Overview: Model Checking

1. Introduction
2. Modelling parallel systems
3. Linear Time Properties
4. Regular Properties
5. Linear Temporal Logic
6. Computation Tree Logic
7. Equivalences and Abstraction
8. **Partial Order Reduction**
9. Timed Automata
10. Probabilistic Systems

A few historical notes

1977 temporal logics (**LTL**) as specification formalism
for parallel systems [Pnueli]

1981 model checking
- for **CTL** [Clarke/Emerson], [Queile/Sifakis]
- for **LTL** [Lichtenstein/Pnueli], [Vardi/Wolper]
- for **CTL*** [Emerson/Lei]
- ...

A few historical notes

1977 temporal logics (**LTL**) as specification formalism
for parallel systems [Pnueli]

1981 model checking
- for **CTL** [Clarke/Emerson], [Queile/Sifakis]
- for **LTL** [Lichtenstein/Pnueli], [Vardi/Wolper]
- for **CTL*** [Emerson/Lei]
- ...

.. state explosion problem ...

A few historical notes

1977 temporal logics (**LTL**) as specification formalism
for parallel systems [Pnueli]

1981 model checking

... to attack the state explosion problem ...

1987 symbolic model checking with BDDs [McMillan]
1990 partial order reduction
[Godefroid], [Valmari], [Peled]
1992 net unfoldings
abstraction-refinement
symmetry reduction

A few historical notes

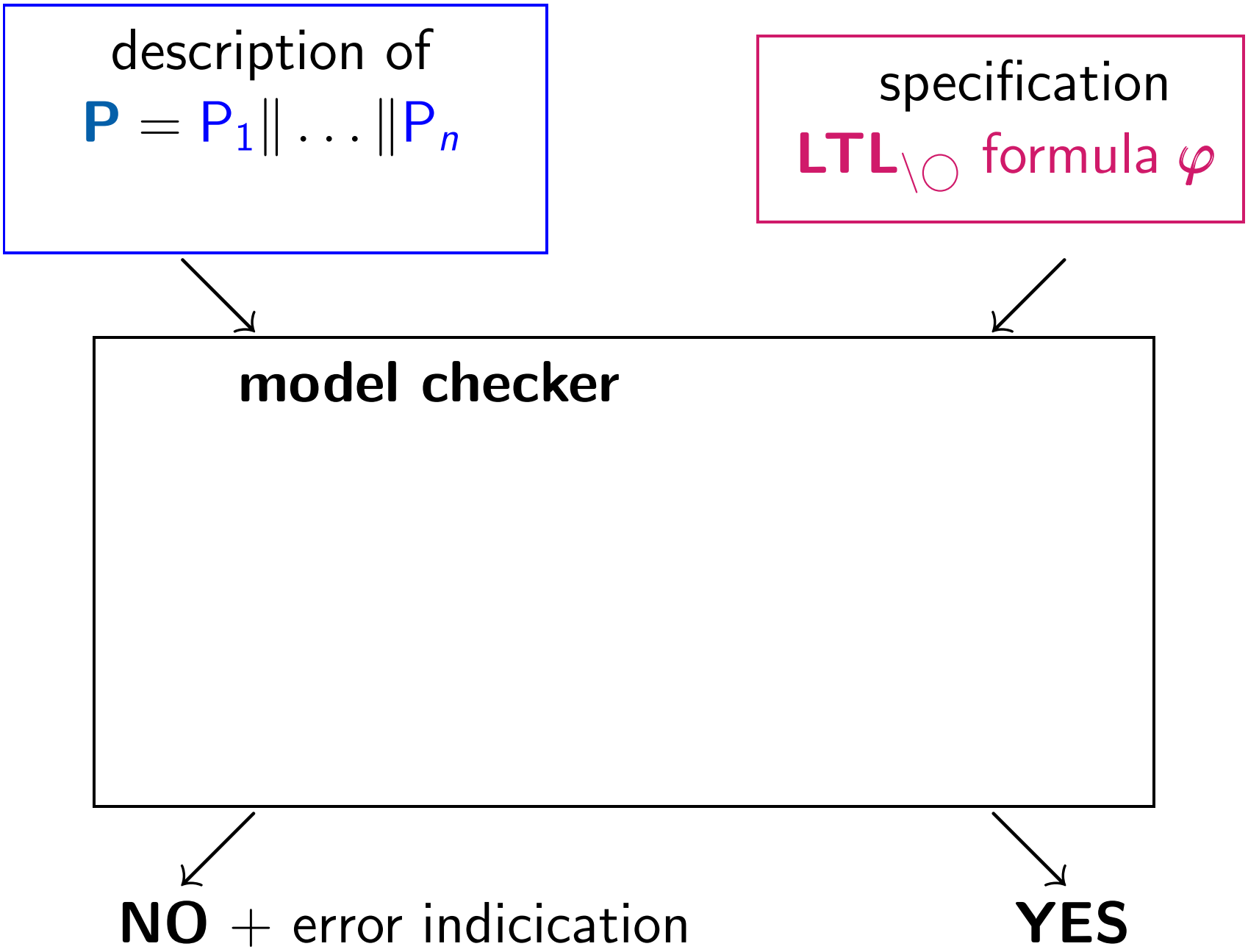
1977 *temporal logics* (LTL) as specification formalism
for parallel systems [Pnueli]

1981 *model checking*

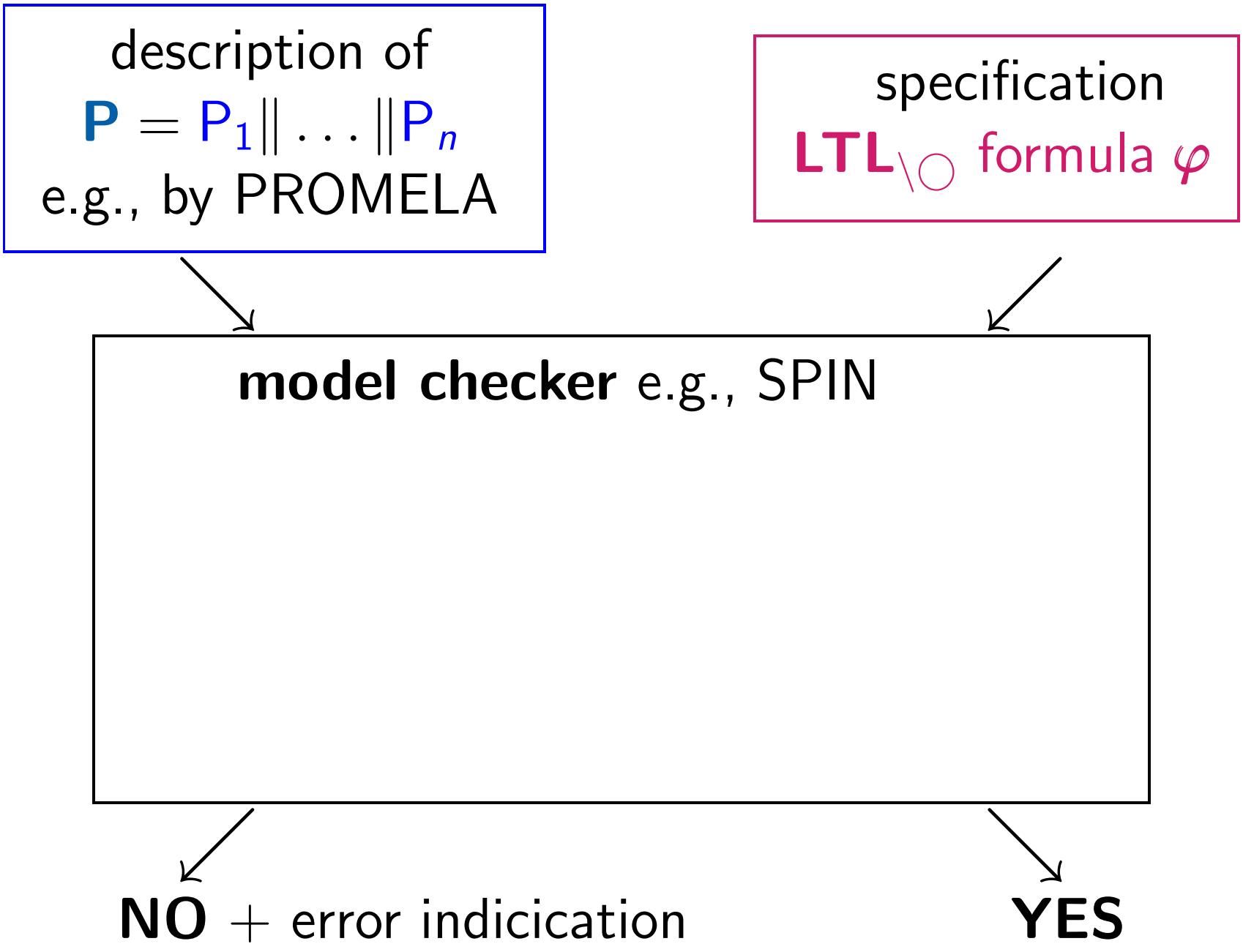
... to attack the *state explosion problem* ...

1987 symbolic model checking with BDDs [McMillan]
1990 **partial order reduction** for **LTL**$_{\setminus\bigcirc}$
[Godefroid], [Valmari], [Peled]
1992 net unfoldings
abstraction-refinement
symmetry reduction

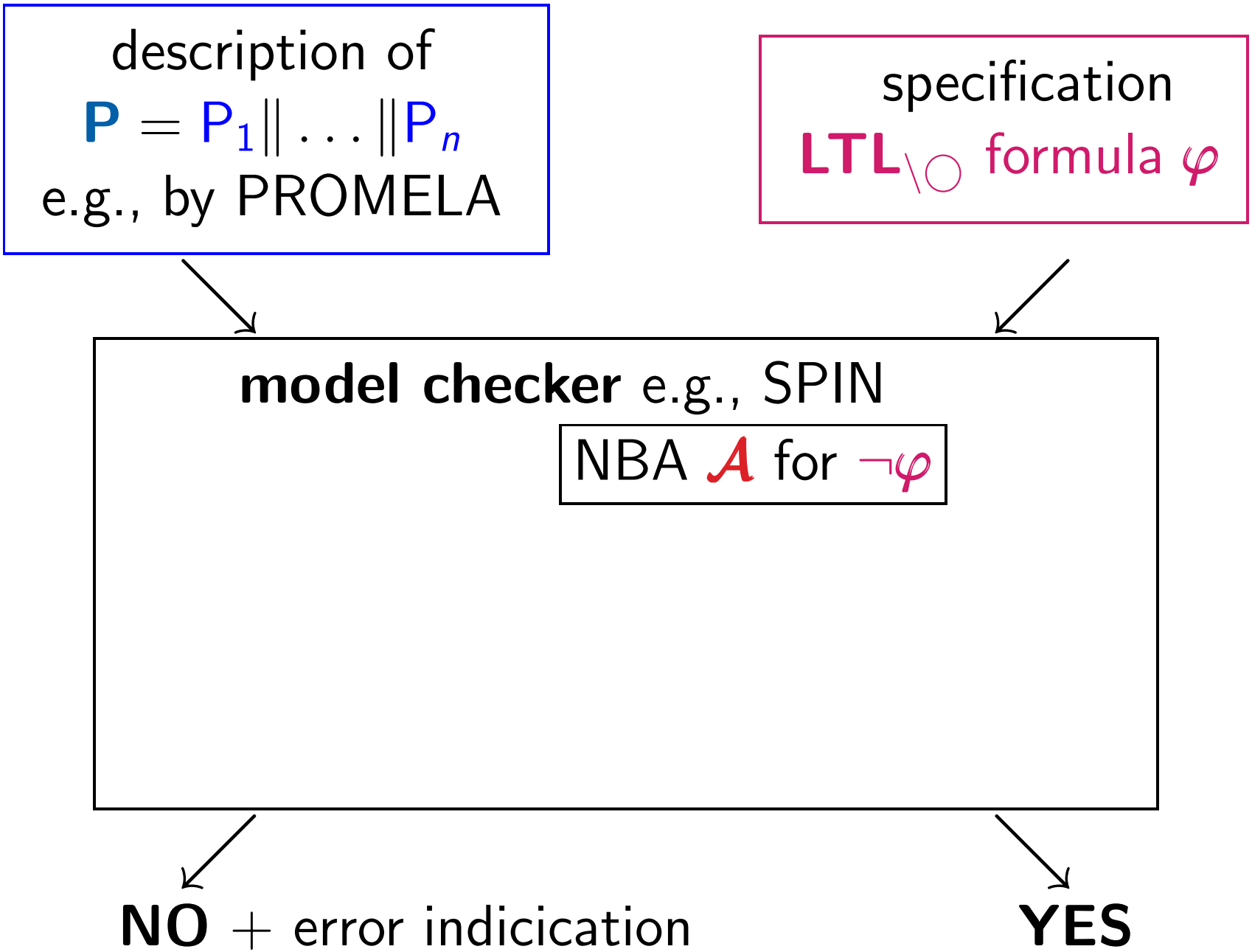
Partial order reduction



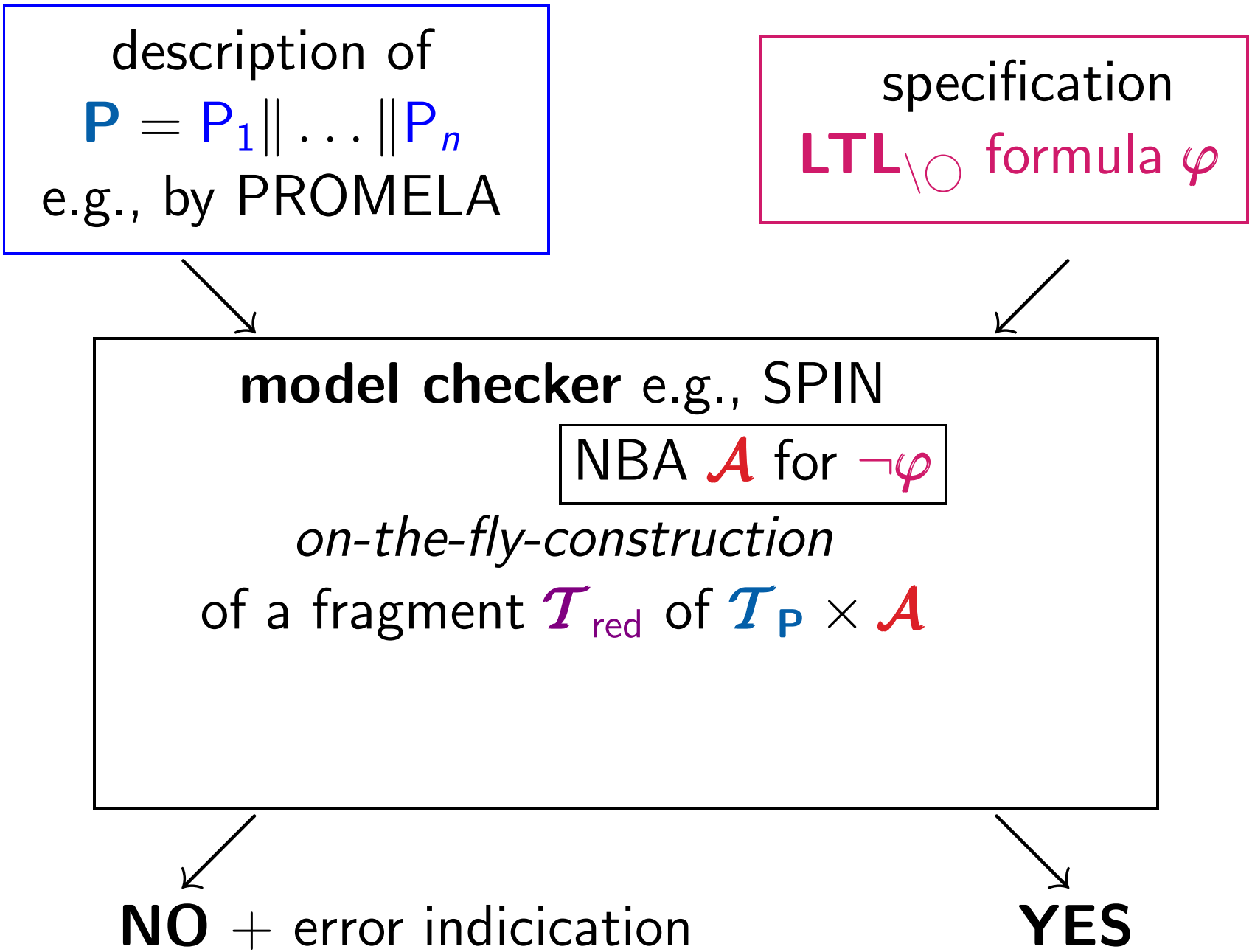
Partial order reduction



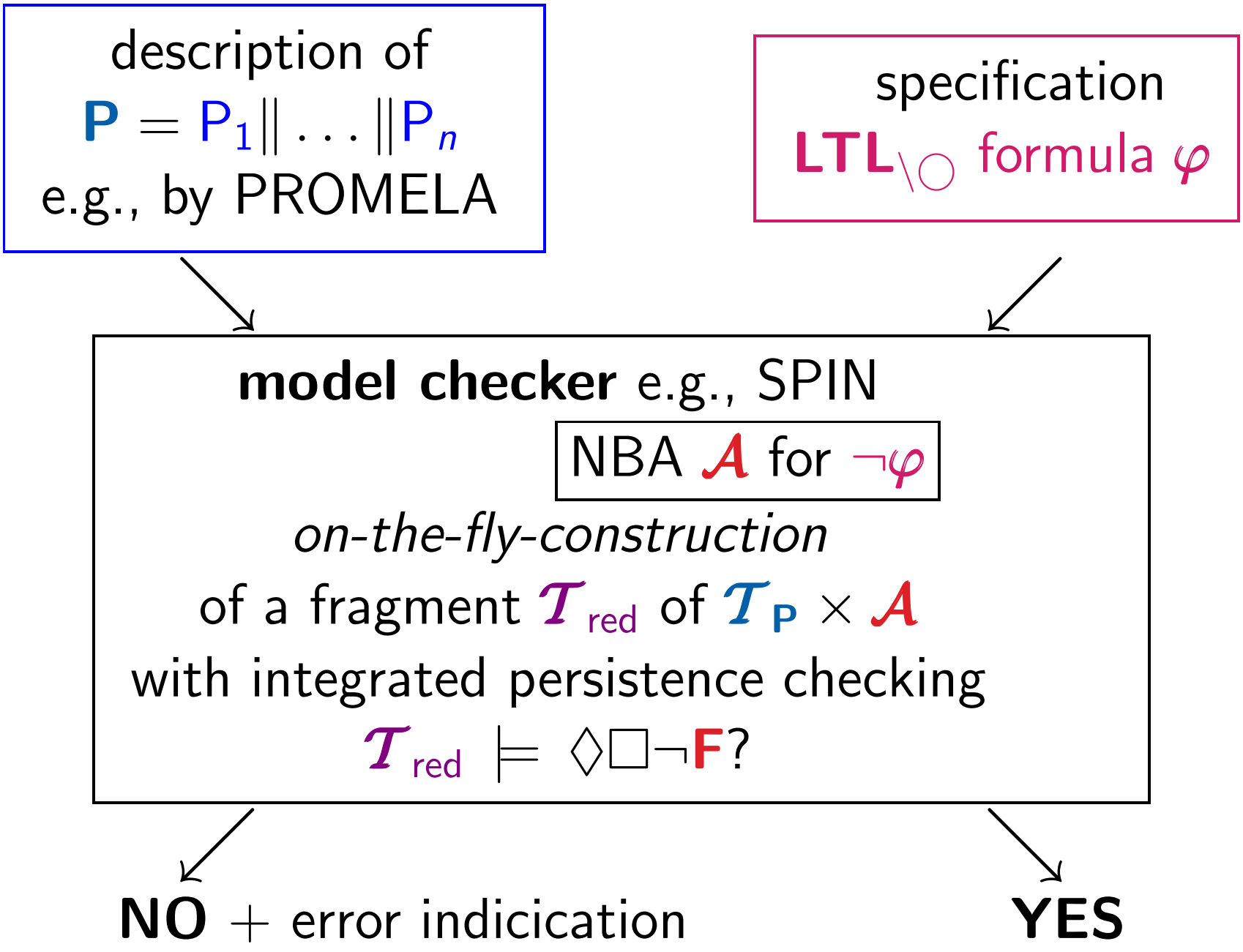
Partial order reduction



Partial order reduction



Partial order reduction



Basic idea of partial order reduction

- for asynchronous systems

Basic idea of partial order reduction

- for asynchronous systems
- analyze representatives of path equivalence classes

Basic idea of partial order reduction

- for asynchronous systems
- analyze representatives of path equivalence classes that represent the same the same behavior up to the interleaving order

Basic idea of partial order reduction

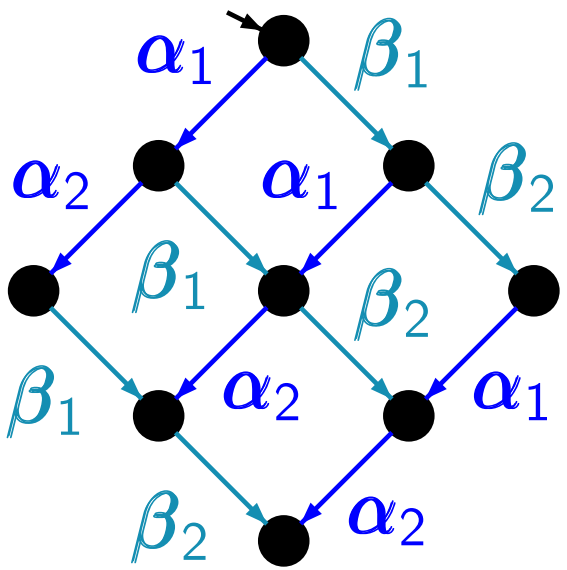
- for asynchronous systems
- analyze representatives of path equivalence classes that represent the same the same behavior up to the interleaving order

$$\mathcal{T} = \mathcal{T}_1 ||| \mathcal{T}_2$$

Basic idea of partial order reduction

- for asynchronous systems
- analyze representatives of path equivalence classes that represent the same the same behavior up to the interleaving order

