_1\} \{\text{crit}_1\} \emptyset \{\text{wait}_1\} \{\text{crit}_1\} \dots \in MUTEX$$

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

safety:

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

$$\emptyset \{\text{wait}_1\} \{\text{crit}_1\} \emptyset \{\text{wait}_1\} \{\text{crit}_1\} \dots \in \text{MUTEX}$$

$$\emptyset \{\text{wait}_1\} \{\text{crit}_1\} \{\text{crit}_1, \text{wait}_2\} \{\text{crit}_1, \text{crit}_2\} \dots \notin \text{MUTEX}$$

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

safety:

set of all infinite words $A_0 A_1 A_2 \dots$
 $MUTEX =$ over 2^{AP} such that for all $i \in \mathbb{N}$:
 $\text{crit}_1 \notin A_i$ or $\text{crit}_2 \notin A_i$

$\emptyset \{\text{wait}_1\} \{\text{crit}_1\} \emptyset \{\text{wait}_1\} \{\text{crit}_1\} \dots \in MUTEX$

$\emptyset \{\text{wait}_1\} \{\text{crit}_1\} \{\text{crit}_1, \text{wait}_2\} \{\text{crit}_1, \text{crit}_2\} \dots \notin MUTEX$

$\emptyset \emptyset \{\text{wait}_1, \text{crit}_1, \text{crit}_2\} \dots \notin MUTEX$

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

safety:

$$\begin{aligned} \text{MUTEX} = & \text{ set of all infinite words } A_0 A_1 A_2 \dots \\ & \text{ over } 2^{AP} \text{ such that for all } i \in \mathbb{N}: \\ & \text{crit}_1 \notin A_i \text{ or } \text{crit}_2 \notin A_i \end{aligned}$$

liveness (starvation freedom):

$$\begin{aligned} \text{LIVE} = & \text{ set of all infinite words } A_0 A_1 A_2 \dots \text{ s.t.} \\ & \exists^{\infty} i \in \mathbb{N}. \text{wait}_1 \in A_i \implies \exists^{\infty} i \in \mathbb{N}. \text{crit}_1 \in A_i \\ & \wedge \exists^{\infty} i \in \mathbb{N}. \text{wait}_2 \in A_i \implies \exists^{\infty} i \in \mathbb{N}. \text{crit}_2 \in A_i \end{aligned}$$

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

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

Satisfaction relation \models for TS:

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

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

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

Satisfaction relation \models for TS and states:

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

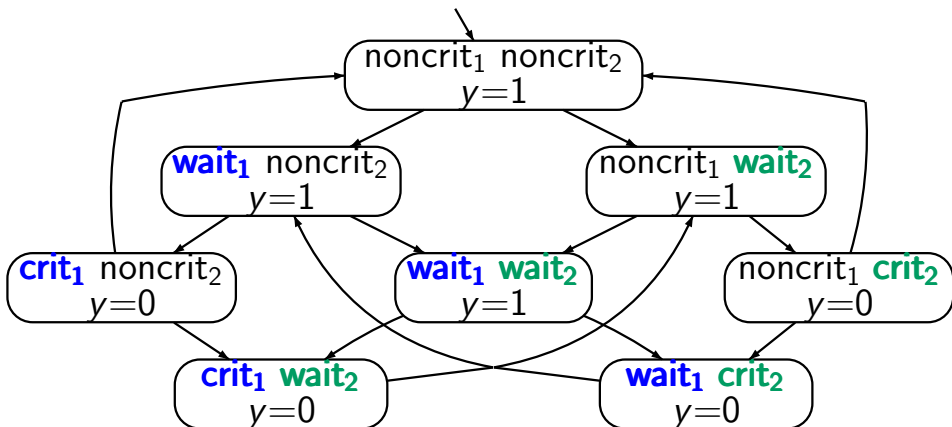
$$\mathcal{T} \models E \quad \text{iff} \quad \text{Traces}(\mathcal{T}) \subseteq E$$

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

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

Mutual exclusion with semaphore

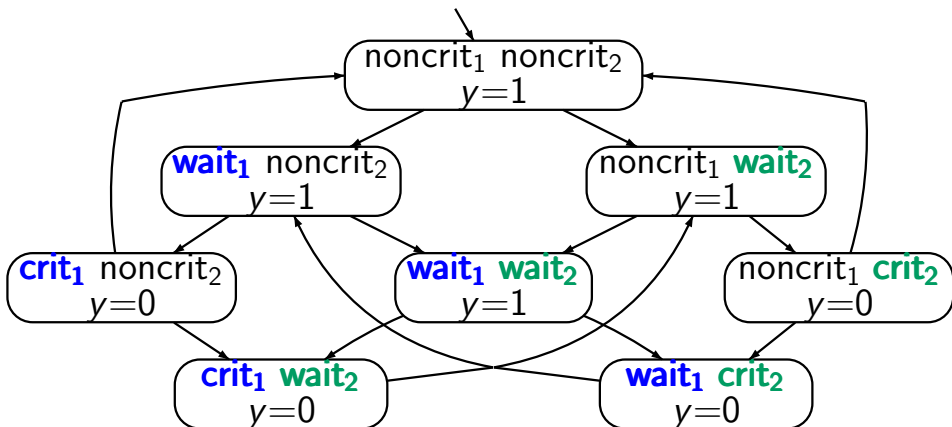
LTB2.4-16



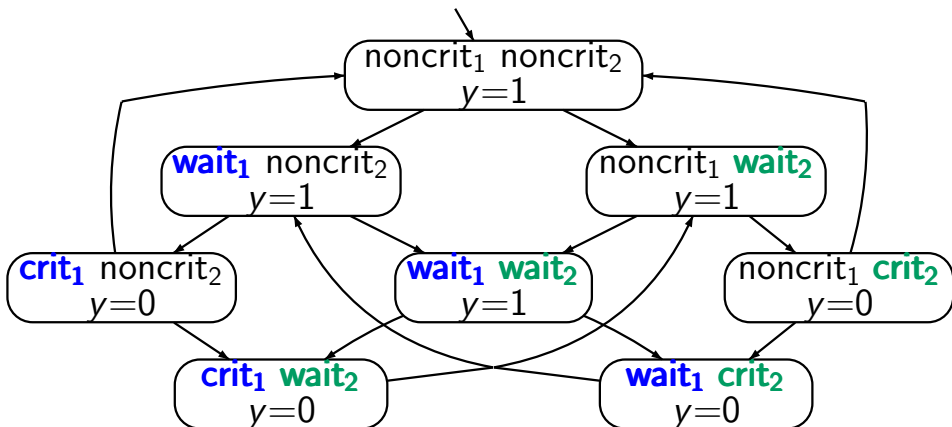
$\mathcal{T}_{Sem} \models \text{MUTEX}$

Mutual exclusion with semaphore

LTB2.4-16



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

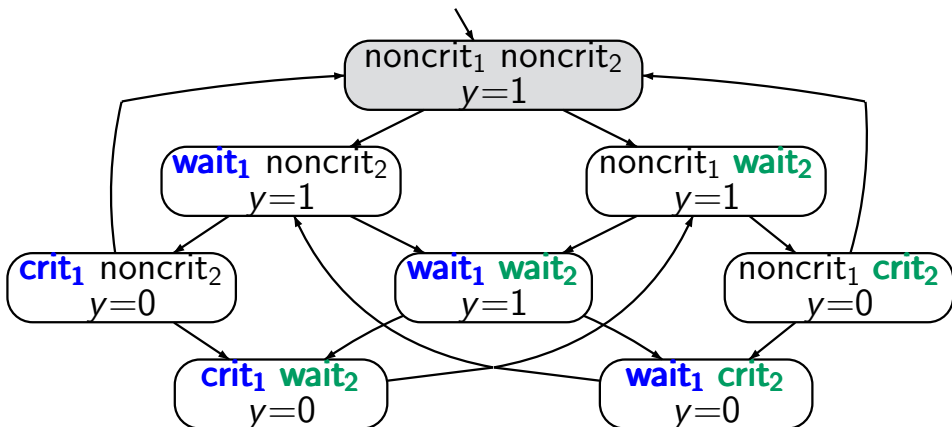


$\mathcal{T}_{Sem} \models \text{MUTEX}, \quad \mathcal{T}_{Sem} \not\models \text{LIVE}$

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

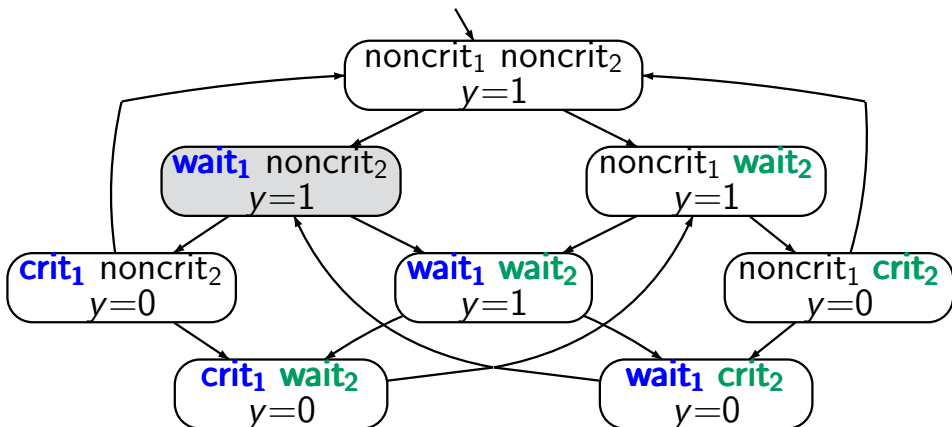
Mutual exclusion with semaphore

LTB2.4-16



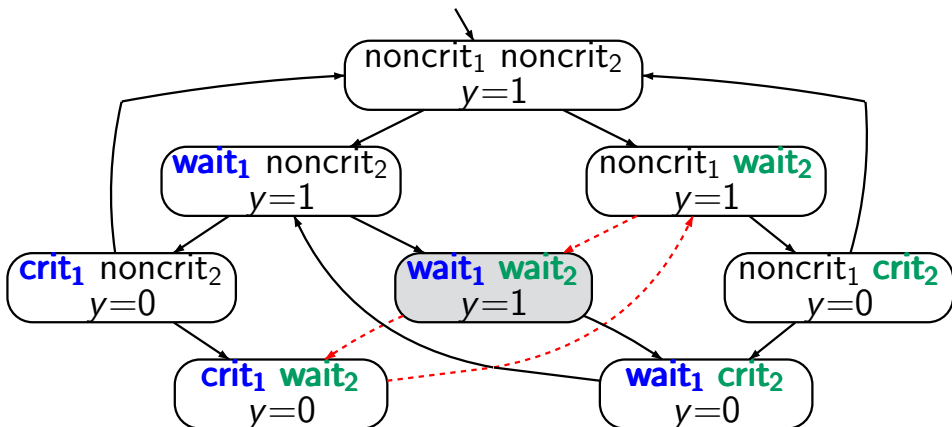
$\mathcal{T}_{Sem} \models \text{MUTEX}, \quad \mathcal{T}_{Sem} \not\models \text{LIVE}$

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



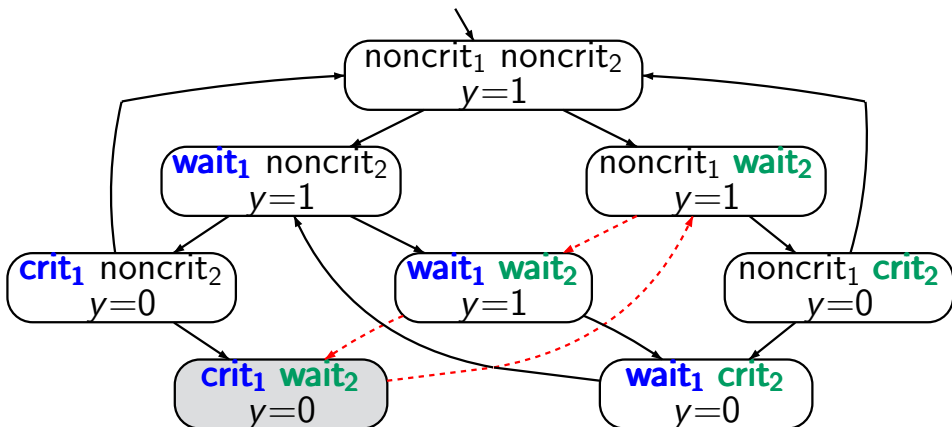
$\mathcal{T}_{Sem} \models \text{MUTEX}, \quad \mathcal{T}_{Sem} \not\models \text{LIVE}$

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



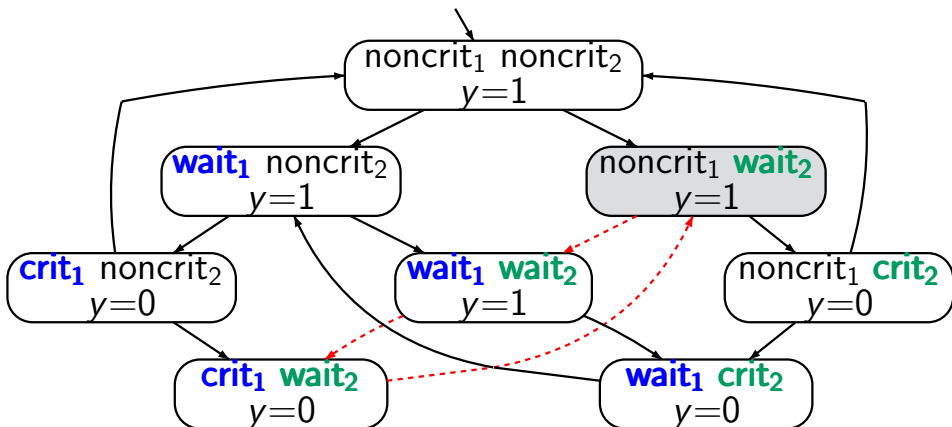
$\mathcal{T}_{Sem} \models \text{MUTEX}, \quad \mathcal{T}_{Sem} \not\models \text{LIVE}$

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



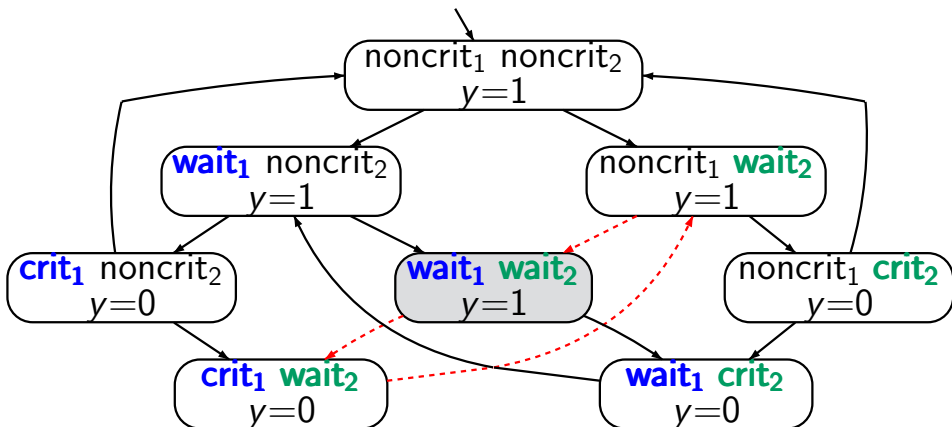
$\mathcal{T}_{Sem} \models \text{MUTEX}, \quad \mathcal{T}_{Sem} \not\models \text{LIVE}$

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



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

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



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

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

Peterson's mutual exclusion algorithm

LTB2.4-17

Peterson's mutual exclusion algorithm

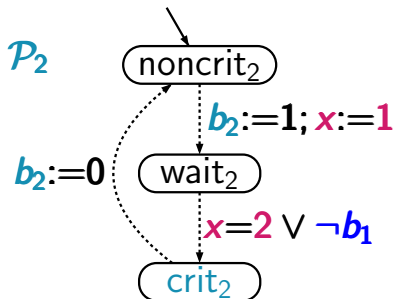
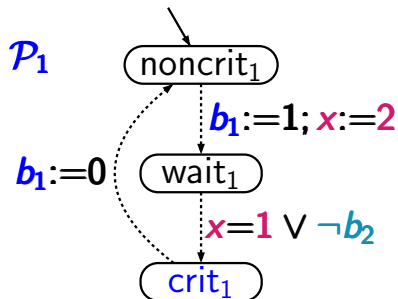
LITB2.4-17

for competing processes P_1 and P_2 ,
using three additional shared variables
 $b_1, b_2 \in \{0, 1\}$, $x \in \{1, 2\}$

Peterson's mutual exclusion algorithm

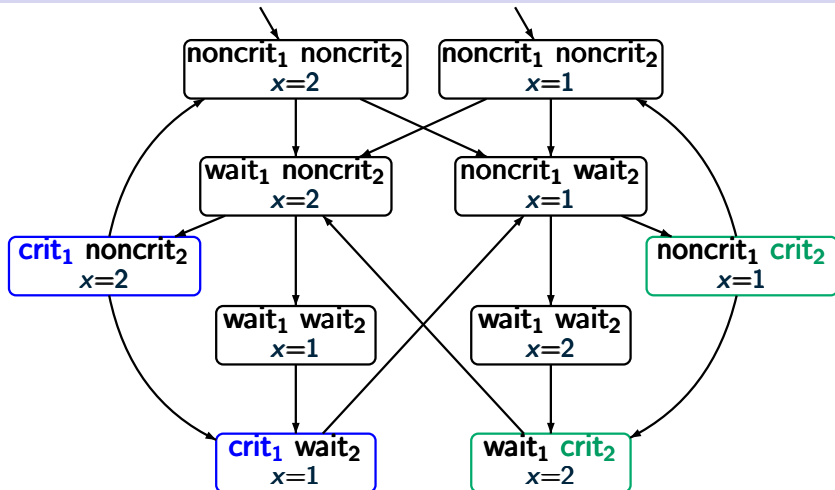
LITB2.4-17

for competing processes P_1 and P_2 ,
using three additional shared variables
 $b_1, b_2 \in \{0, 1\}$, $x \in \{1, 2\}$