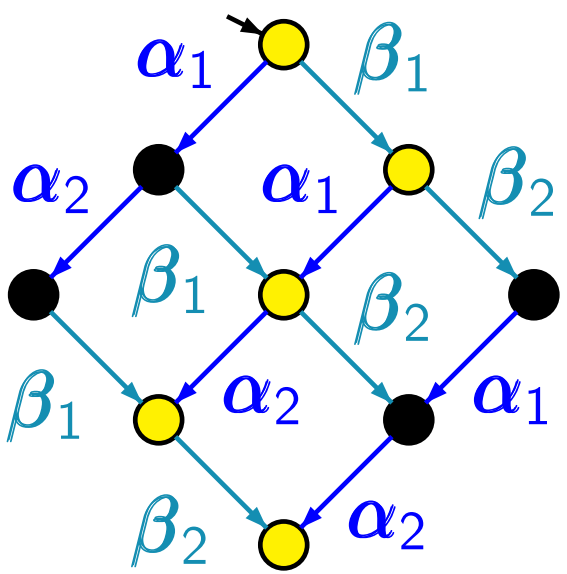
$$\mathcal{T} = \mathcal{T}_1 \,|||\, \mathcal{T}_2$$



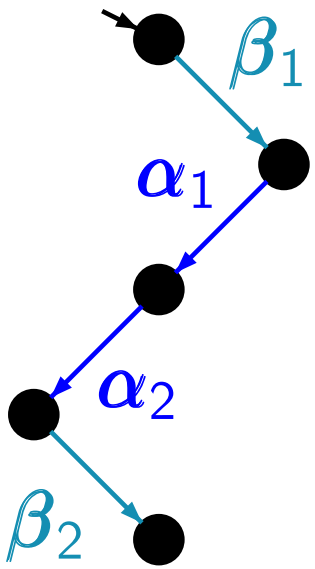
Basic idea of partial order reduction

- for asynchronous systems
- analyze representatives of path equivalence classes that represent the same the same behavior up to the interleaving order

$\mathcal{T} = \mathcal{T}_1 \,|||\, \mathcal{T}_2$

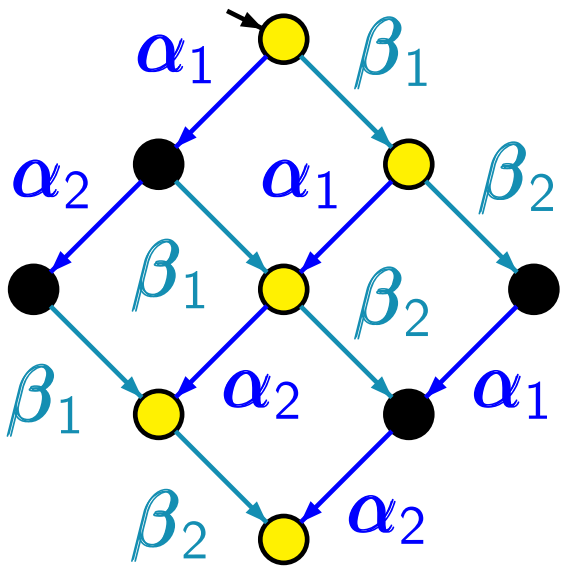


$\mathcal{T}_{\text{red}}$

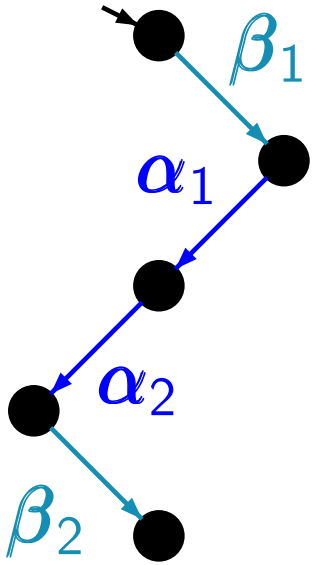


Partial order reduction for $\mathbf{LTL}_{\setminus\bigcirc}$ specifications

$\mathcal{T} = \mathcal{T}_1 \,|||\, \mathcal{T}_2$



$\mathcal{T}_{\text{red}}$



requirement: for all $\mathbf{LTL}_{\setminus\bigcirc}$ formulas φ:

$$\mathcal{T} \models \varphi \;\text{ iff }\; \mathcal{T}_{\text{red}} \models \varphi$$

hence: ensure that the reduction yields $\mathcal{T} \stackrel{\Delta}{=} \mathcal{T}_{\text{red}}$

The ample set method [Peled '93]

given: syntactical representation of processes of TS $\mathcal{T}$

goal: on-the-fly construction of a fragment $\mathcal{T}_{\text{red}}$

The ample set method [Peled '93]

given: syntactical representation of processes of TS $\mathcal{T}$

goal: on-the-fly construction of a fragment $\mathcal{T}_{\text{red}}$
by selecting action-sets $\text{ample}(s) \subseteq \text{Act}(s)$

The ample set method [Peled '93]

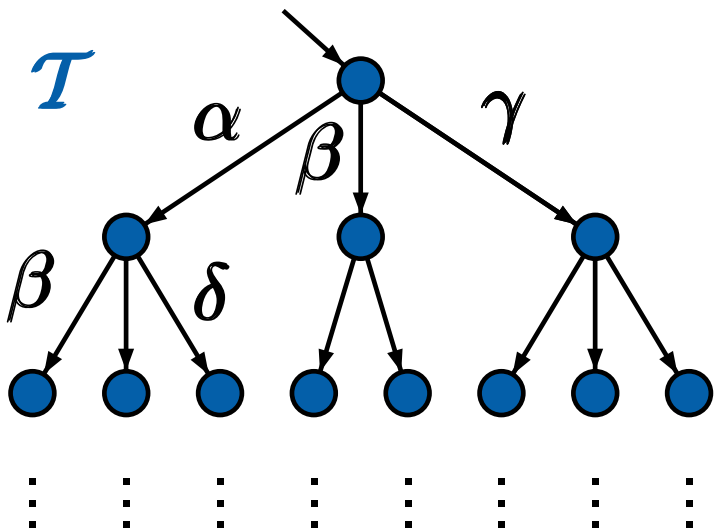
given: syntactical representation of processes of TS $\mathcal{T}$

goal: on-the-fly construction of a fragment $\mathcal{T}_{\text{red}}$
by selecting action-sets $\text{ample}(s) \subseteq \text{Act}(s)$
and expanding only the α-successors of s
where $\alpha \in \text{ample}(s)$

The ample set method [Peled '93]

given: syntactical representation of processes of TS $\mathcal{T}$

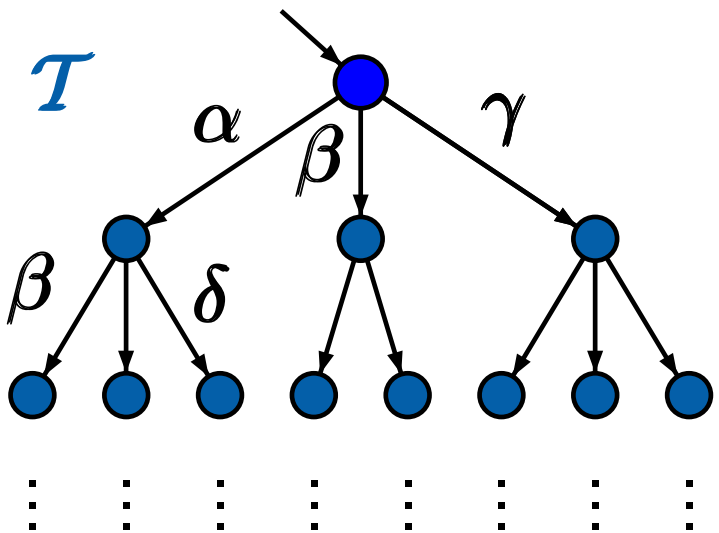
goal: on-the-fly construction of a fragment $\mathcal{T}_{\text{red}}$
by selecting action-sets ample(s) $\subseteq$ Act(s)
and expanding only the α-successors of s
where $\alpha \in$ ample(s)



The ample set method [Peled '93]

given: syntactical representation of processes of TS $\mathcal{T}$

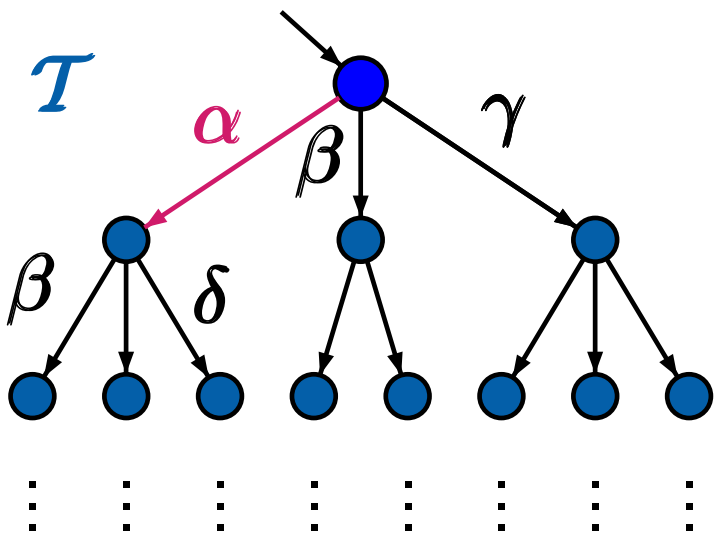
goal: on-the-fly construction of a fragment $\mathcal{T}_{\text{red}}$
by selecting action-sets $\text{ample}(s) \subseteq \text{Act}(s)$
and expanding only the α-successors of s
where $\alpha \in \text{ample}(s)$



The ample set method [Peled '93]

given: syntactical representation of processes of TS $\mathcal{T}$

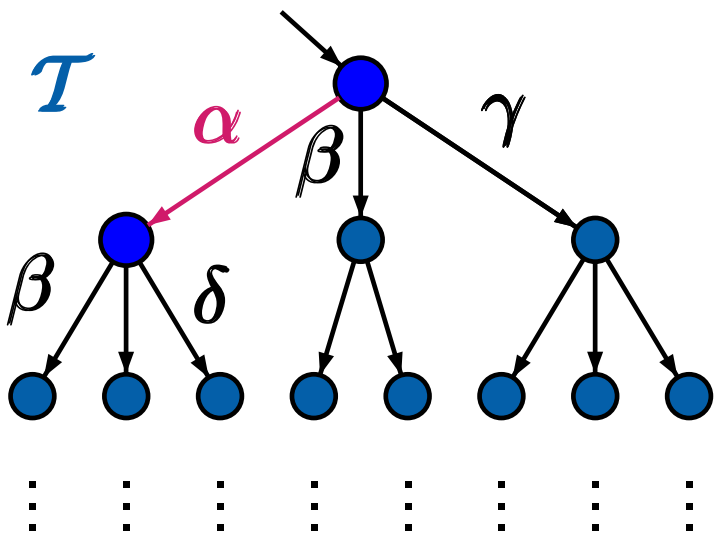
goal: on-the-fly construction of a fragment $\mathcal{T}_{\text{red}}$
by selecting action-sets $\text{ample}(s) \subseteq \text{Act}(s)$
and expanding only the α-successors of s
where $\alpha \in \text{ample}(s)$



The ample set method [Peled '93]

given: syntactical representation of processes of TS $\mathcal{T}$

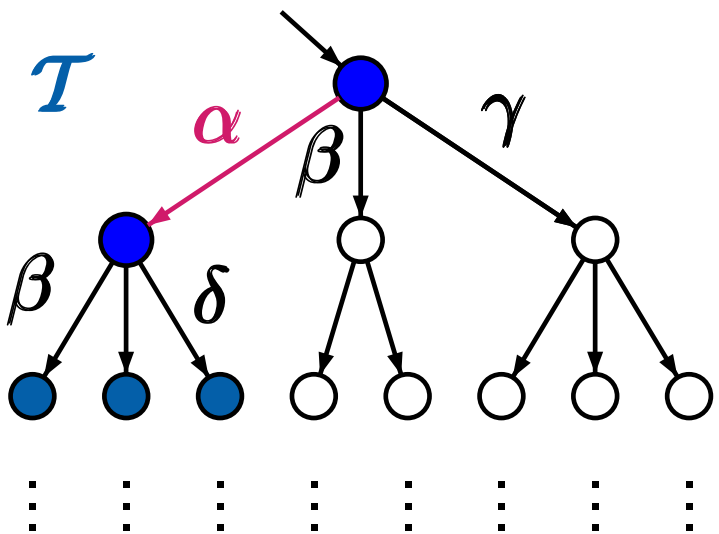
goal: on-the-fly construction of a fragment $\mathcal{T}_{\text{red}}$
by selecting action-sets $\text{ample}(s) \subseteq \text{Act}(s)$
and expanding only the α-successors of s
where $\alpha \in \text{ample}(s)$



The ample set method [Peled '93]

given: syntactical representation of processes of TS $\mathcal{T}$

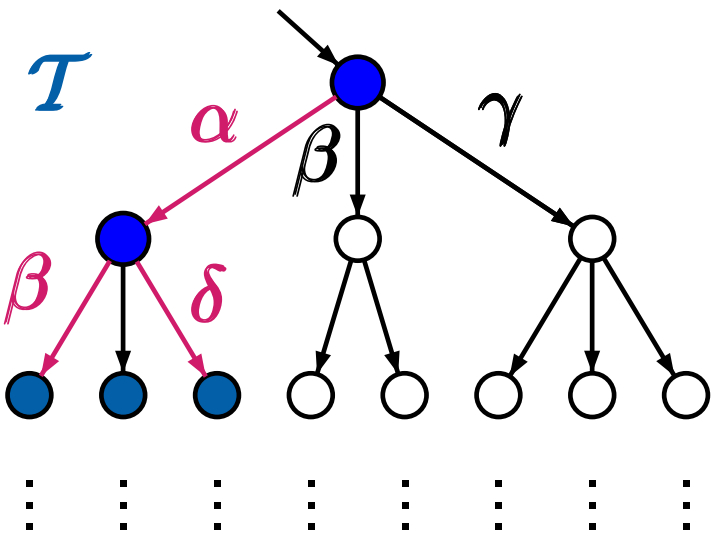
goal: on-the-fly construction of a fragment $\mathcal{T}_{\text{red}}$
by selecting action-sets $\text{ample}(s) \subseteq \text{Act}(s)$
and expanding only the α-successors of s
where $\alpha \in \text{ample}(s)$



The ample set method [Peled '93]

given: syntactical representation of processes of TS $\mathcal{T}$

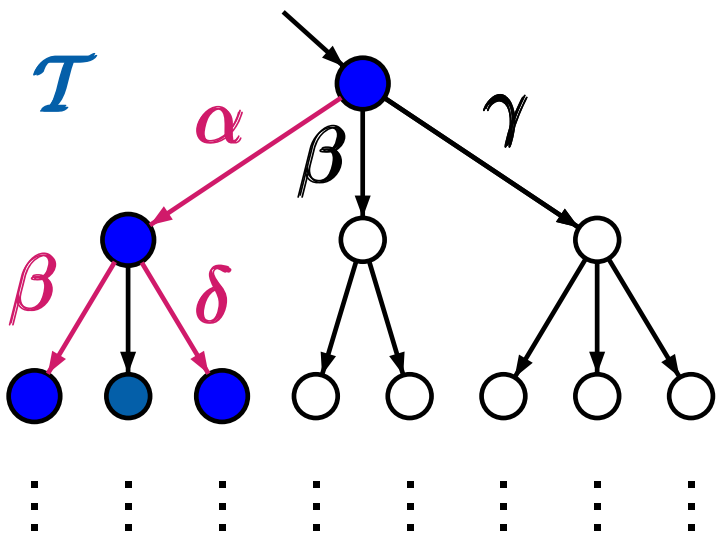
goal: on-the-fly construction of a fragment $\mathcal{T}_{\text{red}}$
by selecting action-sets $\text{ample}(s) \subseteq \text{Act}(s)$
and expanding only the α-successors of s
where $\alpha \in \text{ample}(s)$



The ample set method [Peled '93]

given: syntactical representation of processes of TS $\mathcal{T}$

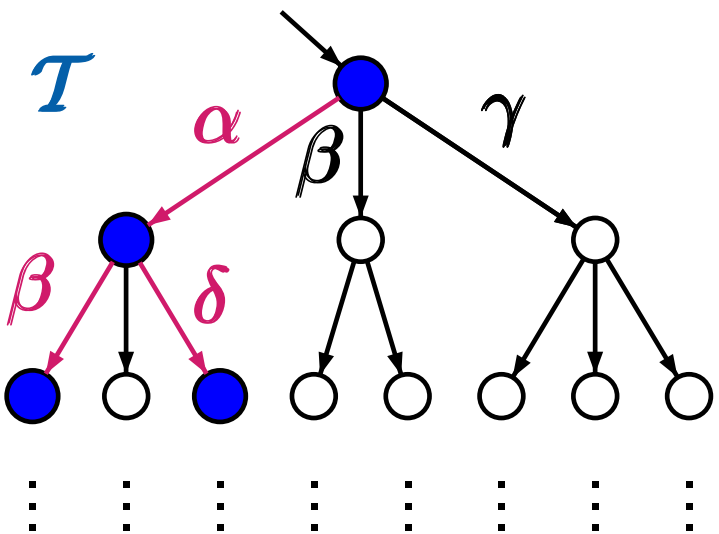
goal: on-the-fly construction of a fragment $\mathcal{T}_{\text{red}}$
by selecting action-sets $\text{ample}(s) \subseteq \text{Act}(s)$
and expanding only the α-successors of s
where $\alpha \in \text{ample}(s)$



The ample set method [Peled '93]

given: syntactical representation of processes of TS $\mathcal{T}$

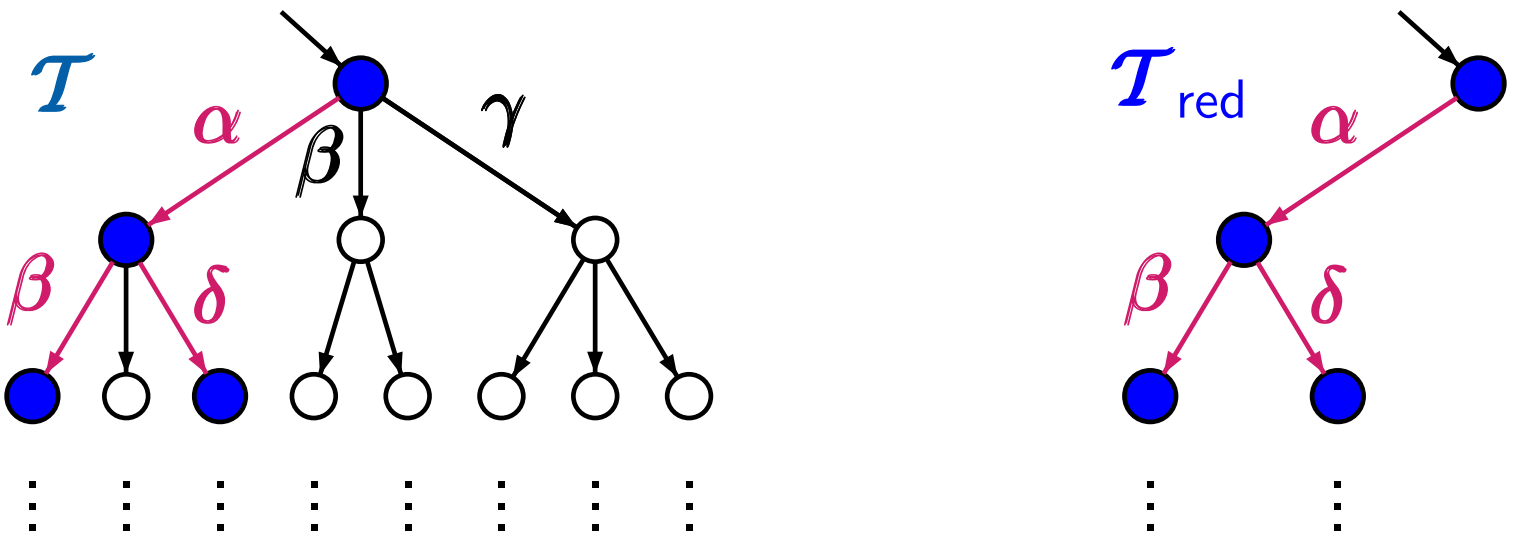
goal: on-the-fly construction of a fragment $\mathcal{T}_{\text{red}}$
by selecting action-sets $\text{ample}(s) \subseteq \text{Act}(s)$
and expanding only the α-successors of s
where $\alpha \in \text{ample}(s)$



The ample set method [Peled '93]

given: syntactical representation of processes of TS $\mathcal{T}$

goal: on-the-fly construction of a fragment $\mathcal{T}_{\text{red}}$
by selecting action-sets ample(s) $\subseteq$ Act(s)
and expanding only the α-successors of s
where $\alpha \in$ ample(s)



The ample set method [Peled '93]

given: syntactical representation of processes of TS $\mathcal{T}$

goal: on-the-fly construction of a fragment $\mathcal{T}_{\text{red}}$
by selecting action-sets $\text{ample}(s) \subseteq \text{Act}(s)$
and expanding only the α-successors of s
where $\alpha \in \text{ample}(s)$

requirements:

The ample set method [Peled '93]

given: syntactical representation of processes of TS $\mathcal{T}$

goal: on-the-fly construction of a fragment $\mathcal{T}_{\mathsf{red}}$
by selecting action-sets $\mathsf{ample(s)} \subseteq \mathsf{Act(s)}$
and expanding only the α-successors of s
where $\alpha \in \mathsf{ample(s)}$

requirements:

- stutter trace equivalence: $\mathcal{T} \stackrel{\Delta}{=} \mathcal{T}_{\mathsf{red}}$

The ample set method [Peled '93]

given: syntactical representation of processes of TS $\mathcal{T}$

goal: on-the-fly construction of a fragment $\mathcal{T}_{\text{red}}$
by selecting action-sets $\text{ample}(s) \subseteq \text{Act}(s)$
and expanding only the α-successors of s
where $\alpha \in \text{ample}(s)$

requirements:

- stutter trace equivalence: $\mathcal{T} \stackrel{\Delta}{=} \mathcal{T}_{\text{red}}$
  hence: $\mathcal{T}$, $\mathcal{T}_{\text{red}}$ are $\mathbf{LTL}_{\setminus\bigcirc}$ equivalent

The ample set method [Peled '93]

given: syntactical representation of processes of TS $\mathcal{T}$

goal: on-the-fly construction of a fragment $\mathcal{T}_{\text{red}}$
by selecting action-sets $\text{ample}(s) \subseteq \text{Act}(s)$
and expanding only the α-successors of s
where $\alpha \in \text{ample}(s)$

requirements:

- stutter trace equivalence: $\mathcal{T} \stackrel{\Delta}{=} \mathcal{T}_{\text{red}}$
  hence: $\mathcal{T}$, $\mathcal{T}_{\text{red}}$ are $\mathbf{LTL}_{\setminus\bigcirc}$ equivalent
- $\mathcal{T}_{\text{red}}$ is smaller than $\mathcal{T}$

The ample set method [Peled '93]

given: syntactical representation of processes of TS $\mathcal{T}$

goal: on-the-fly construction of a fragment $\mathcal{T}_{\text{red}}$
by selecting action-sets $\text{ample}(s) \subseteq \text{Act}(s)$
and expanding only the α-successors of s
where $\alpha \in \text{ample}(s)$

requirements:

- stutter trace equivalence: $\mathcal{T} \stackrel{\Delta}{=} \mathcal{T}_{\text{red}}$
  hence: $\mathcal{T}$, $\mathcal{T}_{\text{red}}$ are $\mathbf{LTL}_{\setminus\bigcirc}$ equivalent
- $\mathcal{T}_{\text{red}}$ is smaller than $\mathcal{T}$
- efficient construction of $\mathcal{T}_{\text{red}}$ is possible

The reduced transition system $\mathcal{T}_{\text{red}}$

is a fragment of $\mathcal{T}$ that results from $\mathcal{T}$ by

- a DFS-based on-the-fly analysis and
- choosing ample sets $\text{ample}(s) \subseteq \mathit{Act}(s)$ for each expanded state,
- expanding only the α-successors of s where $\alpha \in \text{ample}(s)$

The reduced transition system $\mathcal{T}_{\text{red}}$

is a fragment of $\mathcal{T}$ that results from $\mathcal{T}$ by

- a DFS-based on-the-fly analysis and
- choosing ample sets $\text{ample}(s) \subseteq \mathit{Act}(s)$ for each expanded state,
- expanding only the α-successors of s where $\alpha \in \text{ample}(s)$

transition relation $\Rightarrow$ of $\mathcal{T}_{\text{red}}$ is given by:

$$\frac{s \xrightarrow{\alpha} s' \ \wedge \ \alpha \in \text{ample}(s)}{s \overset{\alpha}{\Longrightarrow} s'}$$

The reduced transition system $\mathcal{T}_{\text{red}}$

is a fragment of $\mathcal{T}$ that results from $\mathcal{T}$ by

. . . choosing ample sets $\text{ample}(s) \subseteq Act(s)$

transition relation $\Rightarrow$ of $\mathcal{T}_{\text{red}}$ is given by:

$$\frac{s \xrightarrow{\alpha} s' \ \wedge \ \alpha \in \text{ample}(s)}{s \overset{\alpha}{\Longrightarrow} s'}$$

$\text{ample}(s) = \{\alpha, \beta\}$

The reduced transition system $\mathcal{T}_{red}$

is a fragment of $\mathcal{T}$ that results from $\mathcal{T}$ by

. . . choosing ample sets $\text{ample}(s) \subseteq Act(s)$

transition relation $\Rightarrow$ of $\mathcal{T}_{red}$ is given by:

$$\frac{s \xrightarrow{\alpha} s' \ \wedge \ \alpha \in \text{ample}(s)}{s \stackrel{\alpha}{\Longrightarrow} s'}$$

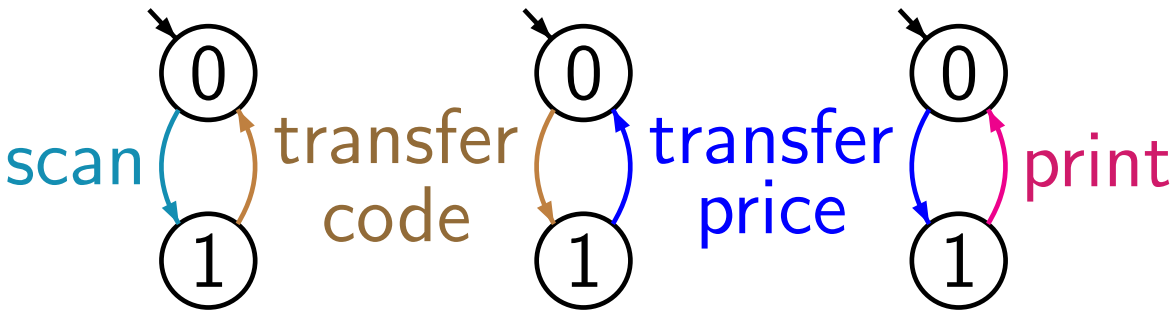
$\text{ample}(s) = \{\alpha, \beta\}$

state space S_{red} of $\mathcal{T}_{red}$: all states that are reachable

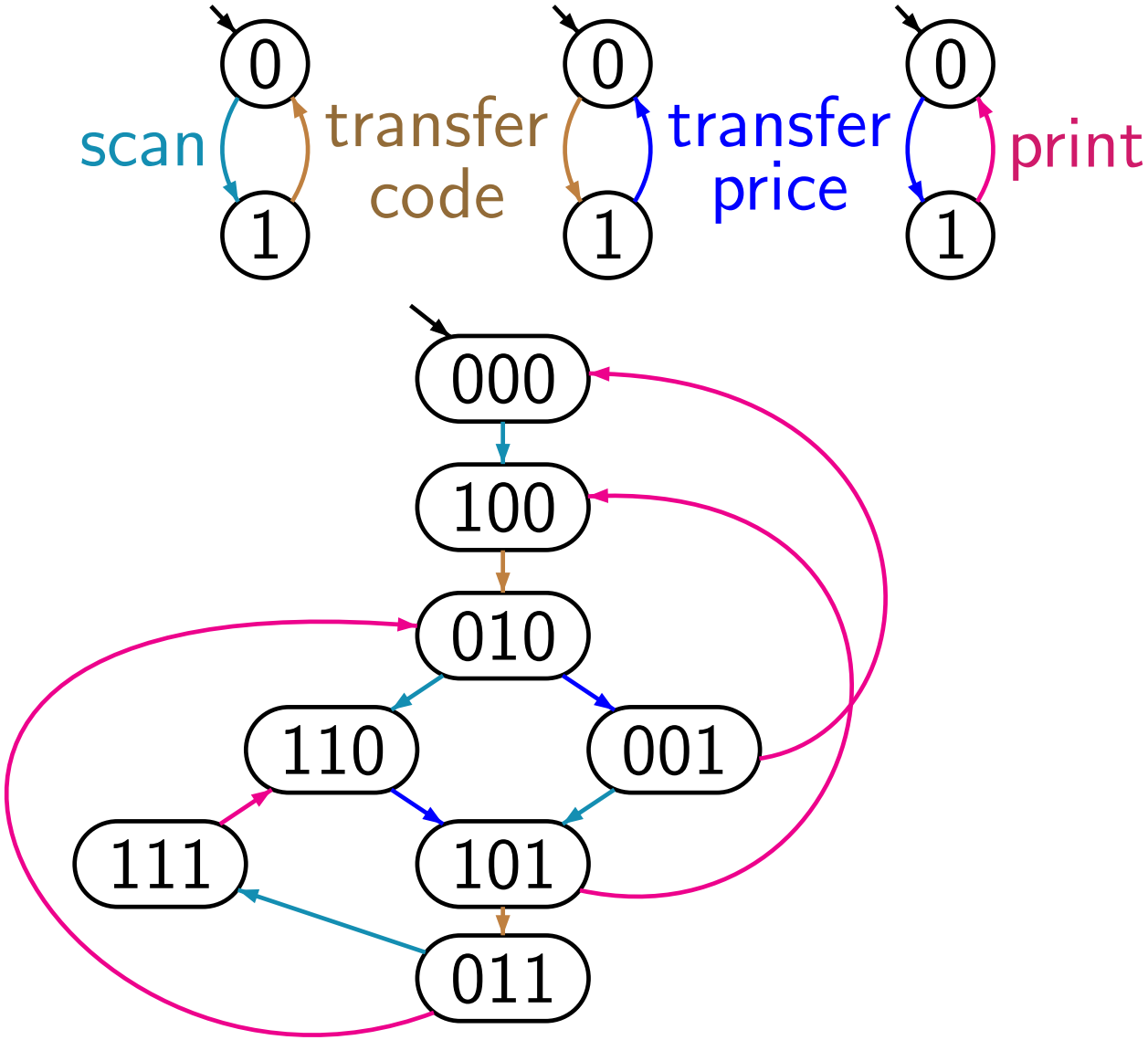
from the initial states in $\mathcal{T}$ via $\Rightarrow$

Booking system

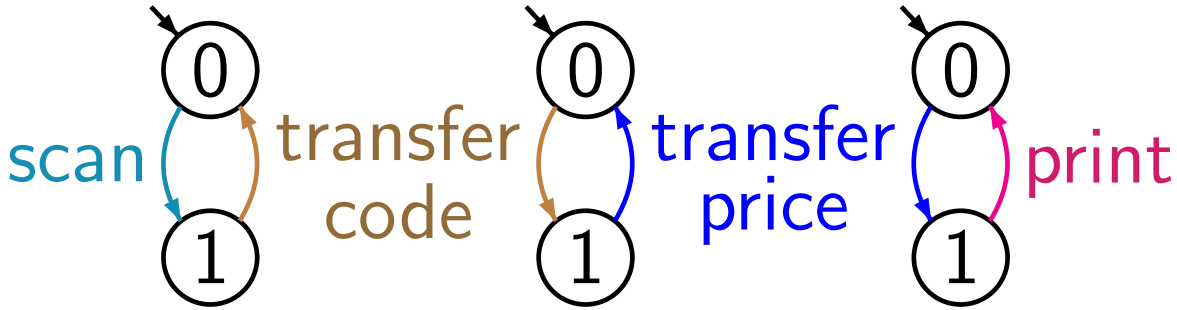
$$\mathcal{T} = \text{SCL} \parallel \text{BP} \parallel \text{Printer}$$

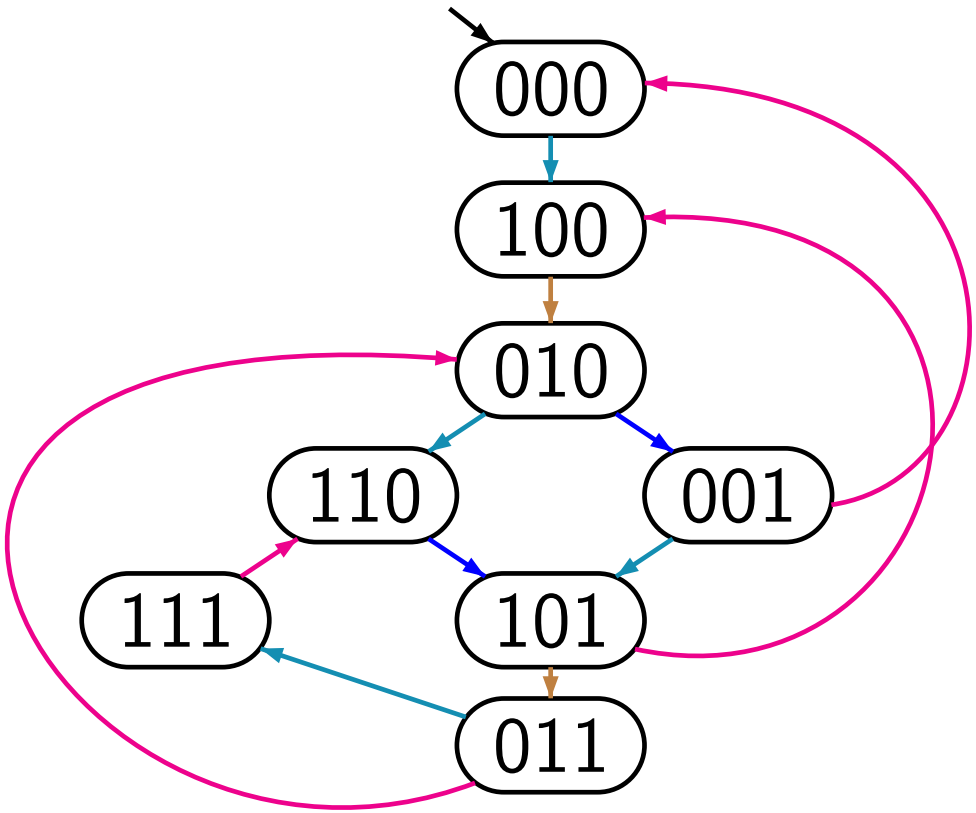


Booking system



Booking system





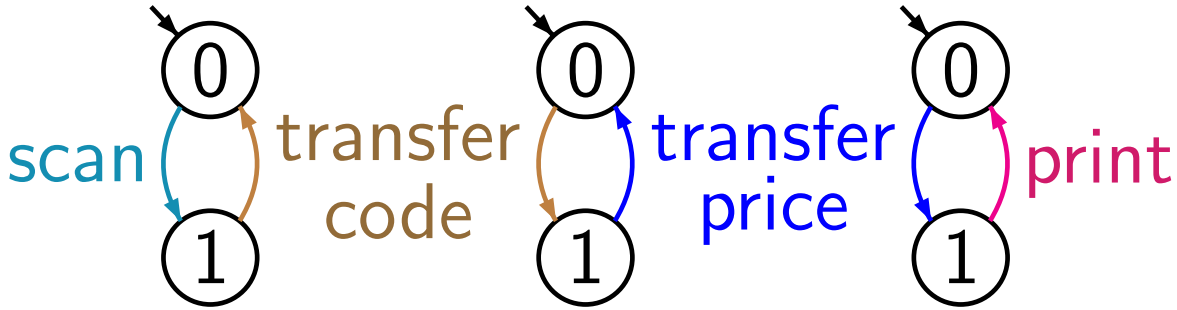
independent, i.e., concurrently executable, actions:

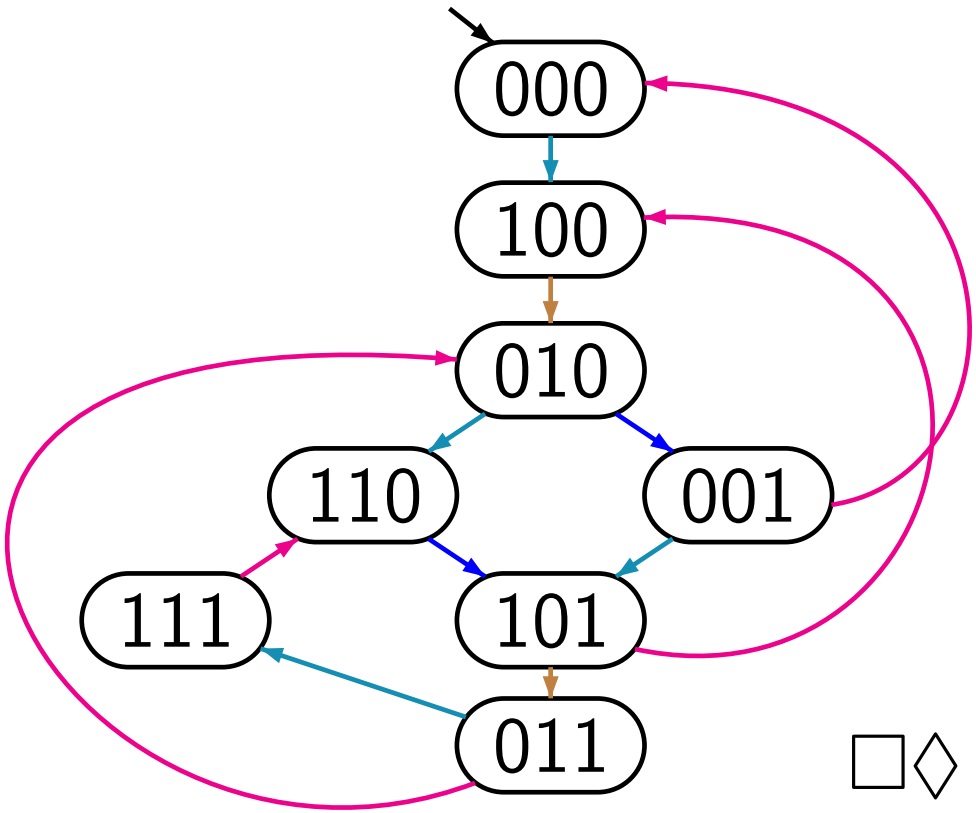
scan - transfer price

transfer code - print

scan - print

Booking system





independent actions:

scan - transf. price

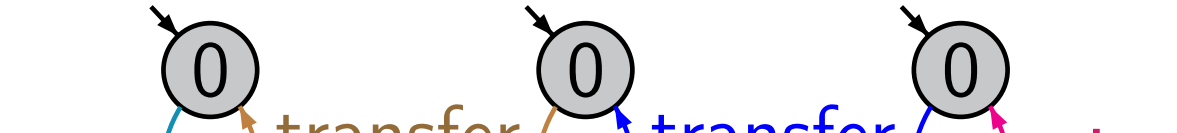
transfer code - print

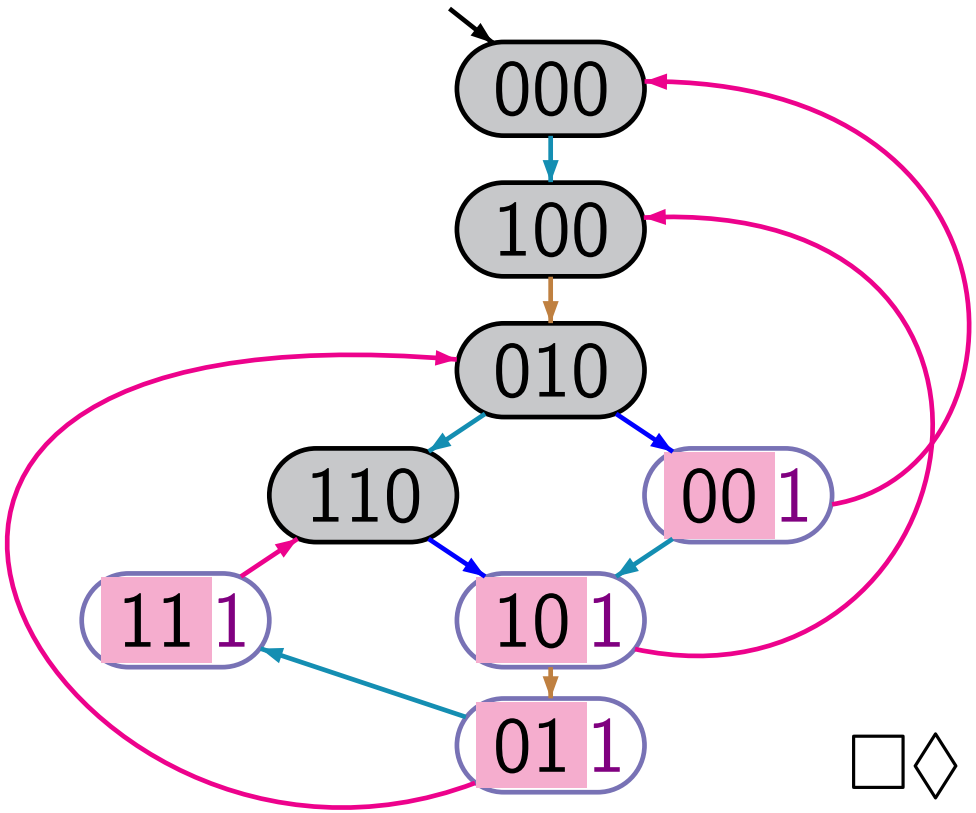
scan - print

LTL$_{\setminus\bigcirc}$ property:

$\square\Diamond$ "printer is in state 1"

Booking system





independent actions:

scan - transf. price

transfer code - print

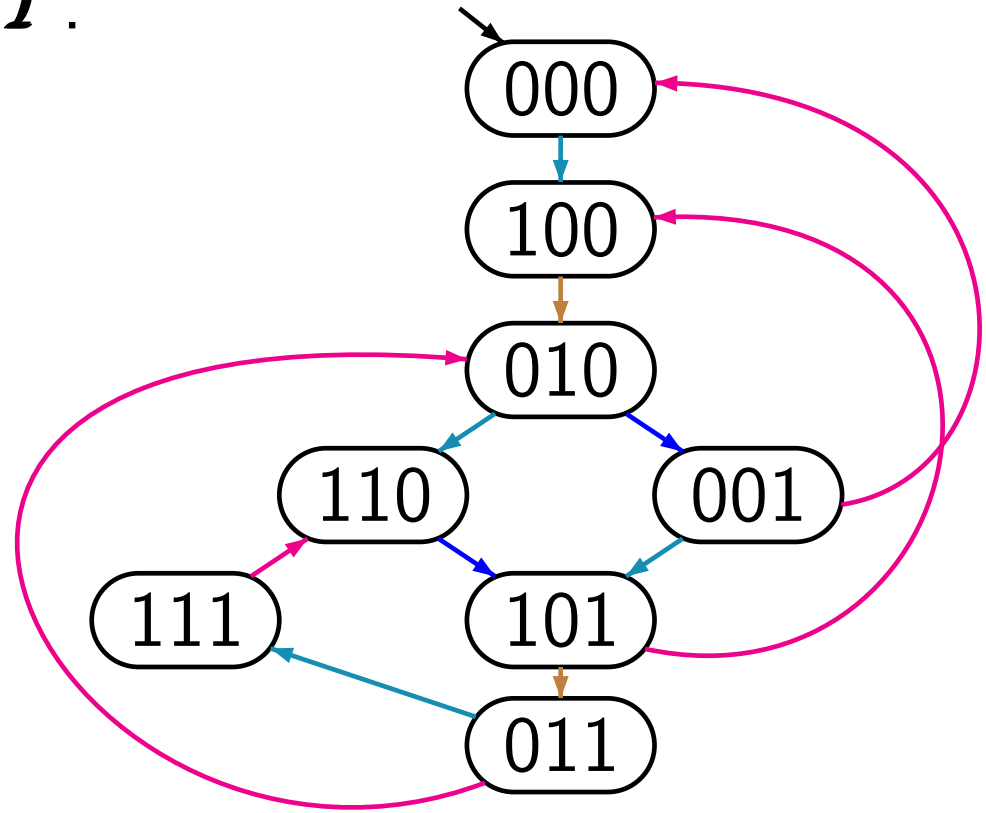
scan - print

LTL$_{\setminus\bigcirc}$ property:

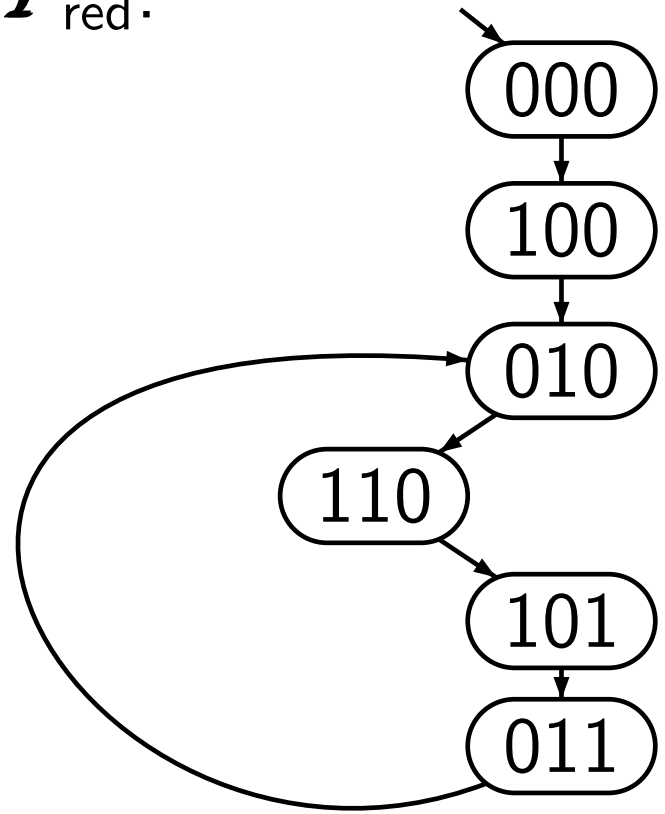
$\square\Diamond$ "printer is in state 1"

Booking system

$\mathcal{T}$:

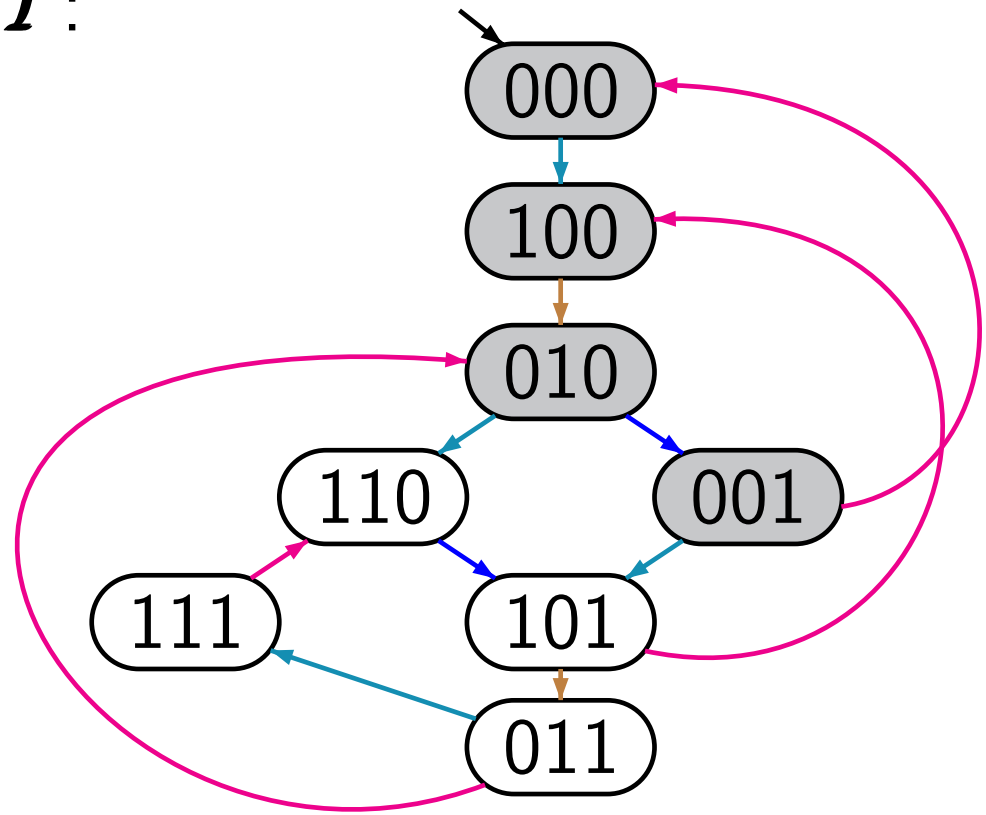


$\mathcal{T}_{\text{red}}$:



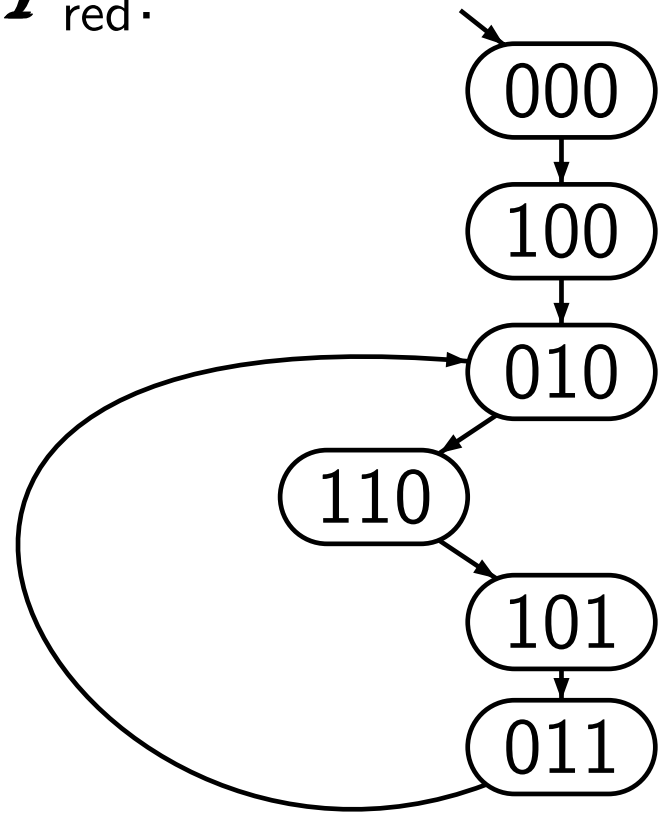
Booking system

$\mathcal{T}$:



scan
code
price
print
scan
code
...

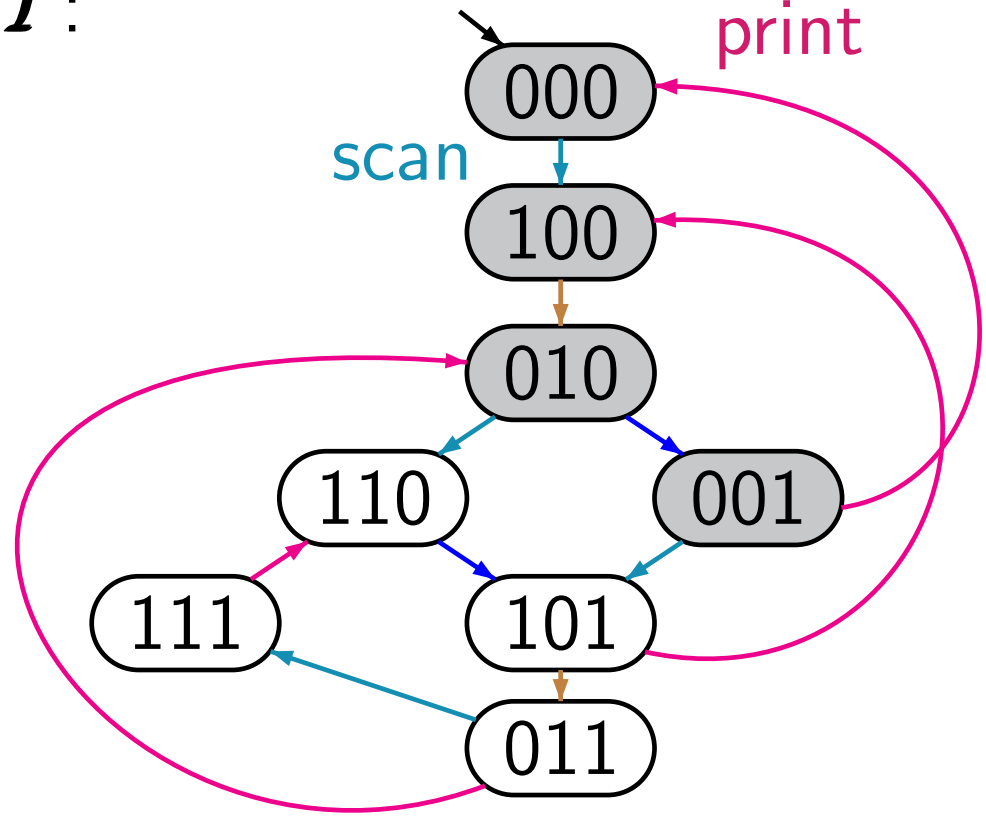
$\mathcal{T}_{\text{red}}$:



scan
code
scan
price
code
print
...

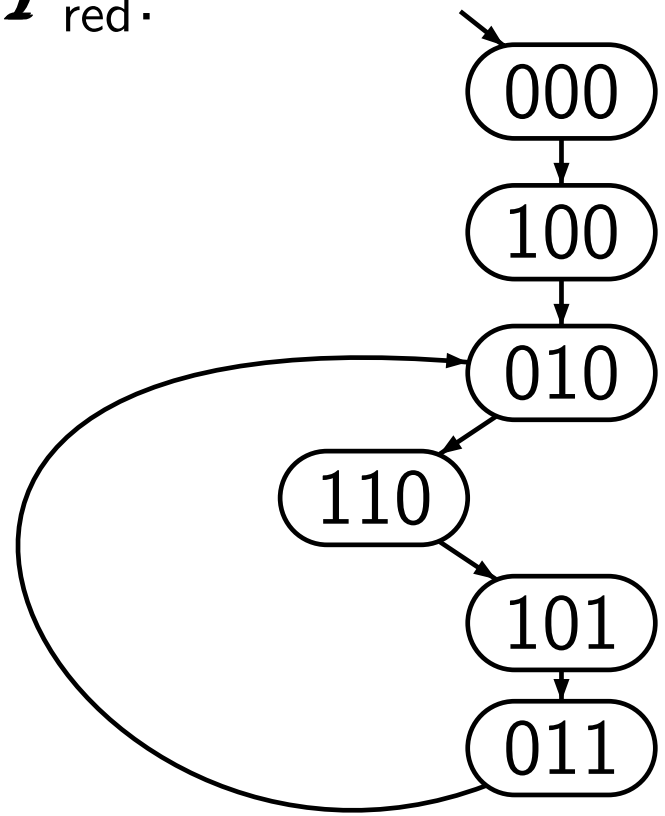
Booking system

$\mathcal{T}$:



scan
code
price
print
scan
code
...

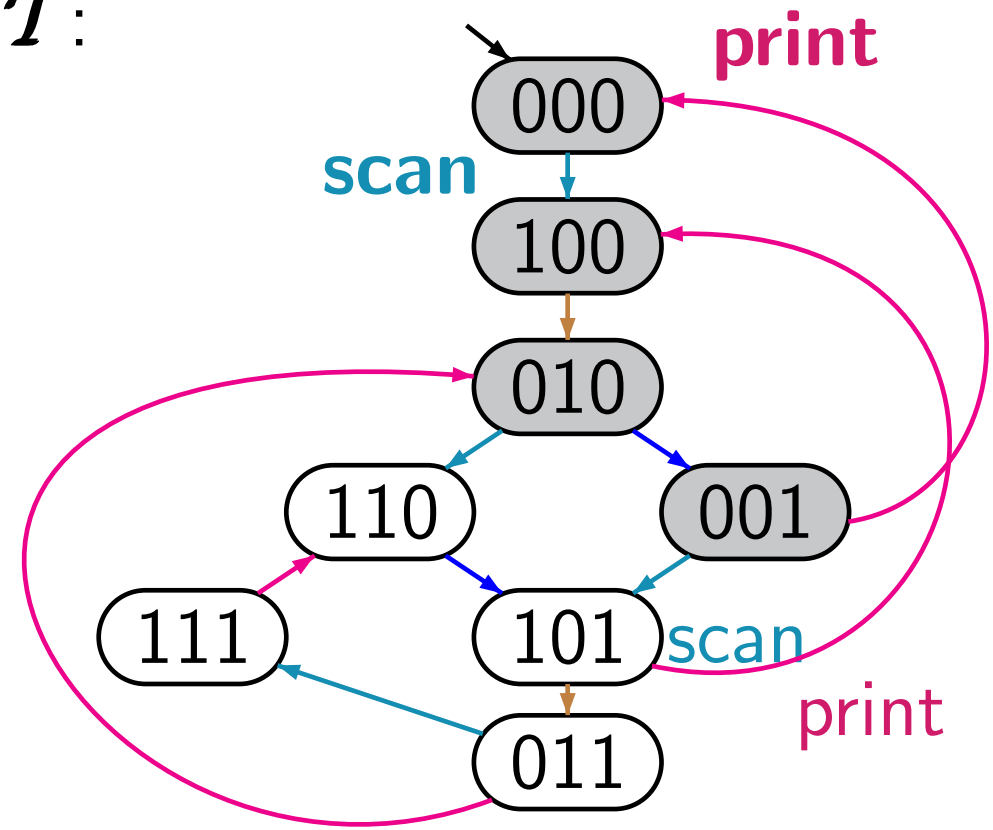
$\mathcal{T}_{\text{red}}$:



scan
code
scan
price
code
print
...

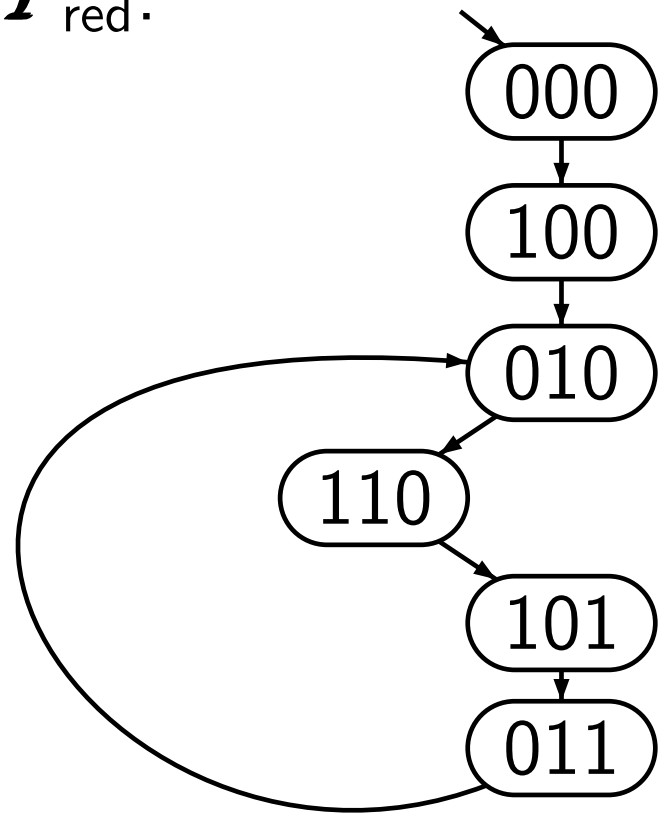
Booking system

$\mathcal{T}$:



scan
code
price
print
scan
code
...

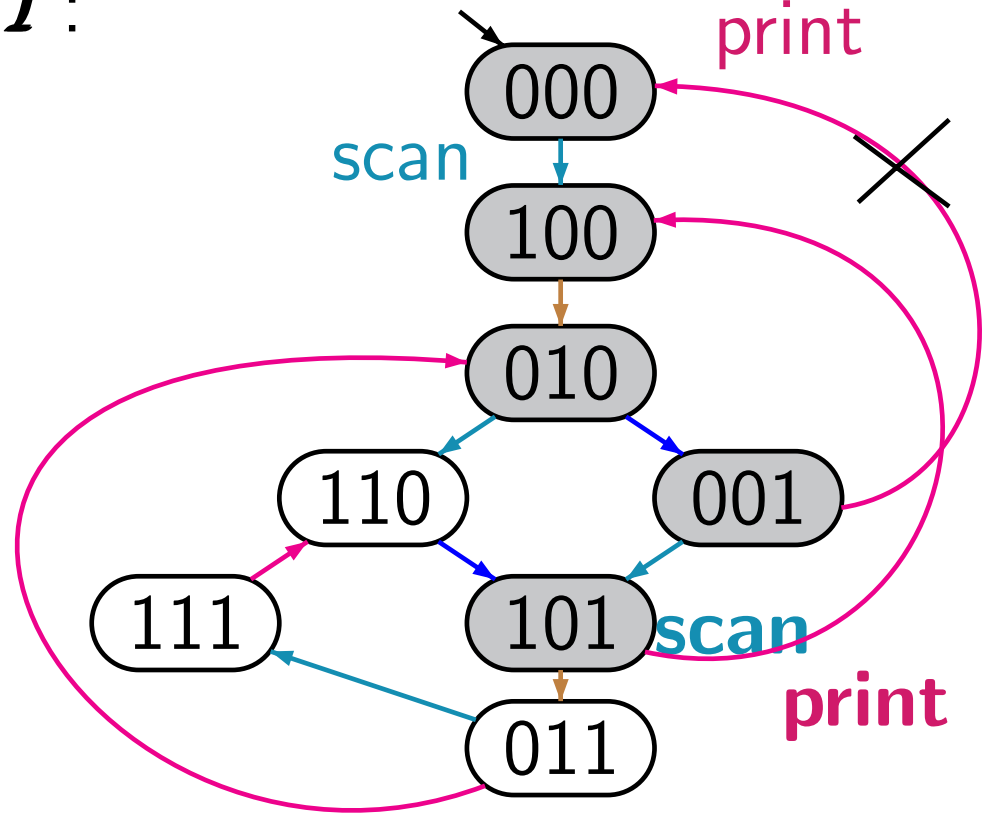
$\mathcal{T}_{\text{red}}$:



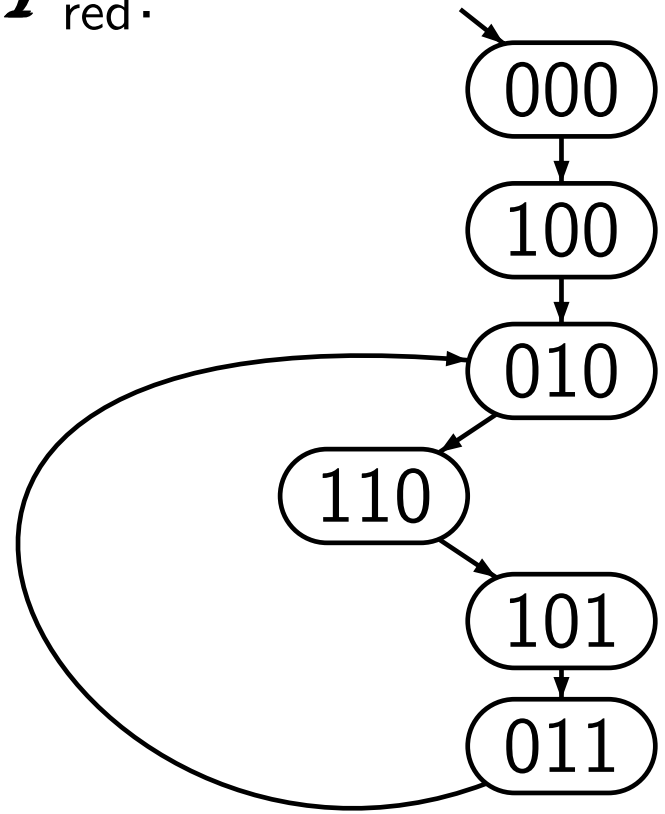
scan
code
scan
price
code
print
...

Booking system

$\mathcal{T}$:



$\mathcal{T}_{\text{red}}$:



scan
code
price
print
scan
code
...

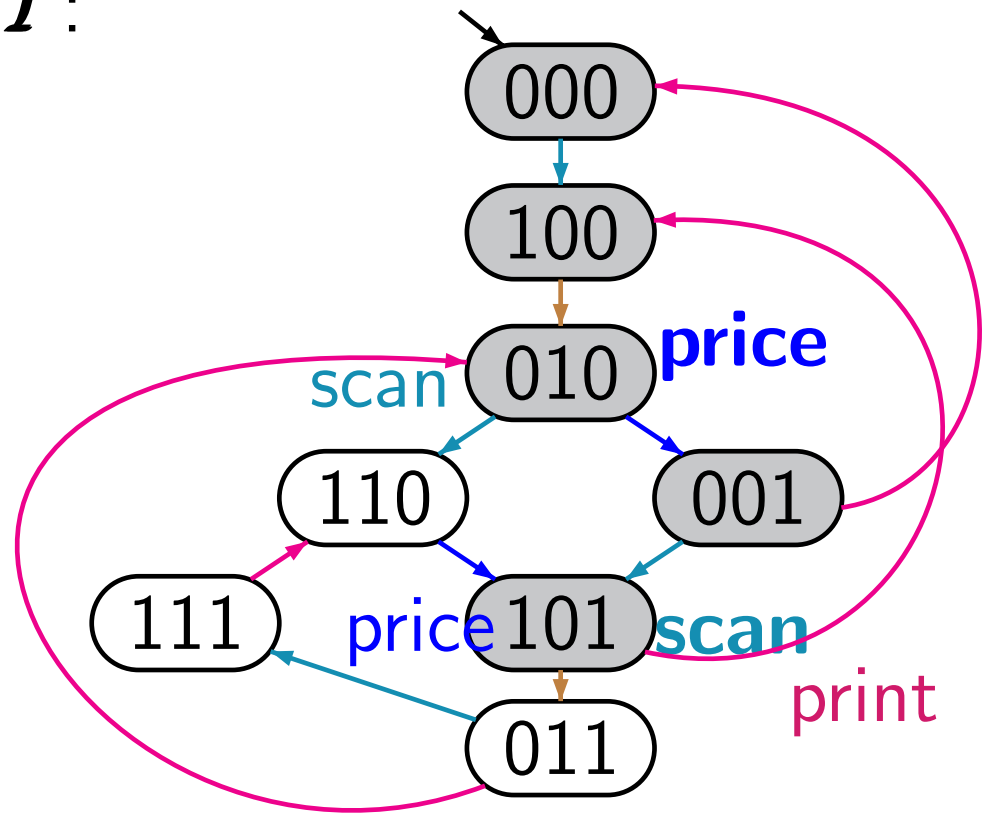
$\rightsquigarrow$

scan
code
price
scan
print
code
...