Peterson's mutual exclusion algorithm

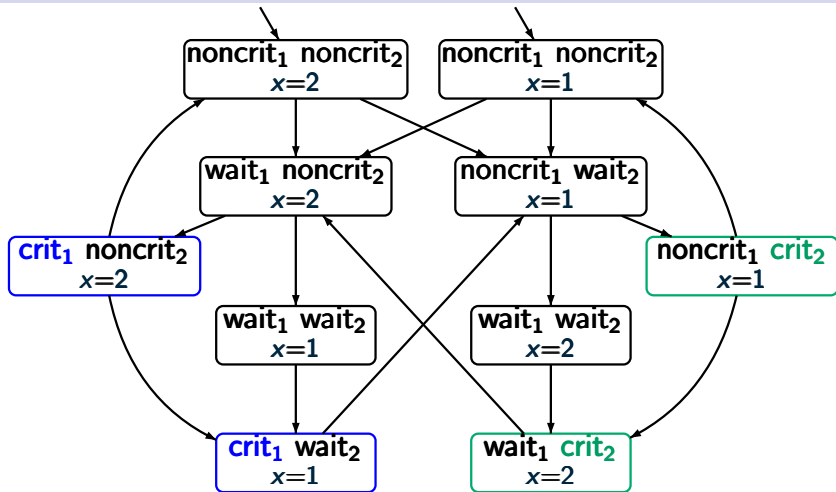
LITB2.4-17



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

Peterson's mutual exclusion algorithm

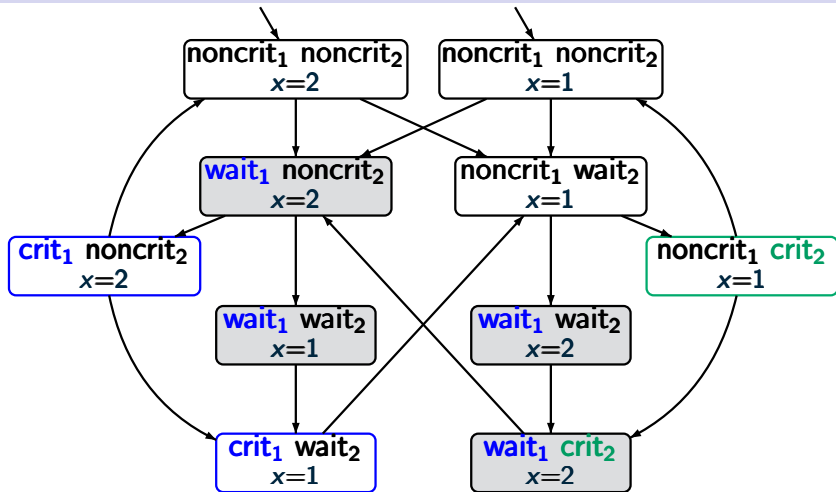
LTB2.4-17



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

Peterson's mutual exclusion algorithm

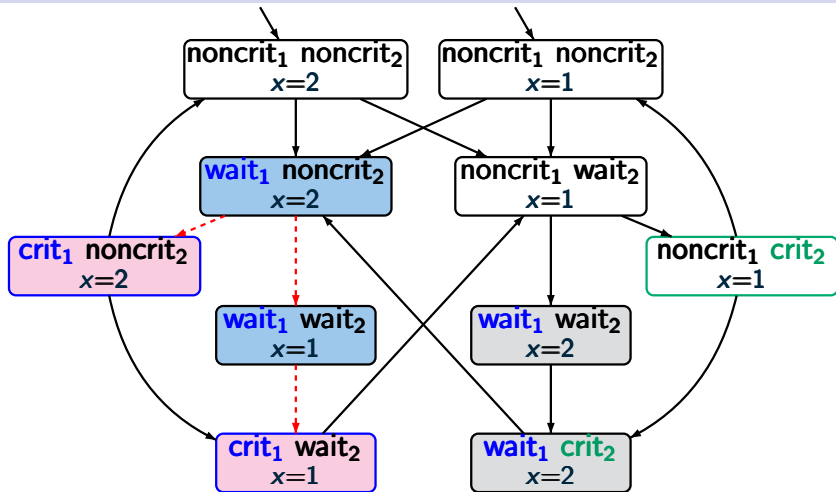
LTB2.4-17



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

Peterson's mutual exclusion algorithm

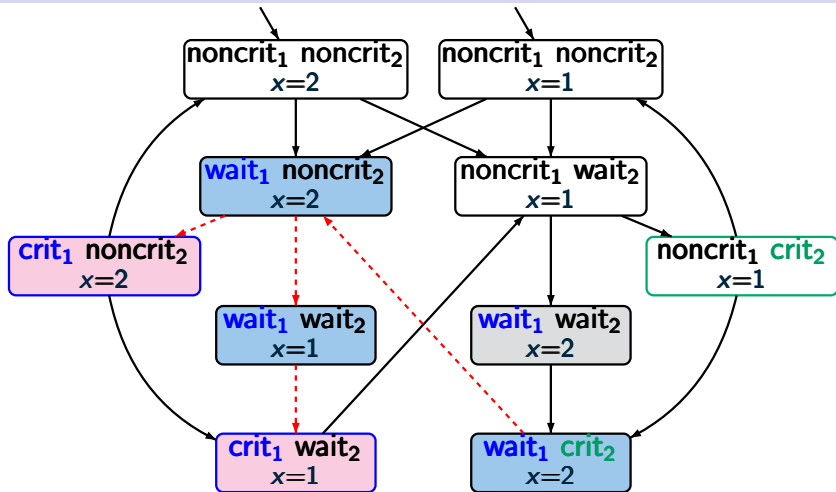
LTB2.4-17



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

Peterson's mutual exclusion algorithm

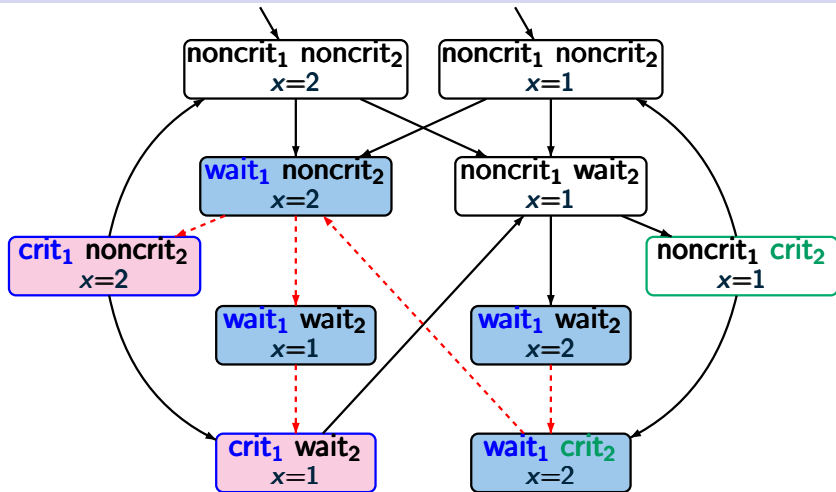
LTB2.4-17



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

Peterson's mutual exclusion algorithm

LTB2.4-17



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

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 $Traces(\mathcal{T}) \subseteq E$.

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

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

Consequence of these definitions:

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

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

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

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

Consequence of these definitions:

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

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

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

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 $\mathcal{T}_1 \models E$

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

If T_1 and T_2 are TS over AP then the following statements are equivalent:

- (1) $Traces(T_1) \subseteq Traces(T_2)$
- (2) for all LT-properties E over AP :
whenever $T_2 \models E$ then $T_1 \models E$

(1) \implies (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 $Traces(\mathcal{T}) \subseteq E$.

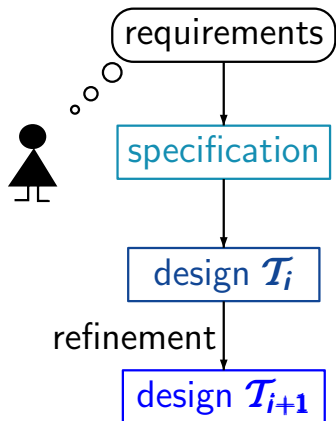
If \mathcal{T}_1 and \mathcal{T}_2 are TS over AP then the following statements are equivalent:

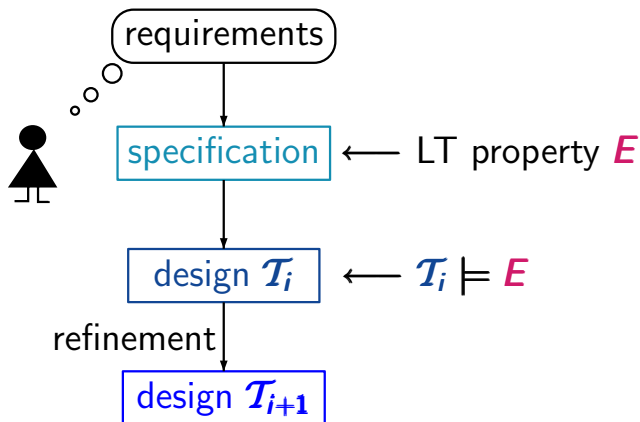
- (1) $Traces(\mathcal{T}_1) \subseteq Traces(\mathcal{T}_2)$
- (2) for all LT-properties E over AP :
whenever $\mathcal{T}_2 \models E$ then $\mathcal{T}_1 \models E$

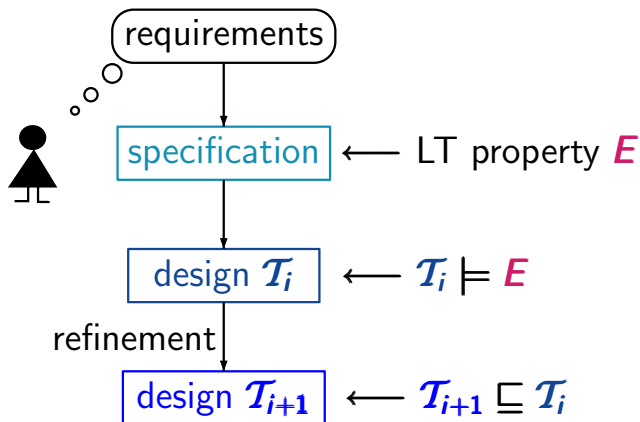
(2) \implies (1): consider $E = Traces(\mathcal{T}_2)$

Trace inclusion appears naturally

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

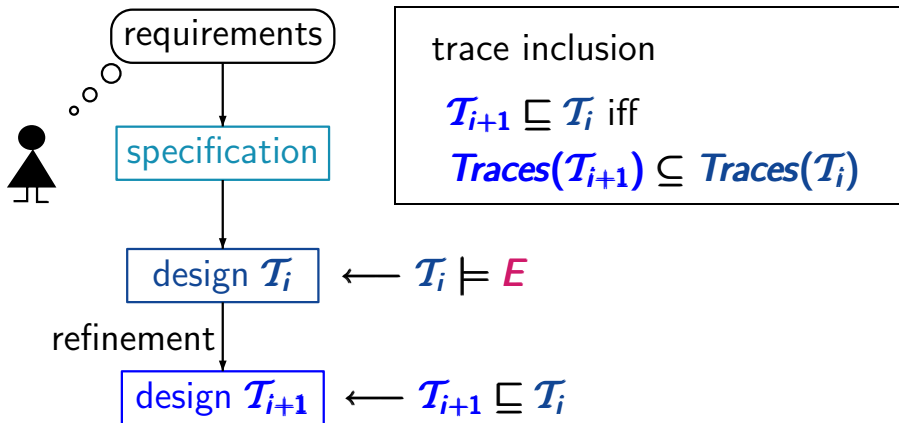






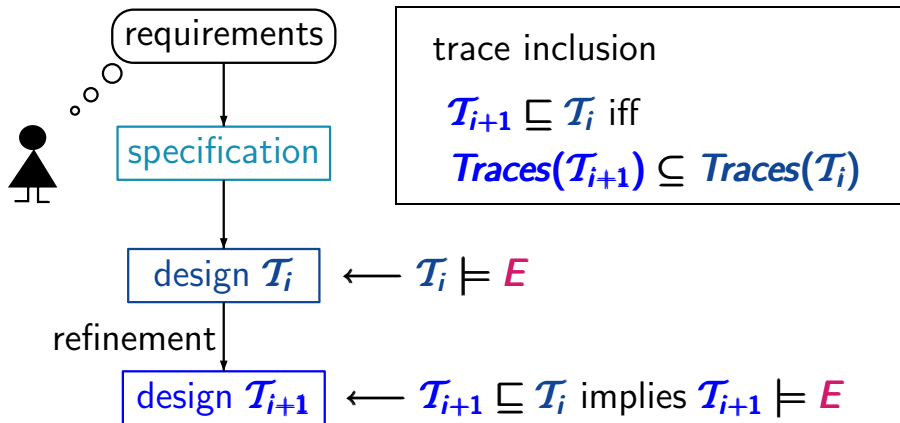
implementation/refinement relation \sqsubseteq :

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

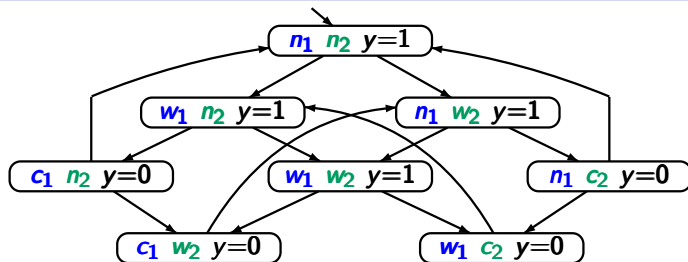


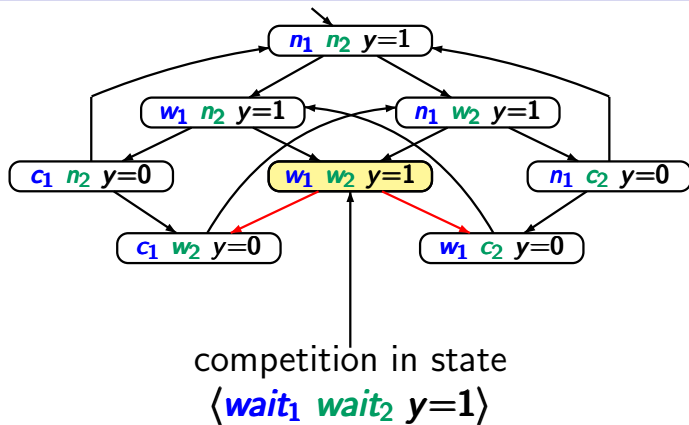
implementation/refinement relation \sqsubseteq :

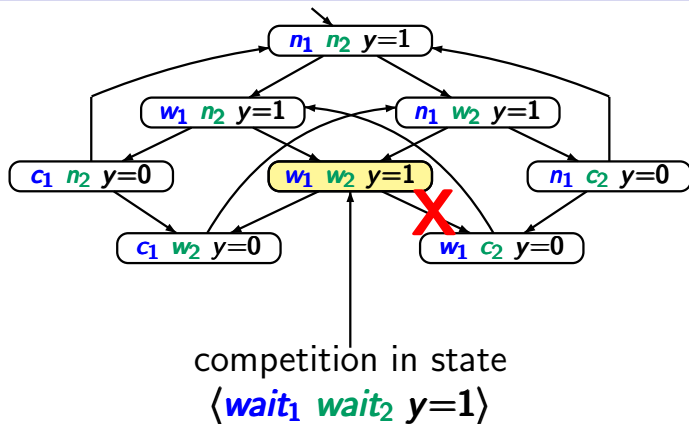
$\mathcal{T}_{i+1} \sqsubseteq \mathcal{T}_i \text{ iff } \text{"}\mathcal{T}_{i+1} \text{ correctly implements } \mathcal{T}_i\text{"}$