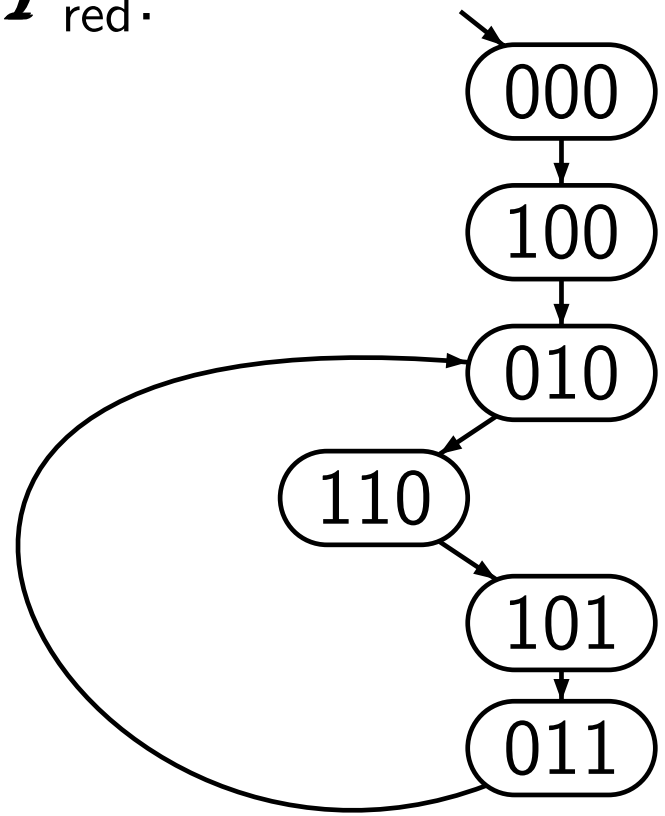
scan
code
scan
price
code
print
...

Booking system

$\mathcal{T}$:



$\mathcal{T}_{\text{red}}$:



scan
code
price
print
scan
code
...

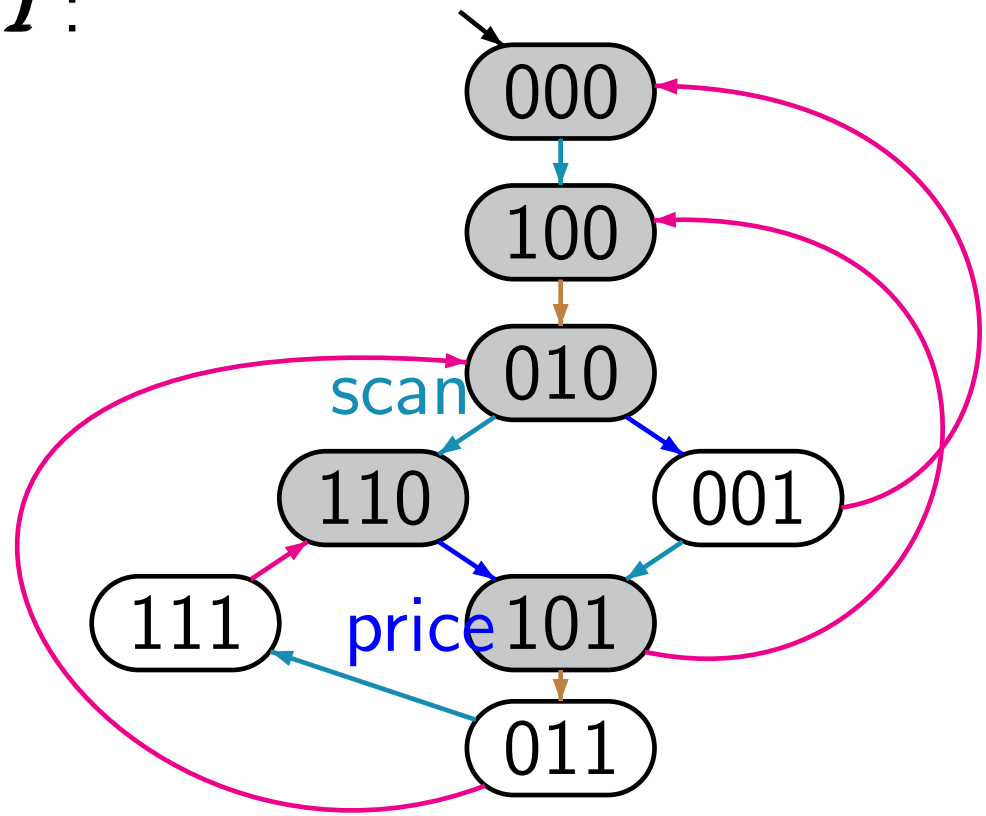
$\rightsquigarrow$

scan
code
price
scan
print
code
...

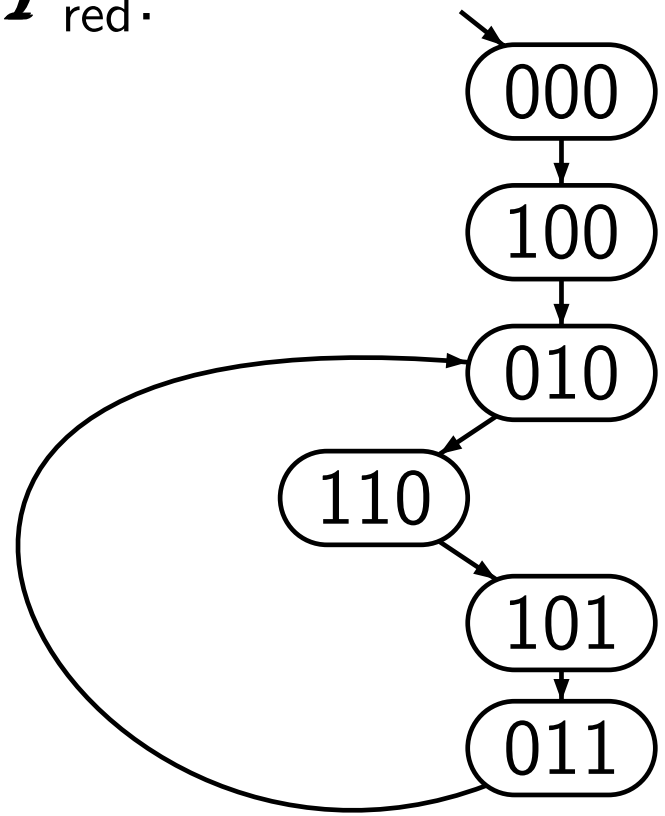
scan
code
scan
price
code
print
...

Booking system

$\mathcal{T}$:



$\mathcal{T}_{\text{red}}$:



scan
code
price
print
scan
code
...

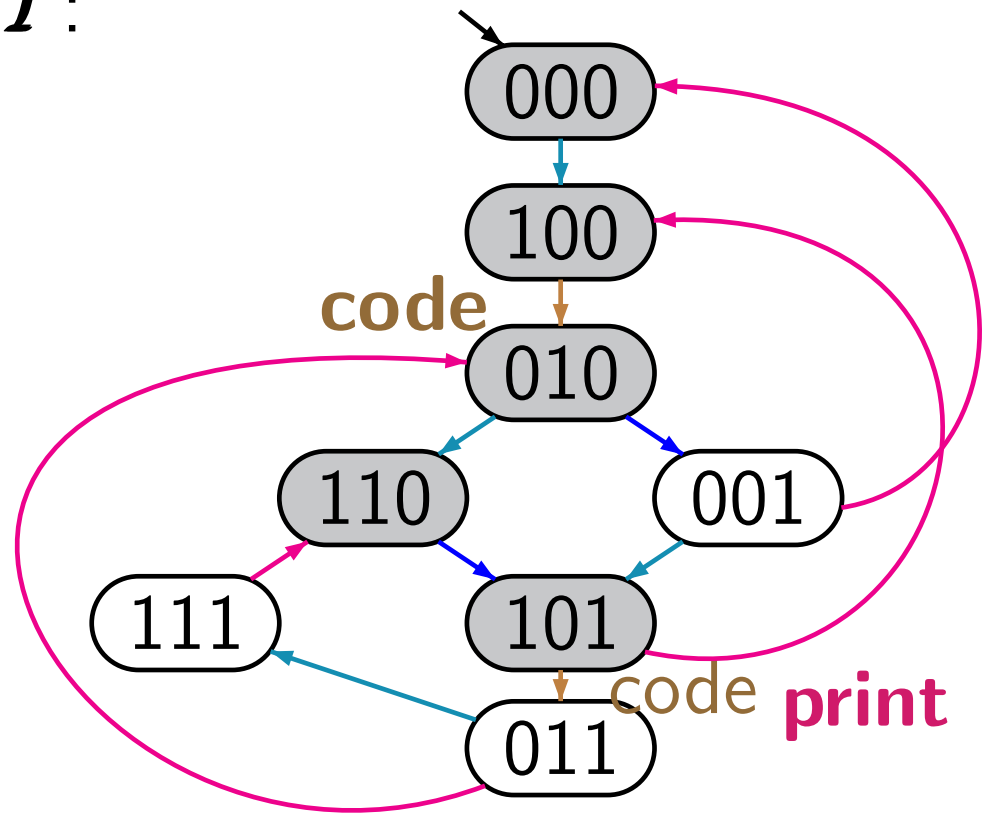
$\rightsquigarrow$

scan
code
scan
price
print
code
...

scan
code
scan
price
code
print
...

Booking system

$\mathcal{T}$:

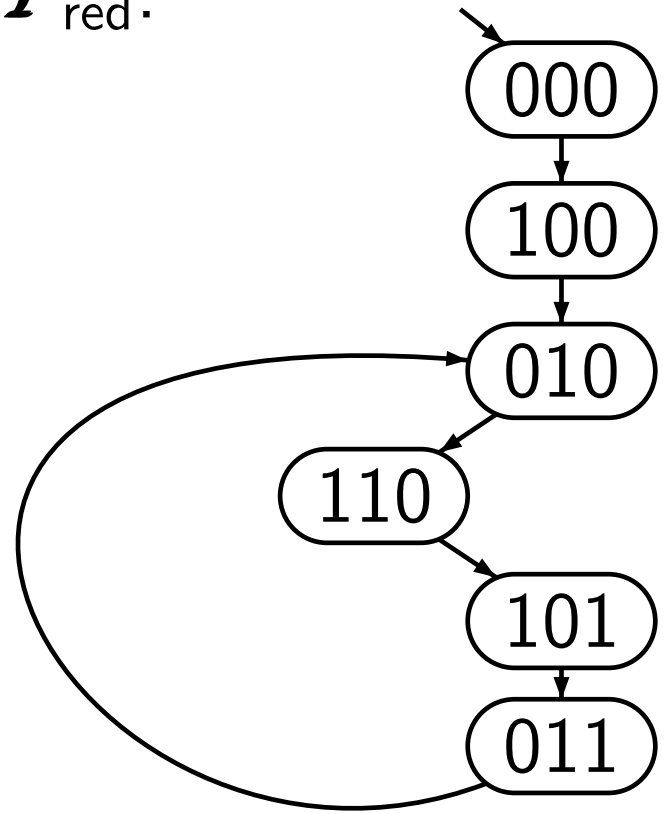


scan
code
price
print
scan
code
...

$\rightsquigarrow$

scan
code
scan
price
print
code
...

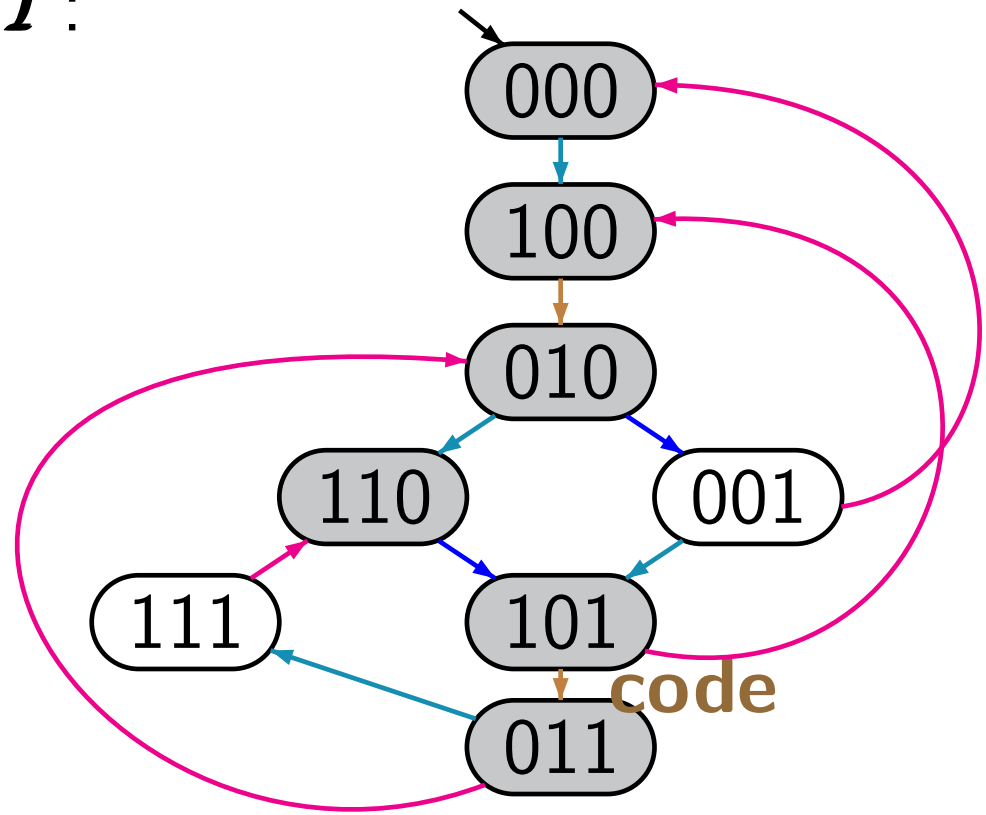
$\mathcal{T}_{\text{red}}$:



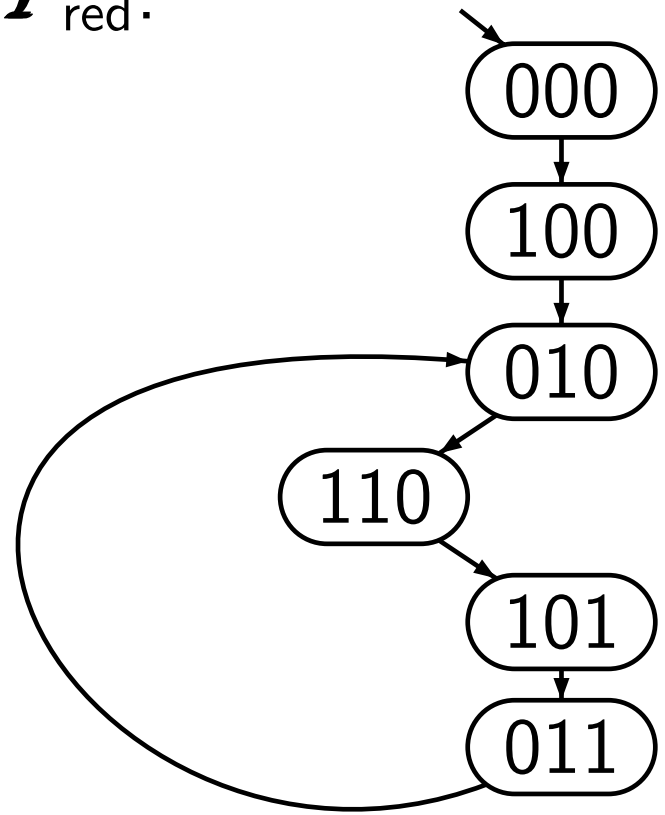
scan
code
scan
price
code
print
...

Booking system

$\mathcal{T}$:



$\mathcal{T}_{\text{red}}$:



scan
code
price
print
scan
code
...

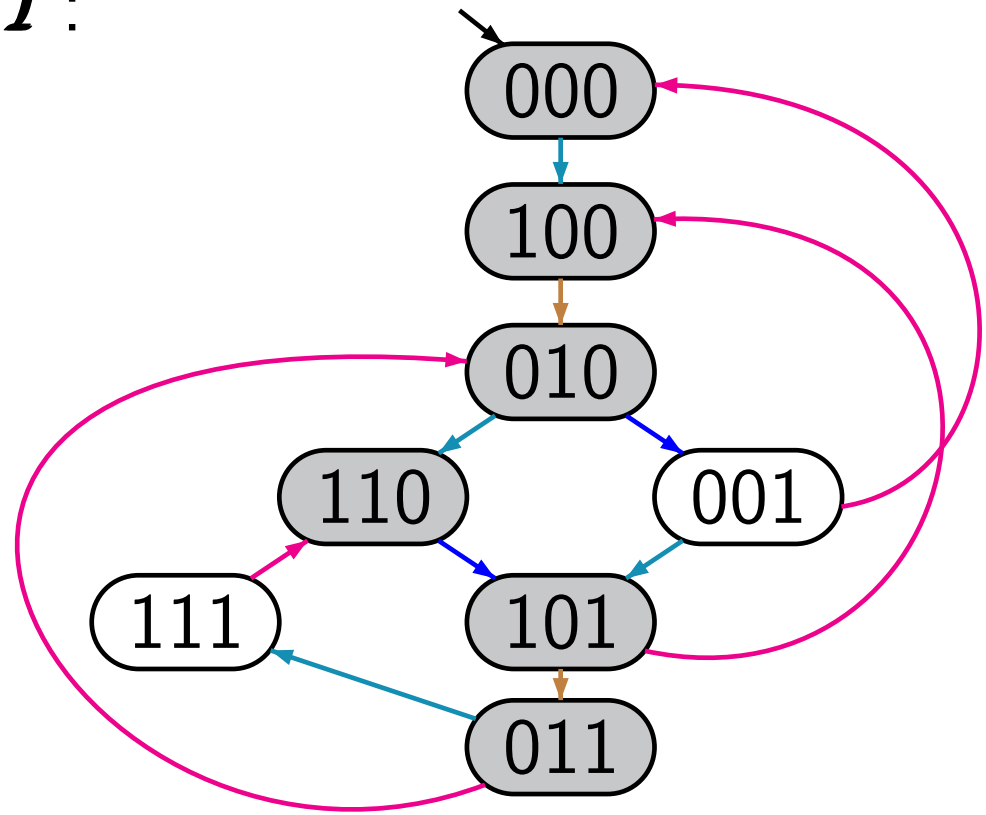
$\rightsquigarrow$

scan
code
scan
price
code
print
...

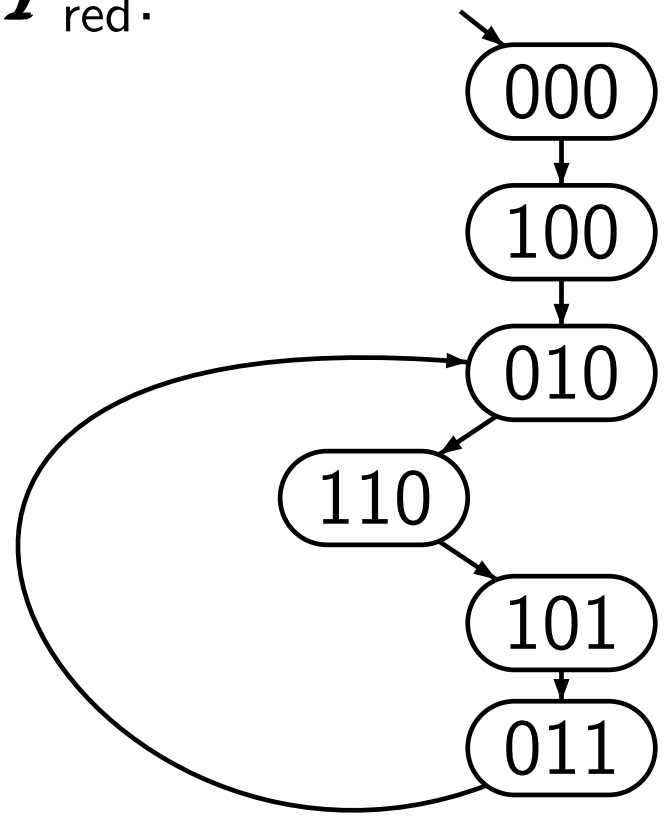
scan
code
scan
price
code
print
...

Booking system

$\mathcal{T}$:



$\mathcal{T}_{\text{red}}$:



scan
code
price
print
scan
code
...

$\rightsquigarrow$

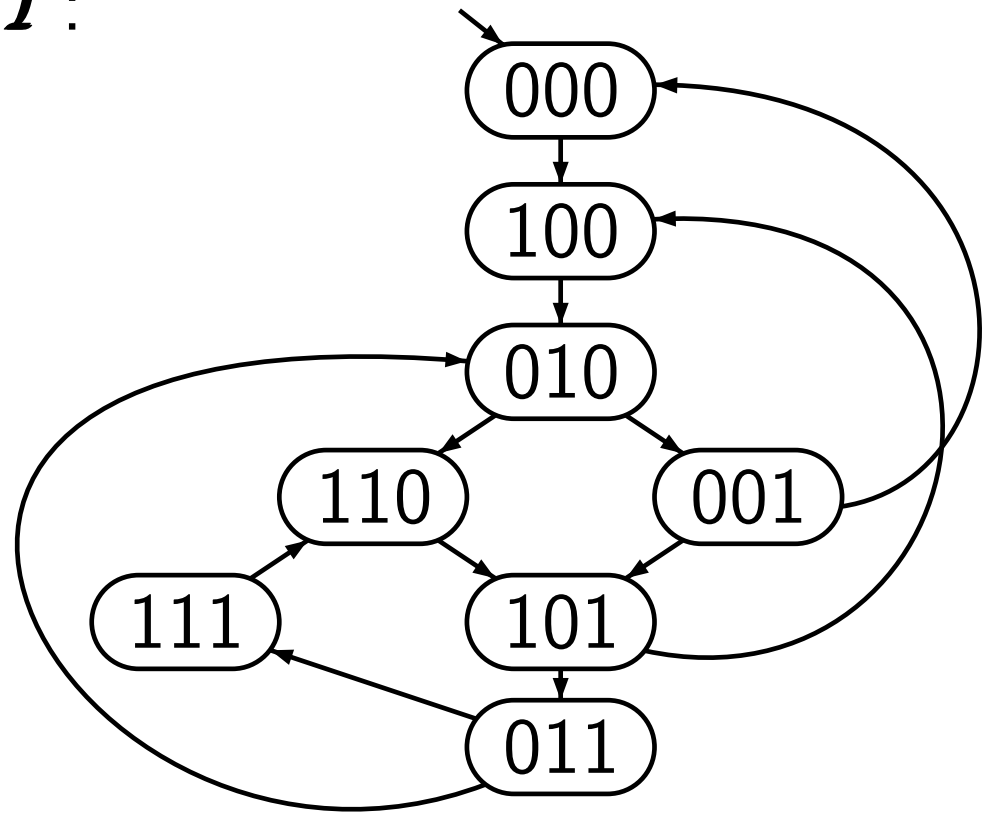
scan
code
scan
price
code
print
...