Mutual exclusion with semaphore

LTB2.4-20



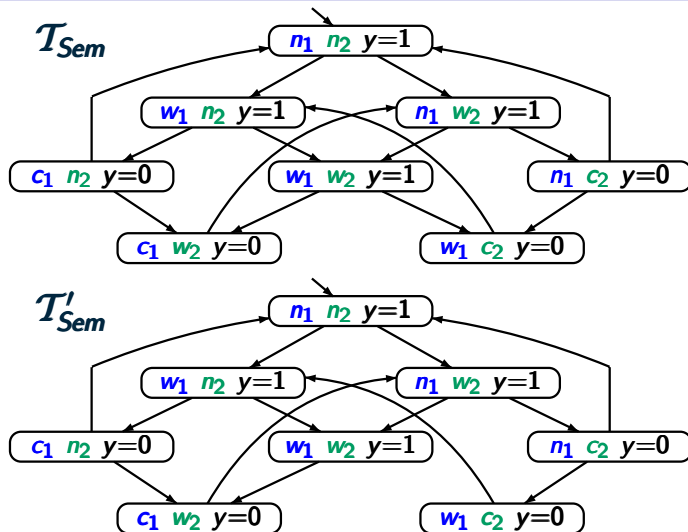




resolve the **nondeterminism** by giving
priority to process P_1

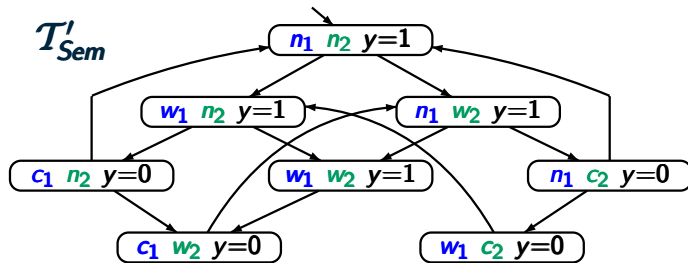
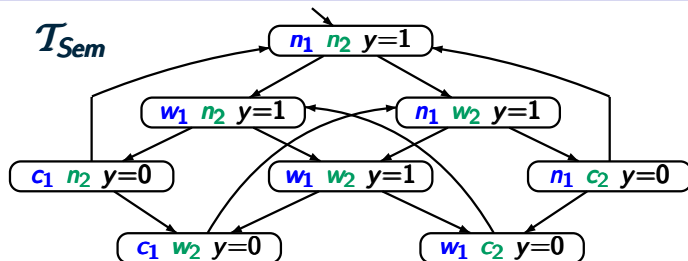
Mutual exclusion with semaphore

LTB2.4-20

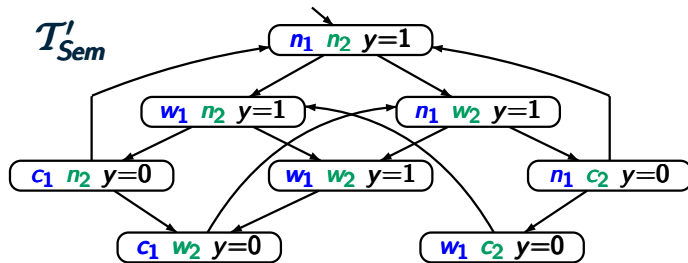
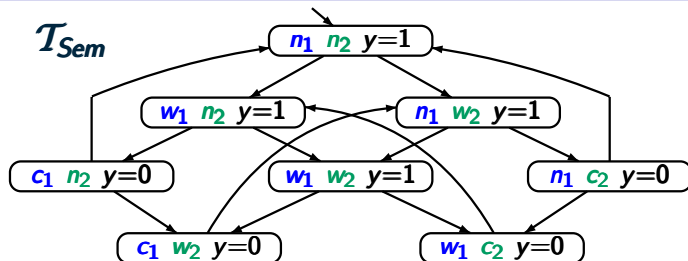


Mutual exclusion with semaphore

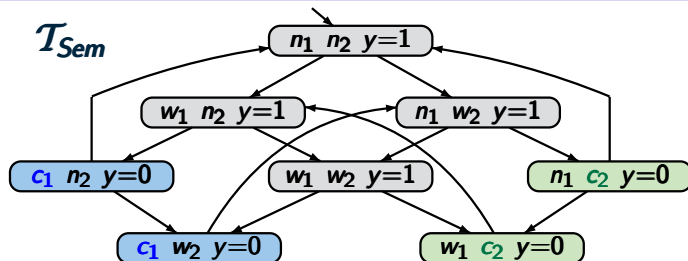
LTB2.4-20



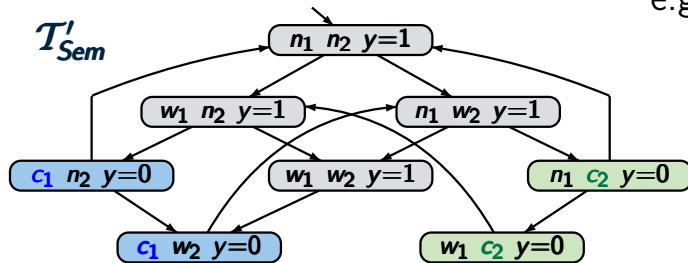
$$Paths(\mathcal{T}'_{Sem}) \subseteq Paths(\mathcal{T}_{Sem})$$



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



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



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

Trace inclusion appears naturally

- as an implementation/refinement relation
- when resolving nondeterminism



e.g., $Traces(\mathcal{T}'_{Sem}) \subseteq Traces(\mathcal{T}_{Sem})$

- in the context of abstractions

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

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

- in the context of abstractions

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

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

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

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

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

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

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

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

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

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

The following statements are equivalent:

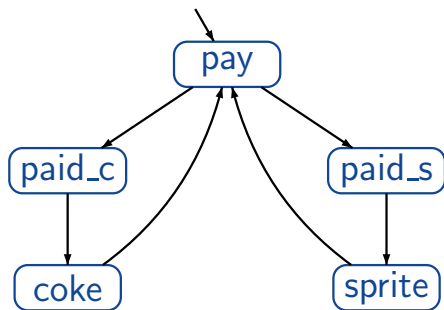
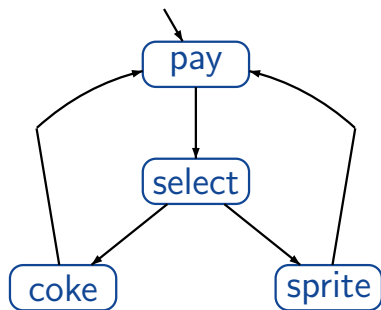
- (1) $Traces(\mathcal{T}_1) \subseteq Traces(\mathcal{T}_2)$
- (2) for all LT-properties E : $\mathcal{T}_2 \models E \implies \mathcal{T}_1 \models E$

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 $\mathcal{T}_2 \models E$

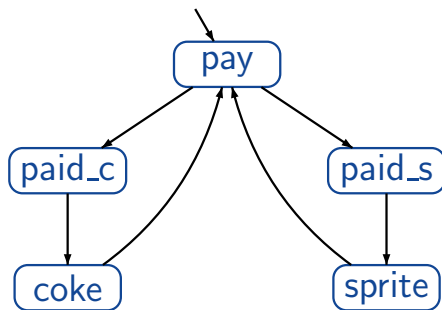
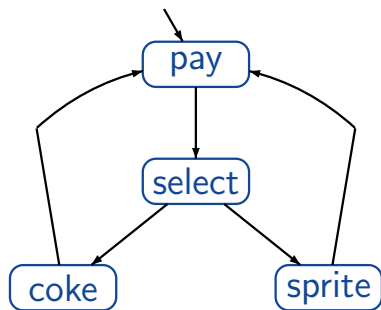
Trace equivalent beverage machines

LTB2.4-22



Trace equivalent beverage machines

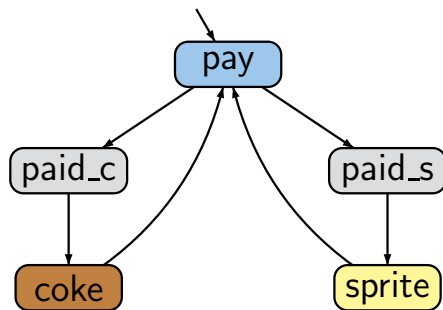
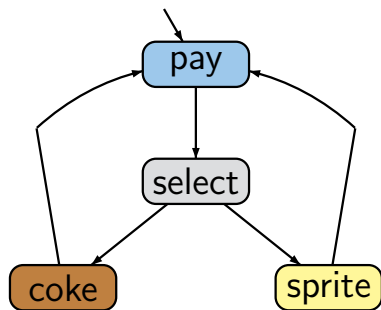
LTB2.4-22



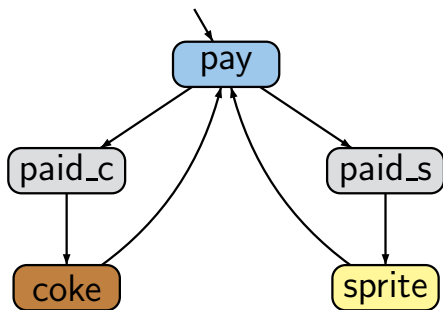
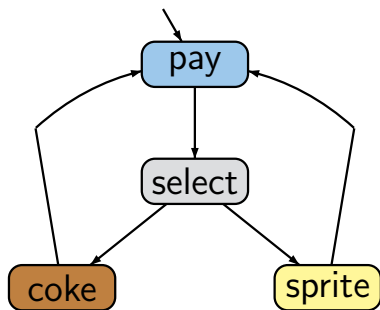
set of atomic propositions $AP = \{\text{pay}, \text{coke}, \text{sprite}\}$

Trace equivalent beverage machines

LTB2.4-22



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



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

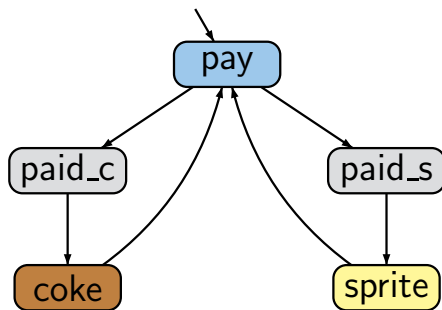
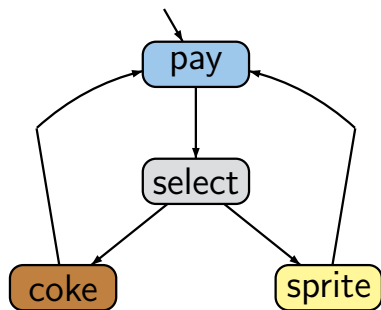
$Traces(\mathcal{T}_1) = Traces(\mathcal{T}_2) =$ set of all infinite words

$\{\text{pay}\} \emptyset \{\text{drink}_1\} \{\text{pay}\} \emptyset \{\text{drink}_2\} \dots$

where $\text{drink}_1, \text{drink}_2, \dots \in \{\text{coke}, \text{sprite}\}$

Trace equivalent beverage machines

LTB2.4-22



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

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

$\{\text{pay}\} \emptyset \{\text{drink}_1\} \{\text{pay}\} \emptyset \{\text{drink}_2\} \dots$

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

Introduction

Modelling parallel systems

Linear Time Properties

state-based and linear time view

definition of linear time properties

invariants and safety



liveness and fairness

Regular Properties

Linear Temporal Logic

Computation-Tree Logic

Equivalences and Abstraction

safety properties *“nothing bad will happen”*

liveness properties *“something good will happen”*

safety properties *“nothing bad will happen”*

examples:

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

liveness properties *“something good will happen”*

safety properties *“nothing bad will happen”*

examples:

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

liveness properties *“something good will happen”*

examples:

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

safety properties *“nothing bad will happen”*

examples:

- mutual exclusion
 - deadlock freedom
 - “every red phase is preceded by a yellow phase”
- } special case: **invariants**
“no bad state will be reached”

liveness properties *“something good will happen”*

examples:

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

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

$$\Phi ::= \text{true} \mid \textcolor{red}{a} \mid \Phi_1 \wedge \Phi_2 \mid \neg \Phi$$

atomic proposition, i.e., $\textcolor{red}{a} \in AP$

$$\Phi ::= \text{true} \mid \textcolor{red}{a} \mid \Phi_1 \wedge \Phi_2 \mid \neg \Phi \mid \Phi_1 \vee \Phi_2 \mid \Phi_1 \rightarrow \Phi_2 \mid \dots$$

atomic proposition, i.e., $\textcolor{red}{a} \in AP$

$$\Phi ::= \text{true} \mid \textcolor{red}{a} \mid \Phi_1 \wedge \Phi_2 \mid \neg \Phi \mid \Phi_1 \vee \Phi_2 \mid \Phi_1 \rightarrow \Phi_2 \mid \dots$$

atomic proposition, i.e., $\textcolor{red}{a} \in AP$

semantics: interpretation over a subsets of AP

$$\Phi ::= \text{true} \mid \textcolor{red}{a} \mid \Phi_1 \wedge \Phi_2 \mid \neg \Phi \mid \Phi_1 \vee \Phi_2 \mid \Phi_1 \rightarrow \Phi_2 \mid \dots$$

atomic proposition, i.e., $\textcolor{red}{a} \in AP$

semantics: Let $A \subseteq AP$

$$A \models \text{true}$$

$$A \models \textcolor{red}{a} \quad \text{iff} \quad \textcolor{red}{a} \in A$$

$$A \models \Phi_1 \wedge \Phi_2 \quad \text{iff} \quad A \models \Phi_1 \text{ and } A \models \Phi_2$$

$$A \models \neg \Phi \quad \text{iff} \quad A \not\models \Phi$$

$$\Phi ::= \text{true} \mid \textcolor{red}{a} \mid \Phi_1 \wedge \Phi_2 \mid \neg \Phi \mid \Phi_1 \vee \Phi_2 \mid \Phi_1 \rightarrow \Phi_2 \mid \dots$$

atomic proposition, i.e., $\textcolor{red}{a} \in AP$

semantics: Let $A \subseteq AP$

$$A \models \text{true}$$

$$A \models \textcolor{red}{a} \quad \text{iff} \quad \textcolor{red}{a} \in A$$

$$A \models \Phi_1 \wedge \Phi_2 \quad \text{iff} \quad A \models \Phi_1 \text{ and } A \models \Phi_2$$

$$A \models \neg \Phi \quad \text{iff} \quad A \not\models \Phi$$

$$\text{e.g., } \{\textcolor{red}{a}, \textcolor{red}{b}\} \not\models (\textcolor{red}{a} \rightarrow \neg \textcolor{red}{b}) \vee \textcolor{red}{c} \quad \{\textcolor{red}{a}, \textcolor{red}{b}\} \models \textcolor{red}{a} \vee \textcolor{red}{c}$$

$$\Phi ::= \text{true} \mid \textcolor{red}{a} \mid \Phi_1 \wedge \Phi_2 \mid \neg \Phi \mid \Phi_1 \vee \Phi_2 \mid \Phi_1 \rightarrow \Phi_2 \mid \dots$$

atomic proposition, i.e., $\textcolor{red}{a} \in AP$

semantics: Let $A \subseteq AP$

$$A \models \text{true}$$

$$A \models \textcolor{red}{a} \quad \text{iff} \quad \textcolor{red}{a} \in A$$

$$A \models \Phi_1 \wedge \Phi_2 \quad \text{iff} \quad A \models \Phi_1 \text{ and } A \models \Phi_2$$

$$A \models \neg \Phi \quad \text{iff} \quad A \not\models \Phi$$

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

Let E be an LT property over AP .

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

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

Let E be an LT property over AP .

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

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

Φ is called the **invariant condition** of E .

mutual exclusion (safety):

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

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

mutual exclusion (safety):

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

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

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

mutual exclusion (safety):

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

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

deadlock freedom for 5 dining philosophers:

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

here: $AP = \{\text{wait}_j : 0 \leq j \leq 4\} \cup \{\dots\}$

mutual exclusion (safety):

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

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

deadlock freedom for 5 dining philosophers:

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

invariant condition: ?

here: $AP = \{\text{wait}_j : 0 \leq j \leq 4\} \cup \{\dots\}$

mutual exclusion (safety):

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

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

deadlock freedom for 5 dining philosophers:

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

invariant condition:

$$\Phi = \neg \text{wait}_0 \vee \neg \text{wait}_1 \vee \neg \text{wait}_2 \vee \neg \text{wait}_3 \vee \neg \text{wait}_4$$

here: $AP = \{\text{wait}_j : 0 \leq j \leq 4\} \cup \{\dots\}$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

↑
set of reachable states in \mathcal{T}

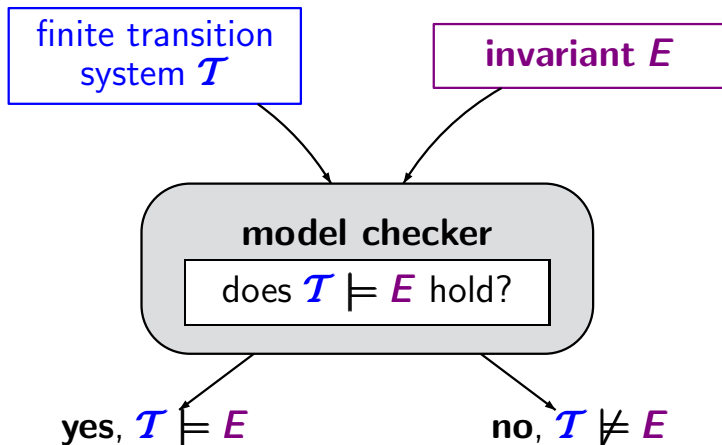
Let E be an LT property over AP . E is called an invariant if there exists a propositional formula Φ s.t.

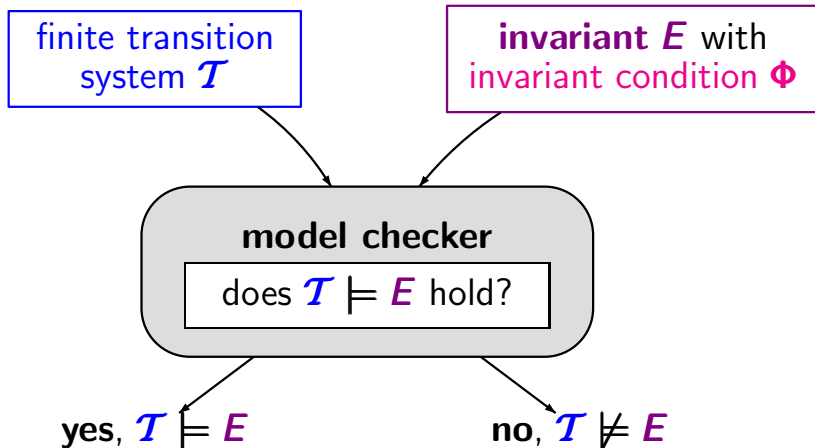
$$E = \{ A_0 A_1 A_2 \dots \in (2^{AP})^\omega : \forall i \geq 0. A_i \models \Phi \}$$

Let T be a TS over AP without terminal states. Then:

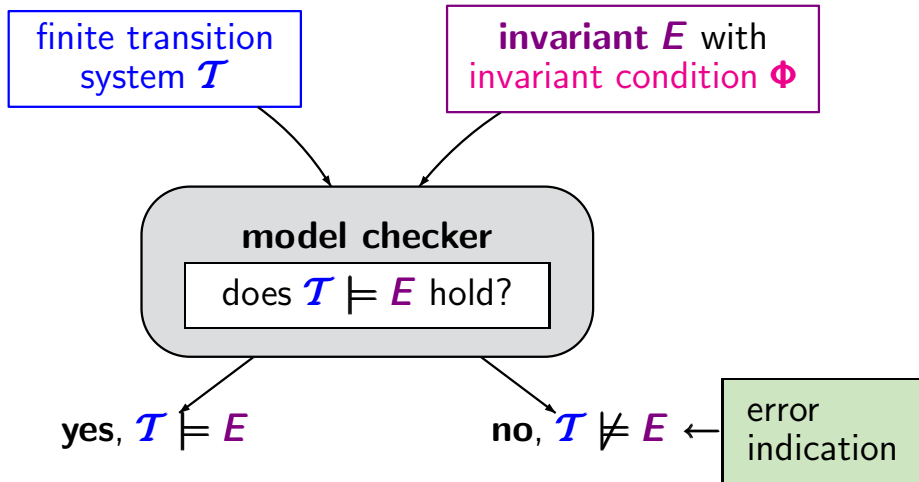
$$\begin{aligned} T \models E & \text{ iff } \text{trace}(\pi) \in E \text{ for all } \pi \in \text{Paths}(T) \\ & \text{ iff } s \models \Phi \text{ for all states } s \text{ on a path of } T \\ & \text{ iff } s \models \Phi \text{ for all states } s \in \text{Reach}(T) \end{aligned}$$

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

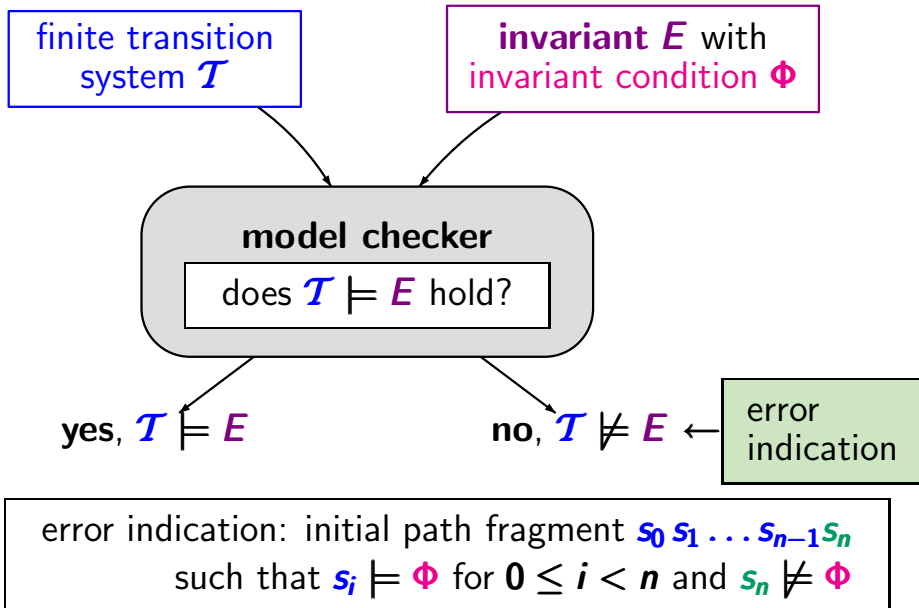




perform a graph analysis (**DFS** or **BFS**) to check whether $s \models \Phi$ for all $s \in \text{Reach}(\mathcal{T})$



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



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

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

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

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

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

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

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

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

FOR ALL $s_0 \in S_0$ DO

IF $DFS(s_0, \Phi)$ THEN

return “no” and $reverse(\pi)$

FI

OD

return “yes”

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

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

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

FOR ALL $s_0 \in S_0$ DO

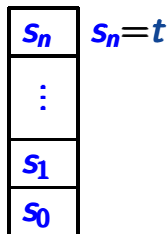
IF $DFS(s_0, \Phi)$ THEN

return “no” and $reverse(\pi)$

FI

OD

return “yes”



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

DFS-based invariant checking

LTPROP/Is2.5-7

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

$U := \emptyset \leftarrow$ stores the “processed” states

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

FOR ALL $s_0 \in S_0$ DO

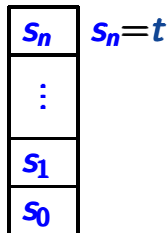
IF $DFS(s_0, \Phi)$ THEN

return “no” and $reverse(\pi)$

FI

OD

return “yes”



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

“searches” for a path fragment $s \dots t$ with $t \notin \Phi$

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

```
IF  $s \notin U$  THEN
  IF  $s \not\models \Phi$  THEN return “true” FI
  IF  $s \models \Phi$  THEN
    :
  FI
FI
return “false”
```

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

```
IF  $s \notin U$  THEN
  IF  $s \not\models \Phi$  THEN return “true” FI
  IF  $s \models \Phi$  THEN
    insert  $s$  in  $U$ ;

FI
return “false”
```

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

```
IF  $s \notin U$  THEN
  IF  $s \not\models \Phi$  THEN return “true” FI
  IF  $s \models \Phi$  THEN
    insert  $s$  in  $U$ ;
    FOR ALL  $s' \in Post(s)$  DO
      IF  $DFS(s', \Phi)$  THEN
        return “true” FI
    OD
  FI
FI
return “false”
```

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

$Push(\pi, s);$

IF $s \notin U$ THEN

IF $s \not\models \Phi$ THEN return “true” FI

IF $s \models \Phi$ THEN

insert s in U ;

FOR ALL $s' \in Post(s)$ DO

IF $DFS(s', \Phi)$ THEN

return “true” FI

OD

FI FI

$Pop(\pi);$ return “false”

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

$Push(\pi, s);$

IF $s \notin U$ THEN

IF $s \not\models \Phi$ THEN return “true” FI

IF $s \models \Phi$ THEN

insert s in U ;

FOR ALL $s' \in Post(s)$ DO

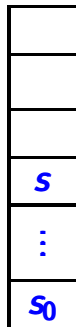
IF $DFS(s', \Phi)$ THEN

return “true” FI

OD

FI FI

$Pop(\pi);$ return “false”



initial
state

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

$Push(\pi, s);$

IF $s \notin U$ THEN

IF $s \not\models \Phi$ THEN return “true” FI

IF $s \models \Phi$ THEN

insert s in U ;

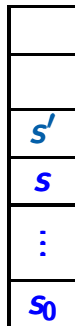
FOR ALL $s' \in Post(s)$ DO

IF $DFS(s', \Phi)$ THEN
return “true” FI

OD

FI FI

$Pop(\pi);$ return “false”



initial
state

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

$Push(\pi, s);$

IF $s \notin U$ THEN

IF $s \not\models \Phi$ THEN return “true” FI

IF $s \models \Phi$ THEN

insert s in U ;

FOR ALL $s' \in Post(s)$ DO

IF $DFS(s', \Phi)$ THEN

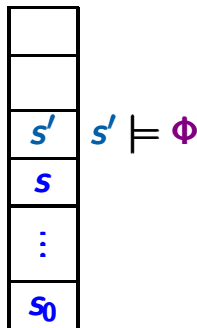
return “true” FI

OD

FI

FI

$Pop(\pi);$ return “false”



“searches” for a path fragment $s \dots s' \dots t$ with $t \not\models \Phi$

$Push(\pi, s);$

IF $s \notin U$ THEN

IF $s \not\models \Phi$ THEN return “true” FI

IF $s \models \Phi$ THEN

insert s in U ;

FOR ALL $s' \in Post(s)$ DO

IF $DFS(s', \Phi)$ THEN

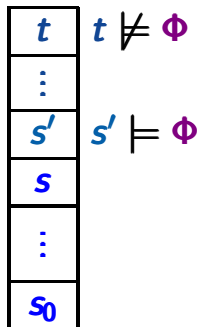
return “true” FI

OD

FI

FI

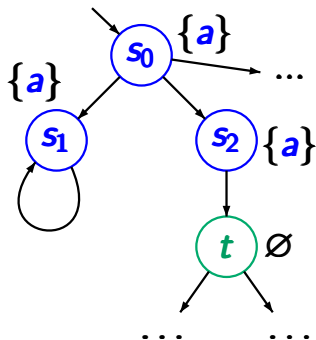
$Pop(\pi);$ return “false”



initial
state

Example: invariant checking

IS2.5-9

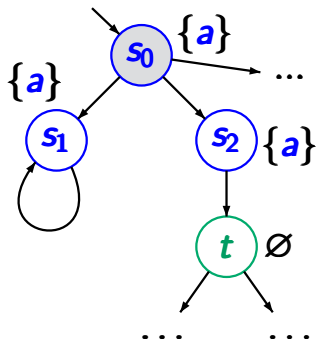


invariant
condition a

$$\begin{array}{lcl} s_0, s_1, s_2 & | & \models a \\ t & | & \not\models a \end{array}$$

Example: invariant checking

IS2.5-9



$DFS(s_0, a)$

stack π

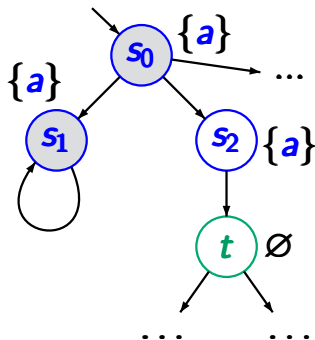


invariant
condition a

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

Example: invariant checking

IS2.5-9



$DFS(s_0, a)$

$DFS(s_1, a)$

stack π

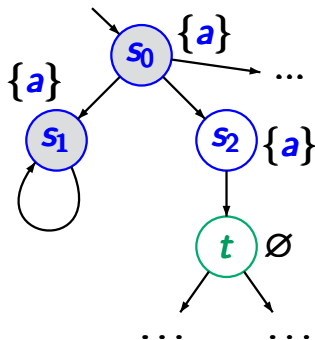


invariant
condition a

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

Example: invariant checking

IS2.5-9

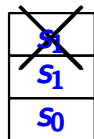


$DFS(s_0, a)$

$DFS(s_1, a)$

$DFS(s_1, a)$

stack π

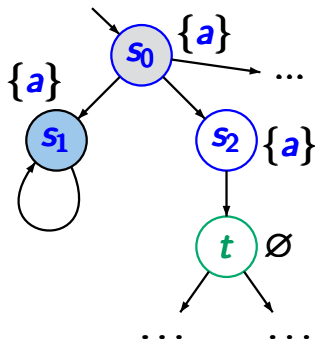


invariant
condition a

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

Example: invariant checking

IS2.5-9



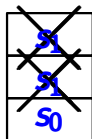
invariant
condition a

$$\begin{array}{lcl} s_0, s_1, s_2 & | \models & a \\ t & | \not\models & a \end{array}$$

$DFS(s_0, a)$

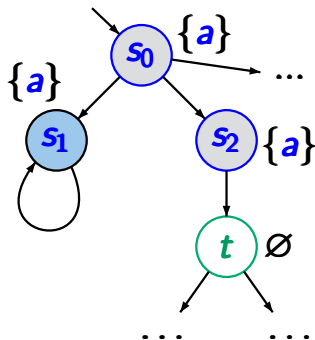


stack π



Example: invariant checking

IS2.5-9



invariant
condition a

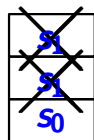
$$\begin{array}{c} s_0, s_1, s_2 \\ t \end{array} \begin{array}{l} \models a \\ \not\models a \end{array}$$

$DFS(s_0, a)$



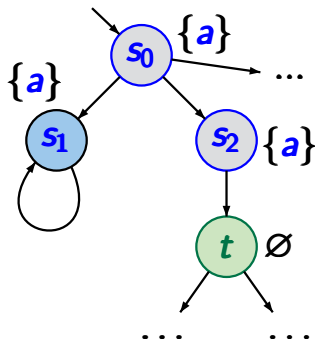
$DFS(s_2, a)$

stack π



Example: invariant checking

IS2.5-9



invariant
condition a

$$\begin{array}{c|c} s_0, s_1, s_2 & \models a \\ t & \not\models a \end{array}$$

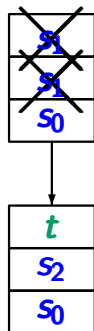
$DFS(s_0, a)$



$DFS(s_2, a)$

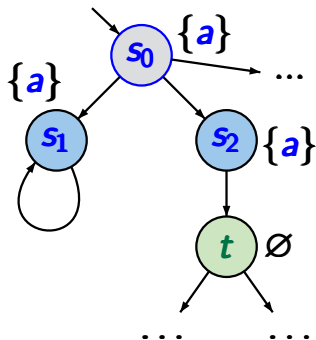


stack π



Example: invariant checking

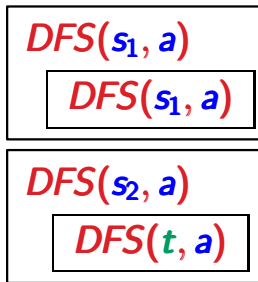
IS2.5-9



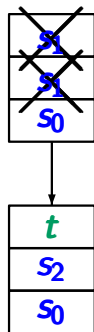
invariant
condition a

$$\begin{array}{lcl} s_0, s_1, s_2 & | \models & a \\ t & | \not\models & a \end{array}$$

$DFS(s_0, a)$

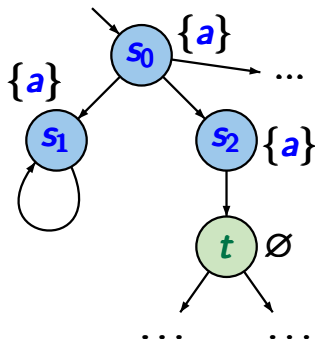


stack π



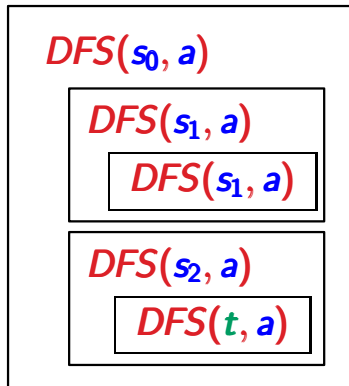
Example: invariant checking

IS2.5-9

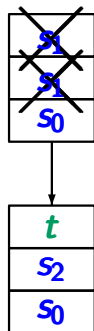


invariant
condition a

$$\begin{array}{l} s_0, s_1, s_2 \mid \models a \\ \quad t \mid \not\models a \end{array}$$

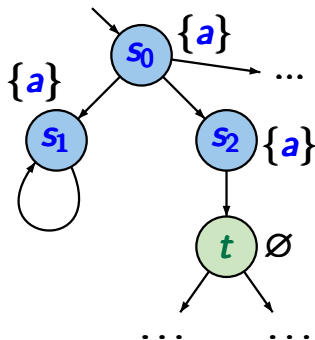


stack π



Example: invariant checking

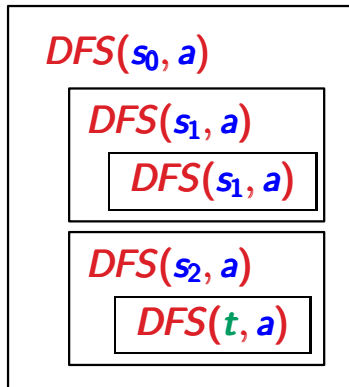
IS2.5-9



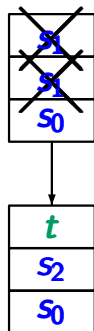
invariant
condition a

$$\begin{array}{l} s_0, s_1, s_2 \models a \\ t \not\models a \end{array}$$

$s_0 \not\models$ "always a "

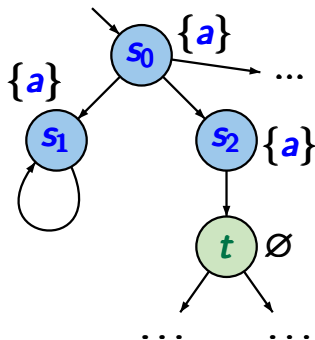


stack π



Example: invariant checking

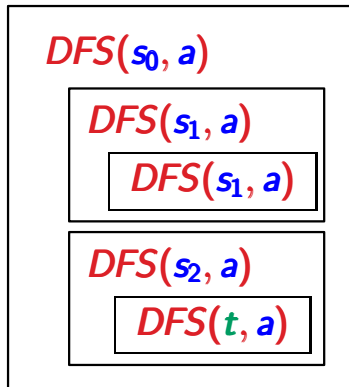
IS2.5-9



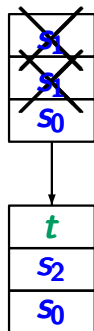
invariant
condition a

$$\begin{array}{c} s_0, s_1, s_2 \\ t \end{array} \models a$$

$$\begin{array}{c} s_0, s_1, s_2 \\ t \end{array} \not\models a$$



stack π



$s_0 \not\models$ "always a "

error
indication:

$s_0 s_2 t$