$=$

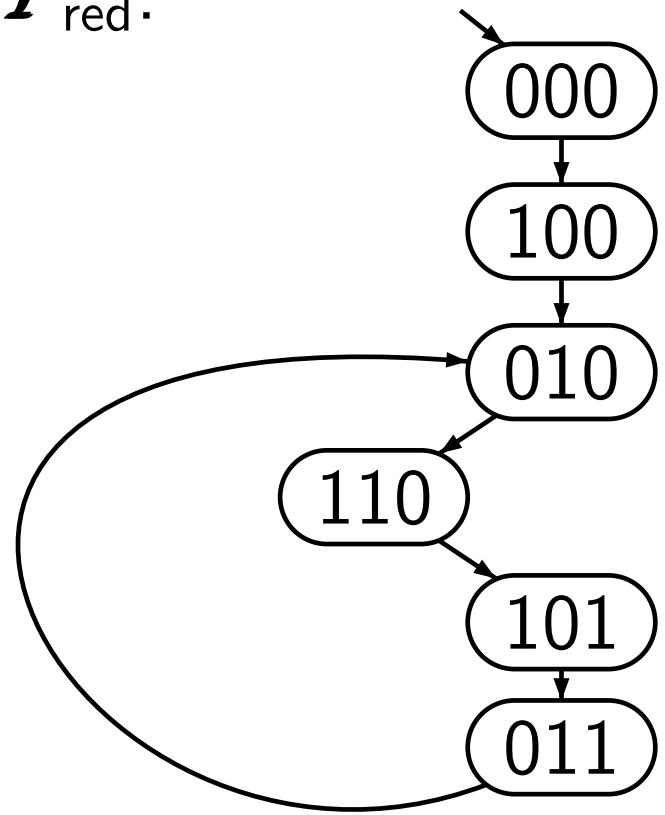
scan
code
scan
price
code
print
...

AP = {"printer in state 1"}

$\mathcal{T}$:



$\mathcal{T}_{\text{red}}$:



scan
code
price
print
scan
code

$\rightsquigarrow$

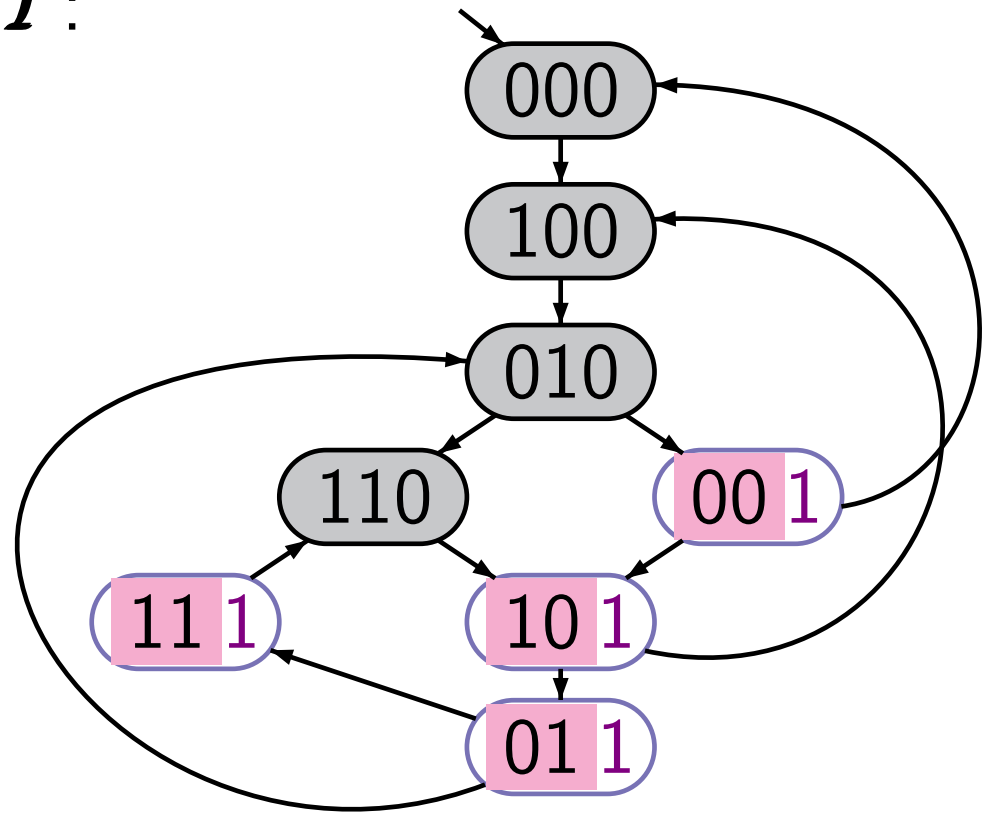
scan
code
price
scan
print
code

$\rightsquigarrow$

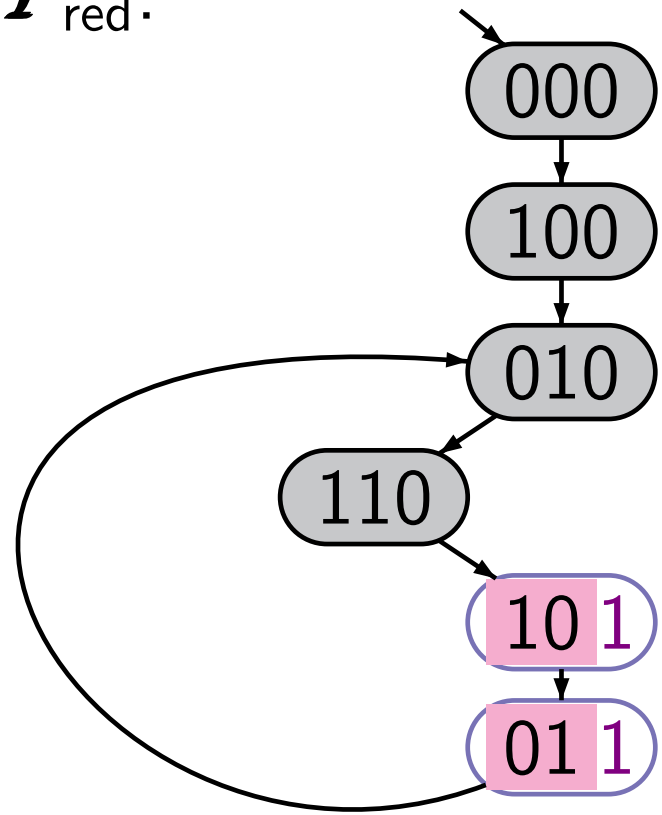
scan
code
scan
price
print
code

AP = {"printer in state 1"}

$\mathcal{T}$:



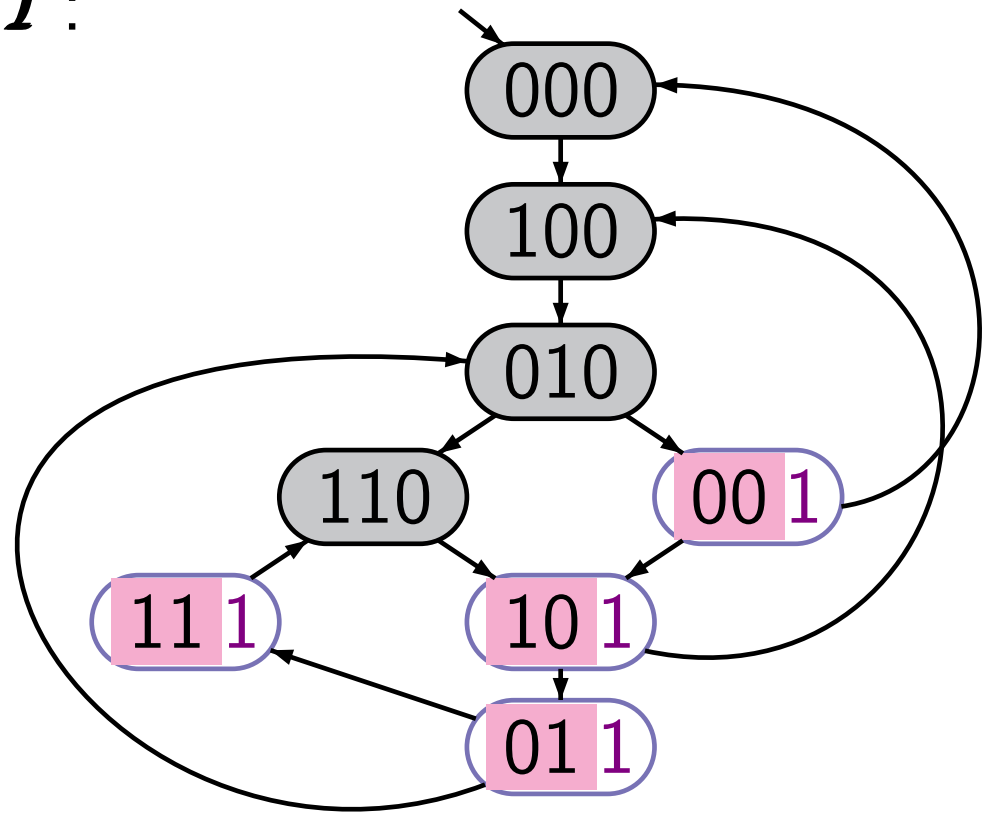
$\mathcal{T}_{\text{red}}$:



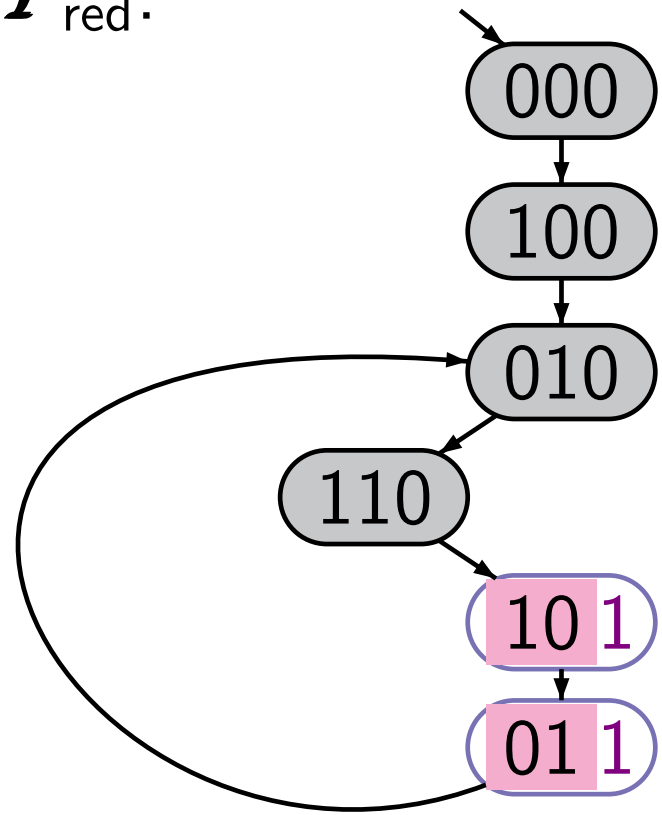
scan	∅		scan	∅		scan	∅
code	∅		code	∅		code	∅
price	∅		price	∅		scan	∅
print	{1}	⇝	scan	{1}	⇝	price	∅
scan	∅		print	{1}		print	{1}
code	∅		code	∅		code	∅

... stuttering equivalent ..

$\mathcal{T}$:



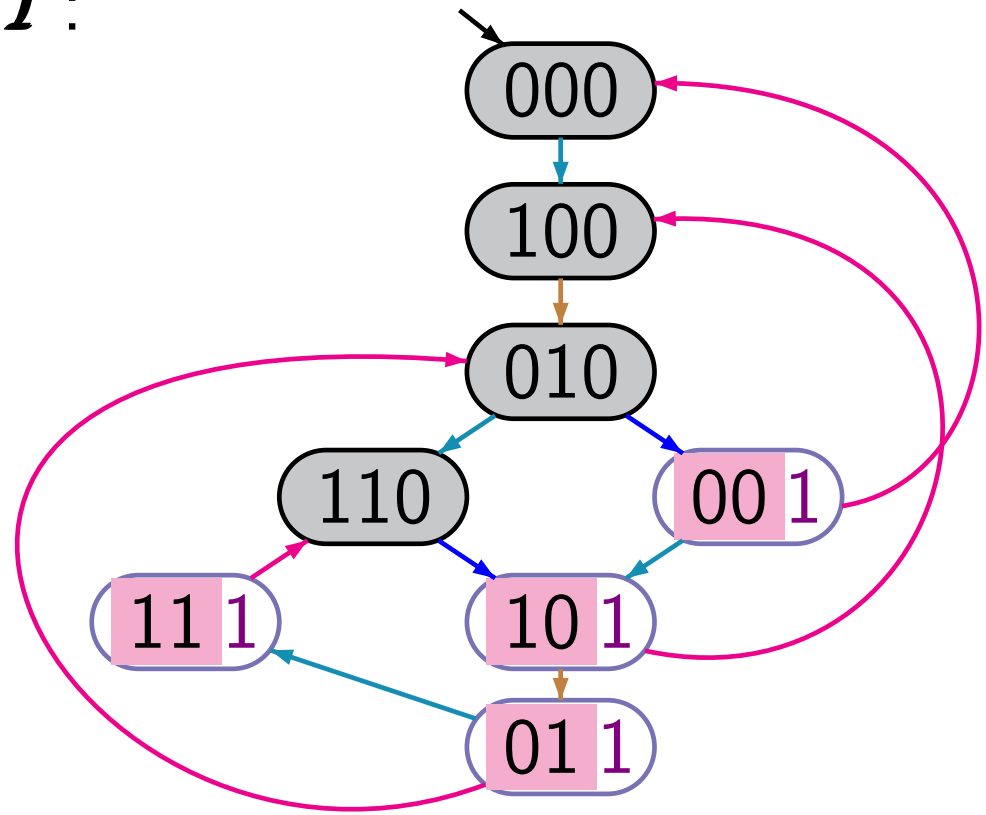
$\mathcal{T}_{\text{red}}$:



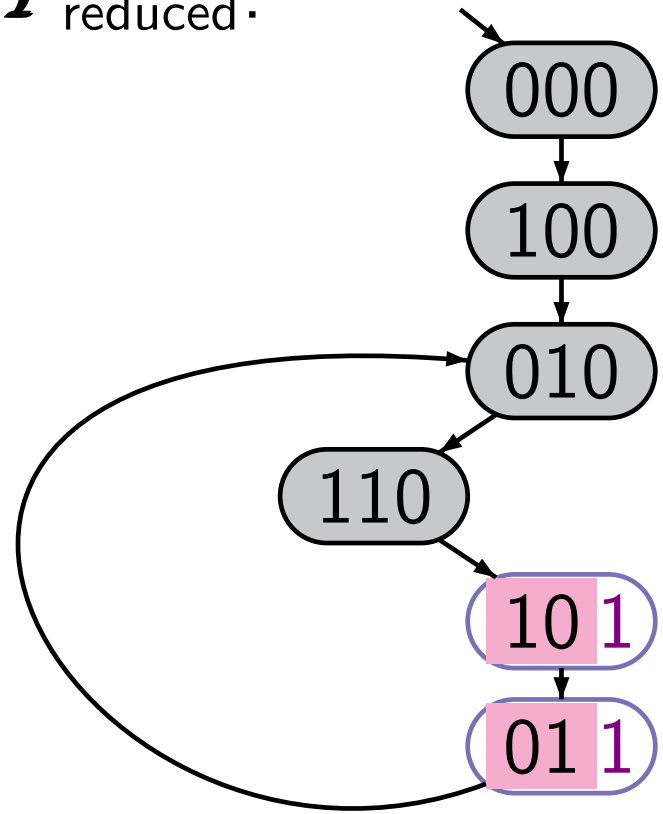
scan	$\emptyset$		scan	$\emptyset$		scan	$\emptyset$
code	$\emptyset$		code	$\emptyset$		code	$\emptyset$
price	$\emptyset$		price	$\emptyset$		scan	$\emptyset$
print	$\{1\}$	$\rightsquigarrow$	scan	$\{1\}$	$\rightsquigarrow$	price	$\emptyset$
scan	$\emptyset$		print	$\{1\}$		print	$\{1\}$
code	$\emptyset$		code	$\emptyset$		code	$\emptyset$

Booking system

$\mathcal{T}$:

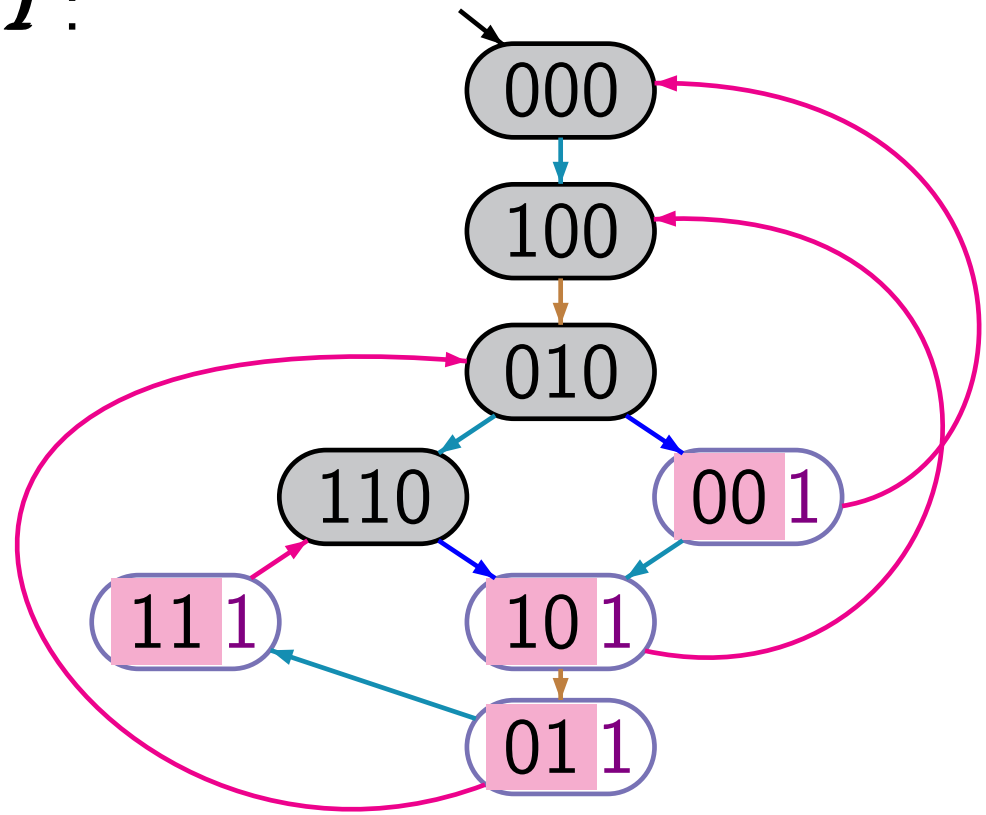


$\mathcal{T}_{\text{reduced}}$:

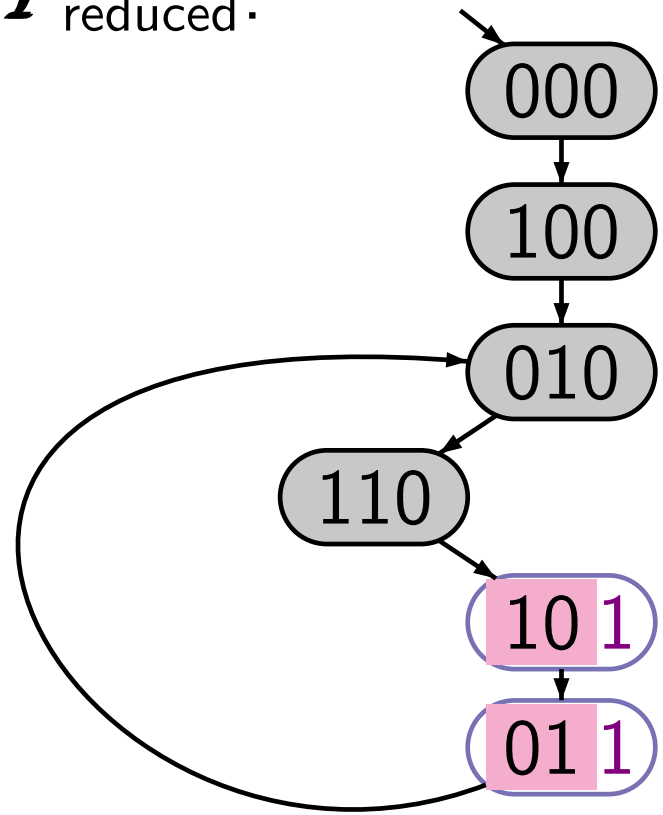


Booking system

$\mathcal{T}$:



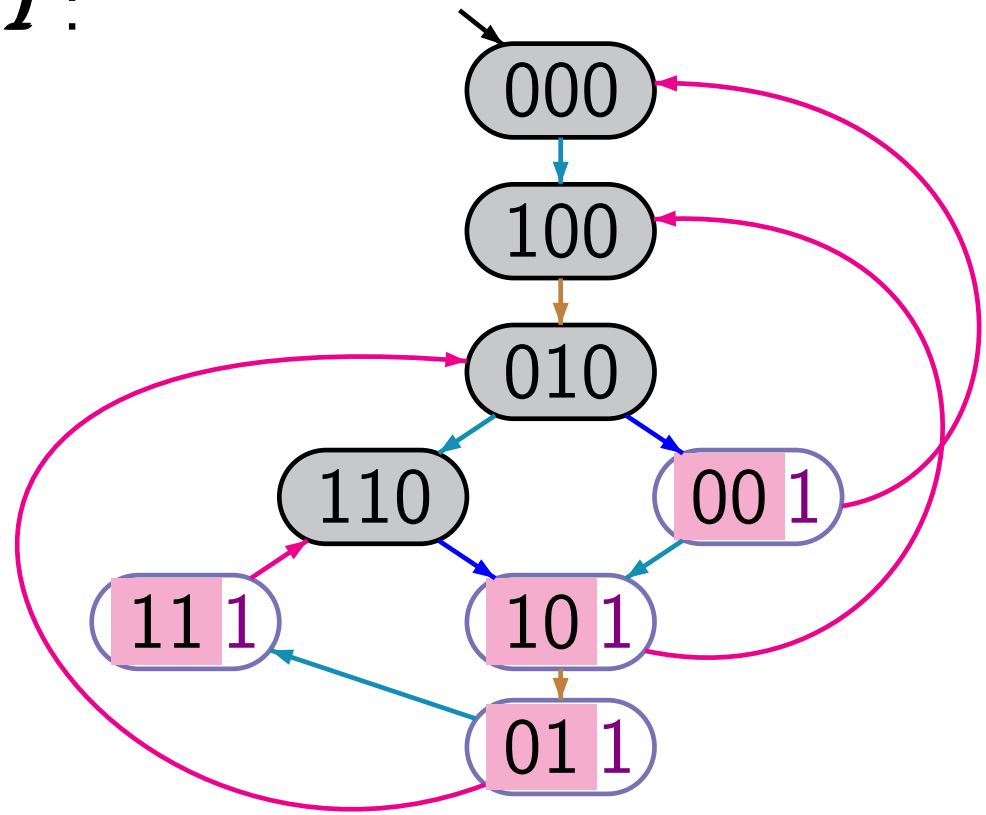
$\mathcal{T}_{\text{reduced}}$:



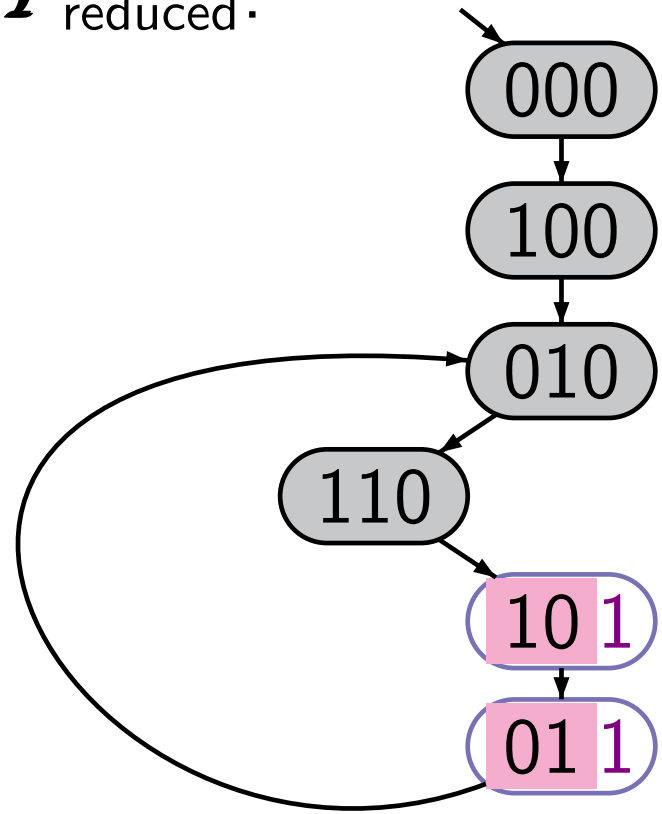
$\mathcal{T} \triangleq \mathcal{T}_{\text{red}}$

Booking system

$\mathcal{T}$:



$\mathcal{T}_{\text{reduced}}$:



$\mathcal{T} \triangleq \mathcal{T}_{\text{red}}$ hence: $\mathcal{T}_{\text{red}} \models \varphi$ implies $\mathcal{T} \models \varphi$ where

$\varphi = \Box\Diamond$ "printer is in control state 1"

Action-determinism

Let $\mathcal{T} = (\mathbf{S}, Act, \rightarrow, \mathbf{S}_0, \mathrm{AP}, \mathrm{L})$ be a TS.

For state $\mathbf{s}$:

$$Act(\mathbf{s}) = \{ \alpha \in Act : \exists \mathbf{t} \in \mathbf{S} \text{ s.t. } \mathbf{s} \xrightarrow{\alpha} \mathbf{t} \}$$

$\mathcal{T}$ is called action-deterministic iff for all states $\mathbf{s}$ and all actions $\alpha \in Act(\mathbf{s})$:

$$| \{ \mathbf{t} \in \mathbf{S} : \mathbf{s} \xrightarrow{\alpha} \mathbf{t} \} | \leq 1$$

notation: if $\alpha \in Act(\mathbf{s})$ then

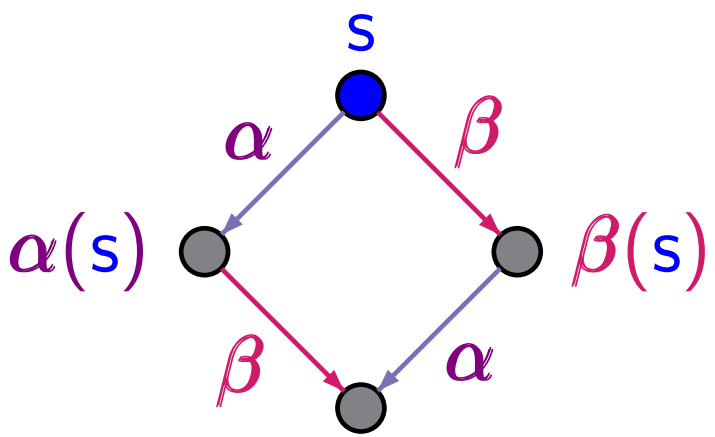
$$\alpha(\mathbf{s}) = \text{unique state } \mathbf{t} \text{ s.t. } \mathbf{s} \xrightarrow{\alpha} \mathbf{t}$$

Independence of actions

Let $\mathcal{T}$ be an action-deterministic transition system with action-set Act, and $\alpha, \beta \in Act$.

α, β are called independent in $\mathcal{T}$ if for all states s s.t. $\alpha, \beta \in Act(s)$:

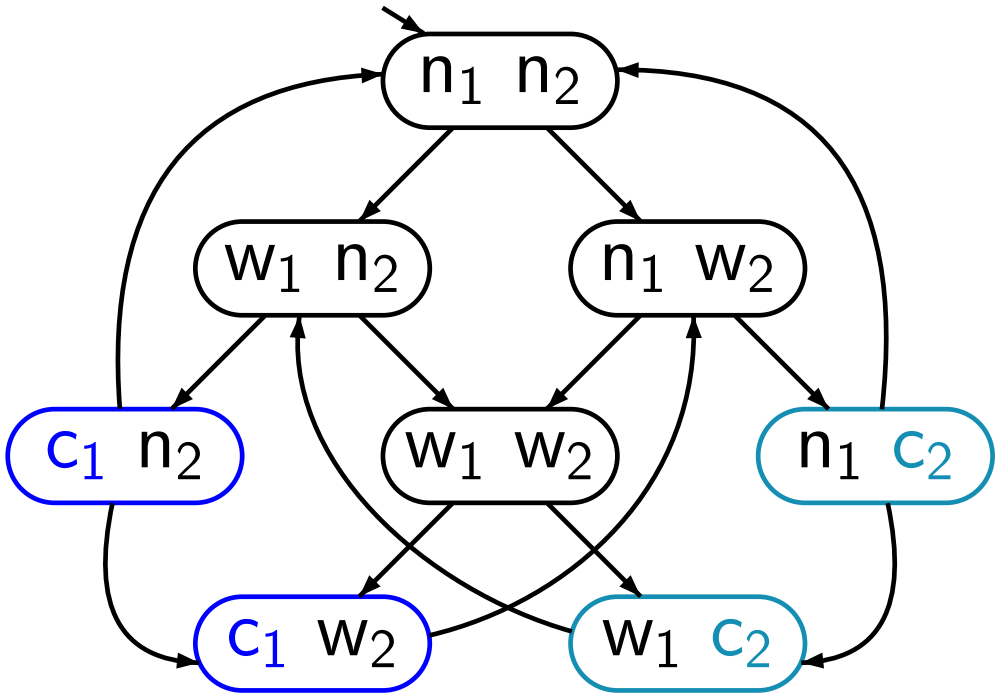
1. $\beta \in Act(\alpha(s))$
2. $\alpha \in Act(\beta(s))$
3. $\beta(\alpha(s)) = \alpha(\beta(s))$



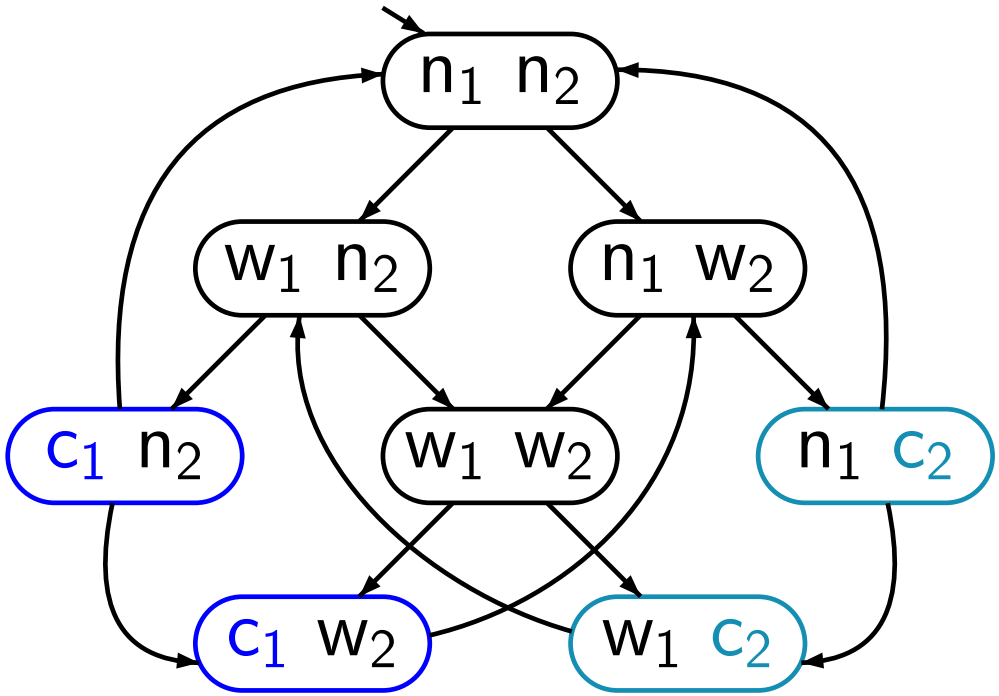
$\beta(\alpha(s)) = \alpha(\beta(s))$

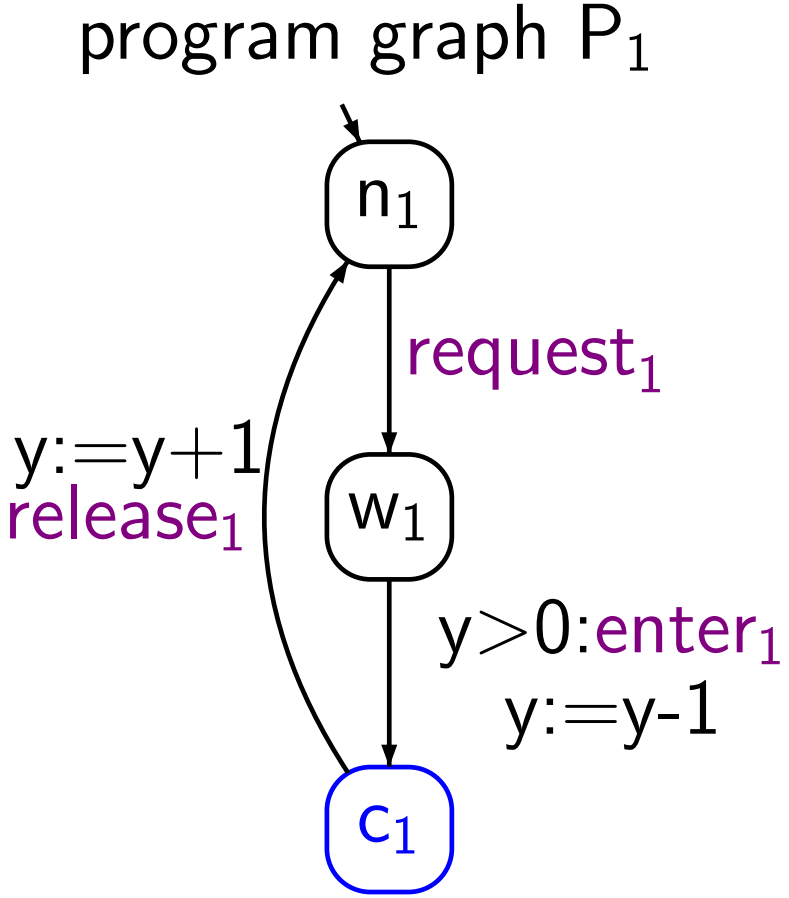
Mutual exclusion with semaphore

LTL3.4-12

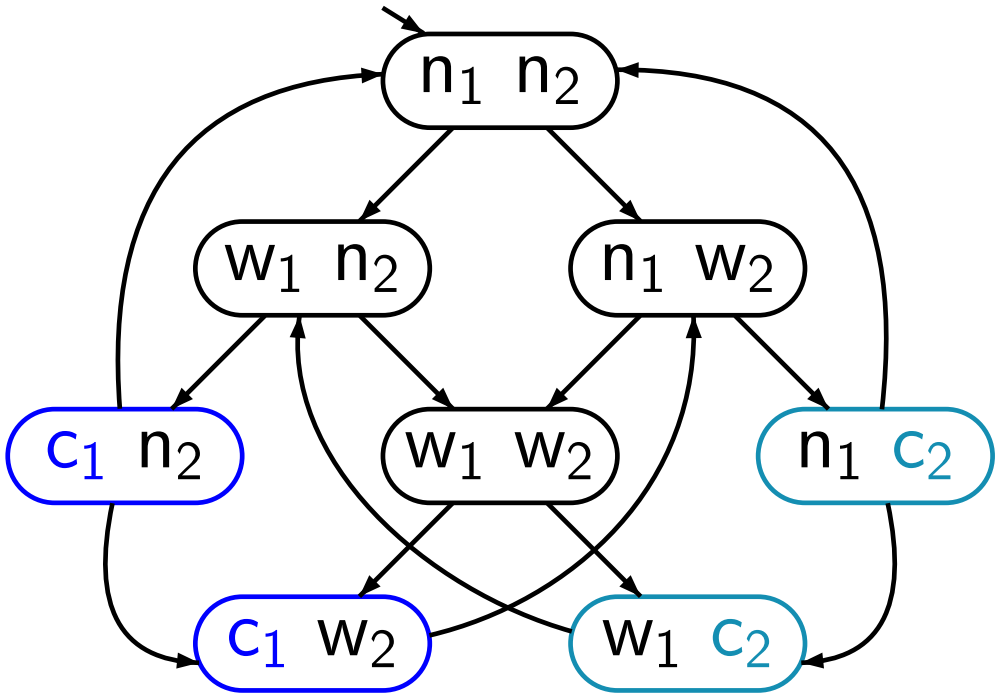


Mutual exclusion with semaphore

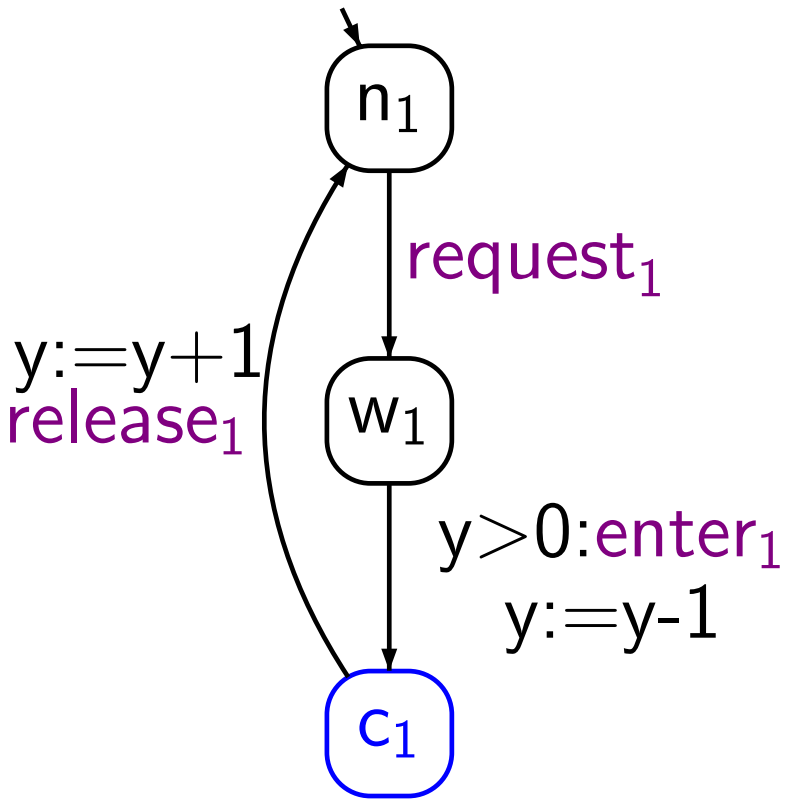




Mutual exclusion with semaphore



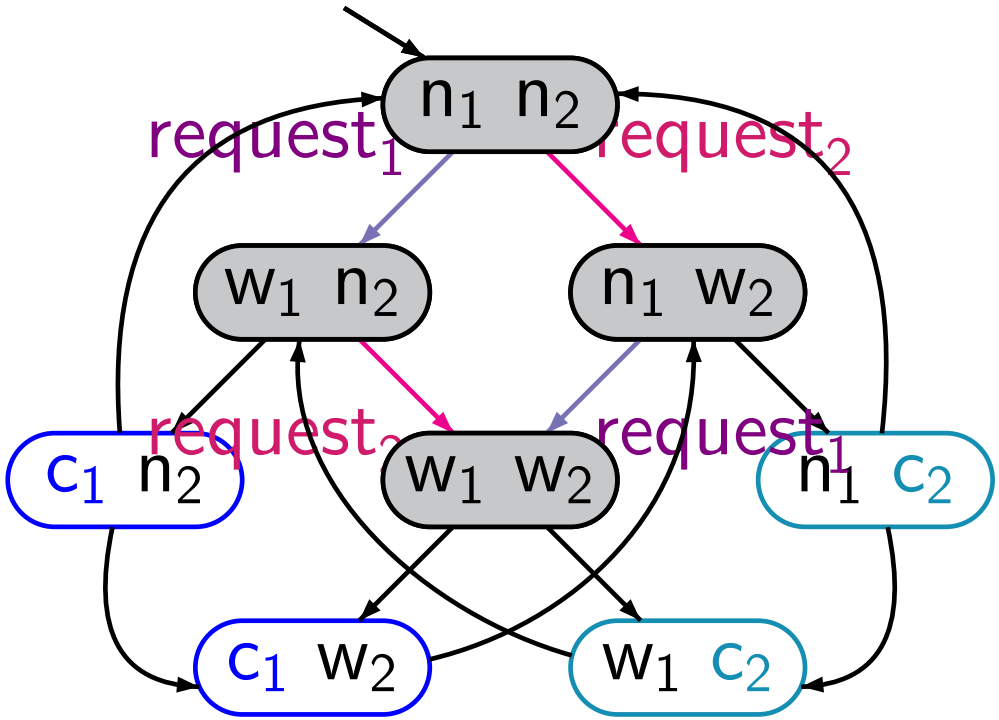
program graph P_1



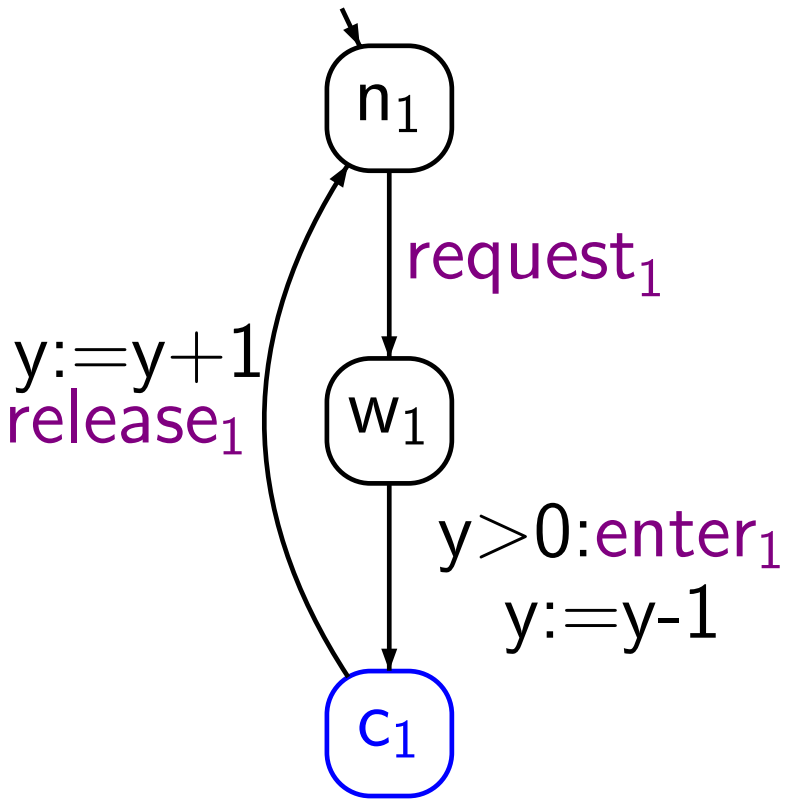
independent actions:

$request_1$, $request_2$

Mutual exclusion with semaphore



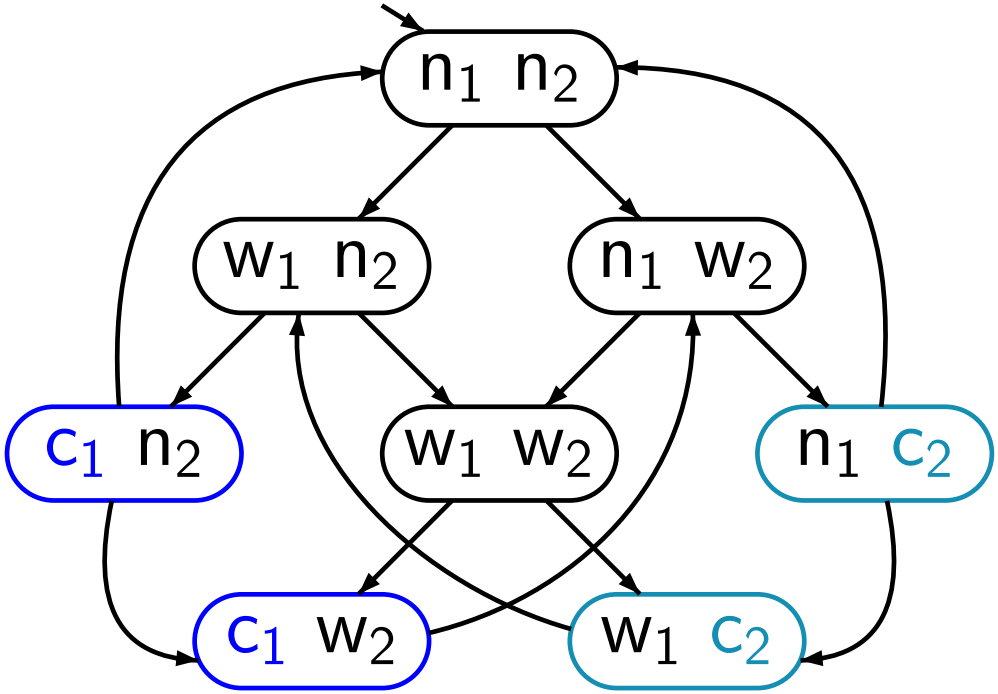
program graph P_1



independent actions:

$request_1$, $request_2$

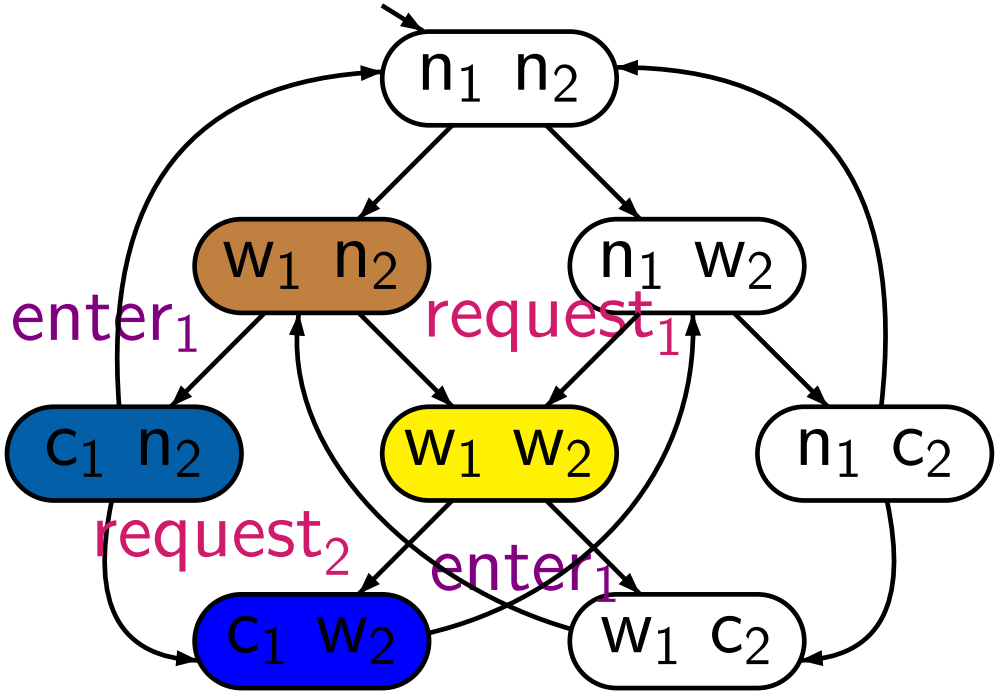
Example: independent actions for MUTEX



independent actions:

request_1, request_2

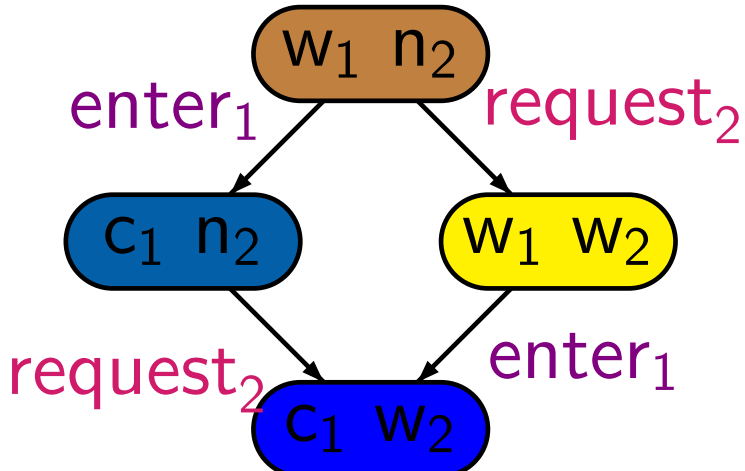
Example: independent actions for MUTEX



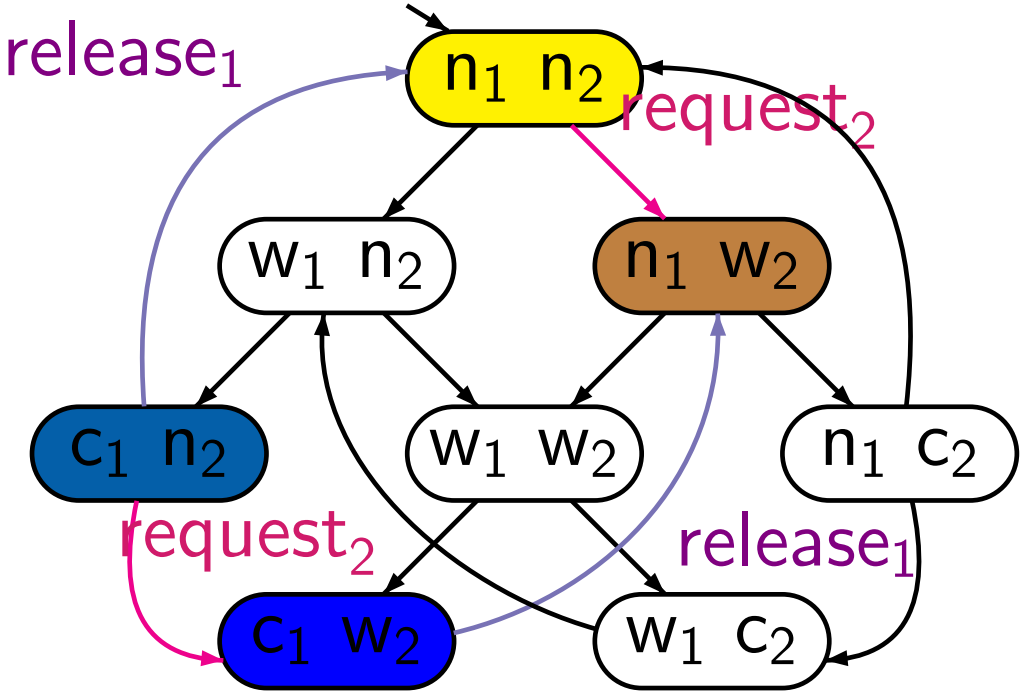
independent actions:

$request_1$, $request_2$

$enter_1$, $request_2$



Example: independent actions for MUTEX

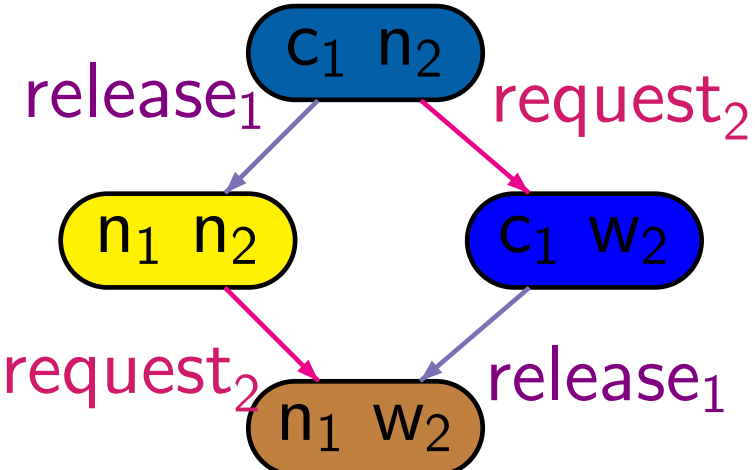


independent actions:

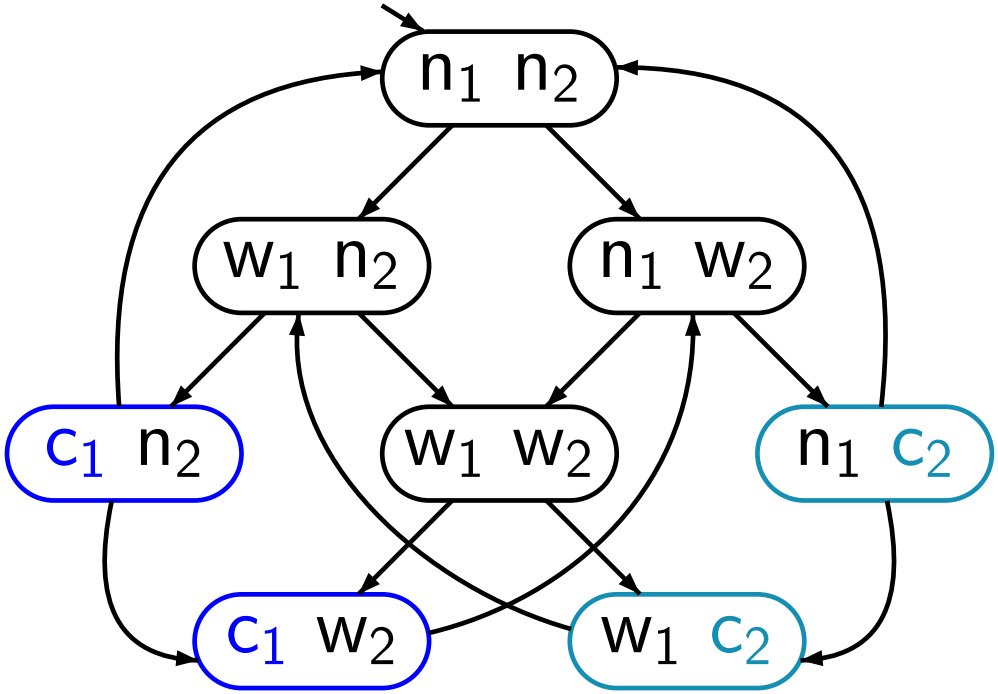
request$_1$, request$_2$

enter$_1$, request$_2$

release$_1$, request$_2$



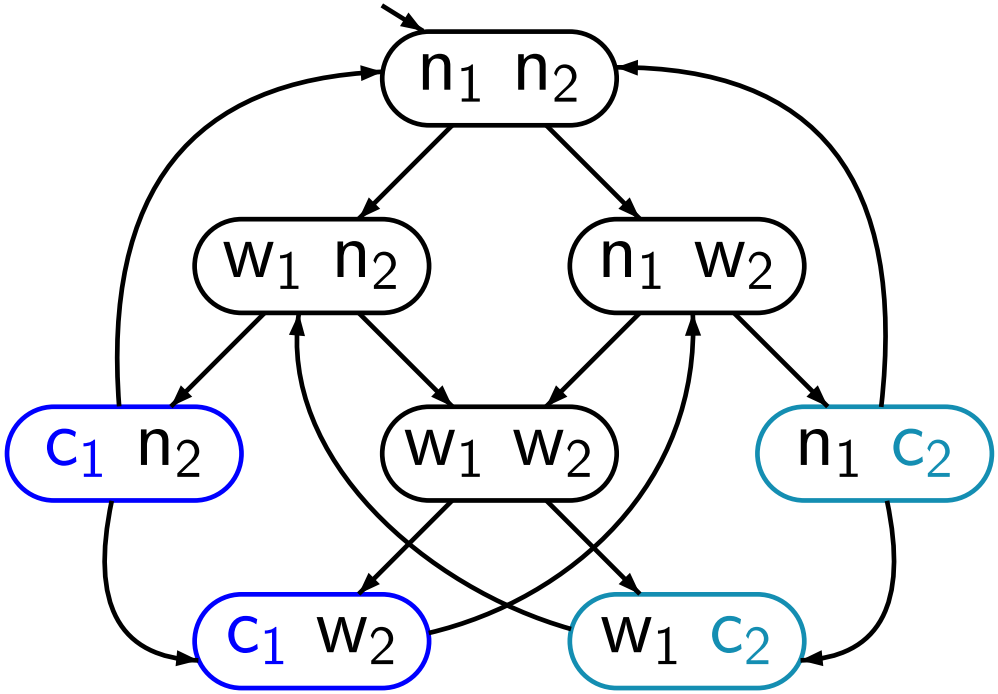
Example: independent actions for MUTEX



independent actions:

$request_1$, $request_2$
$enter_1$, $request_2$
$release_1$, $request_2$
$request_1$, $enter_2$
$request_1$, $release_2$

Example: independent actions for MUTEX

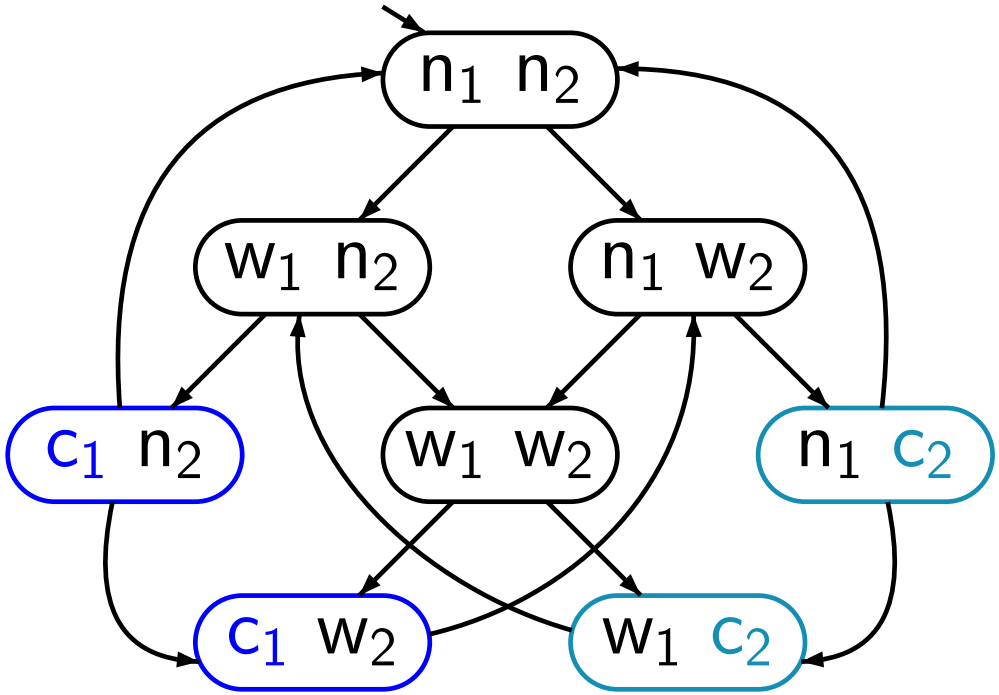


independent actions:

$request_1, request_2$
$enter_1, request_2$
$release_1, request_2$
$request_1, enter_2$
$request_1, release_2$

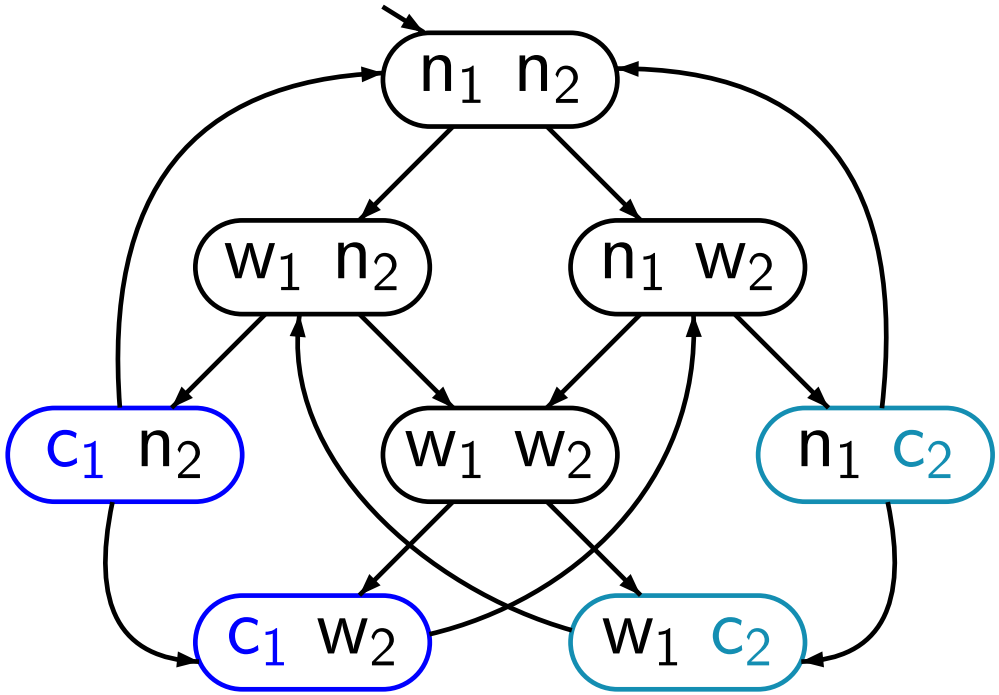
$request_1$ is independent from the action-set $\{request_2, enter_2, release_2\}$

Example: dependent actions for MUTEX



dependent actions:

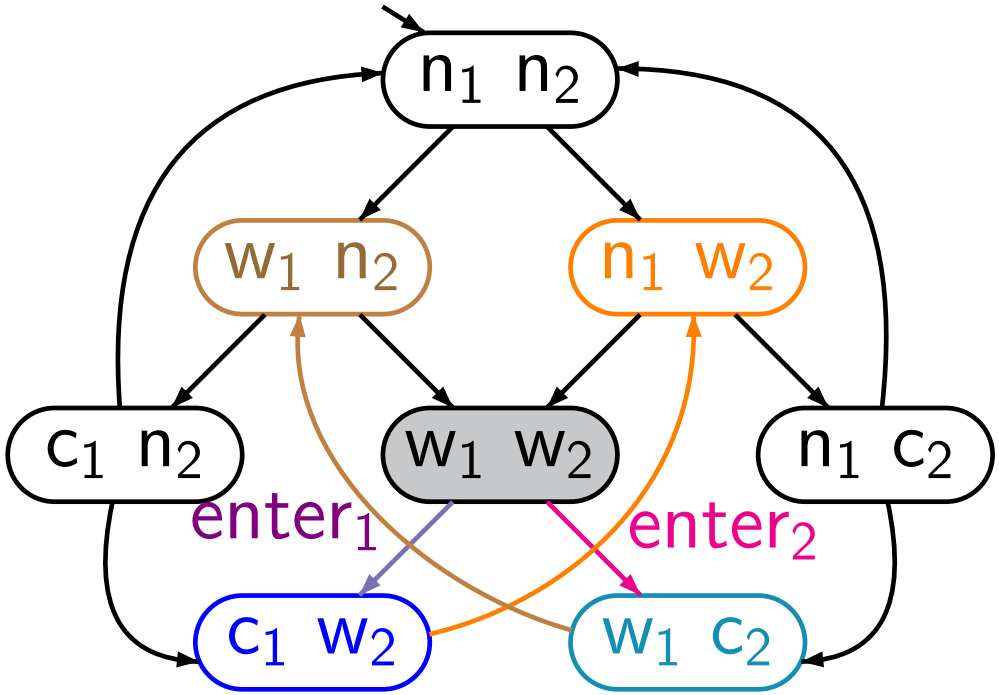
Example: dependent actions for MUTEX

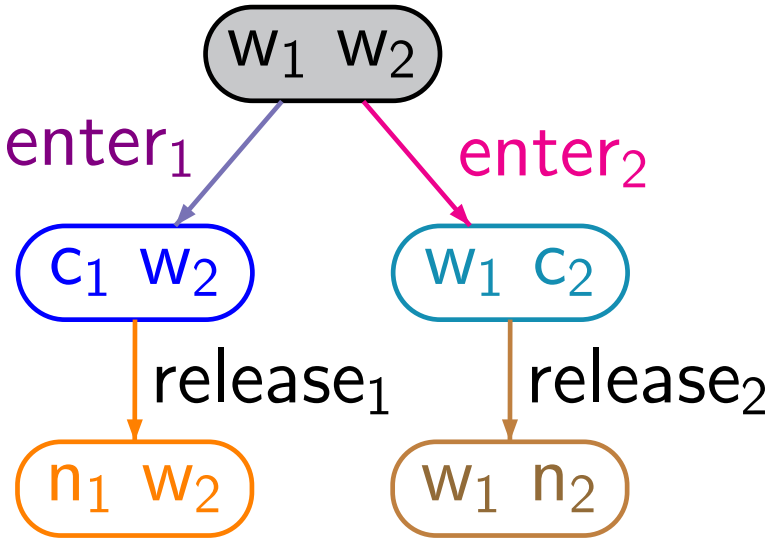


dependent actions:

$enter_1$, $enter_2$

Example: dependent actions for MUTEX





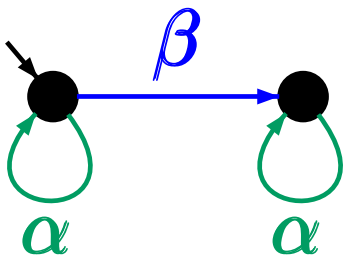
dependent actions:

$enter_1$, $enter_2$

access both to the semaphore

Correct or wrong?

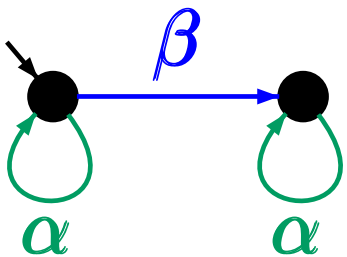
LTL3.4-15



α and β are independent **?**

Correct or wrong?

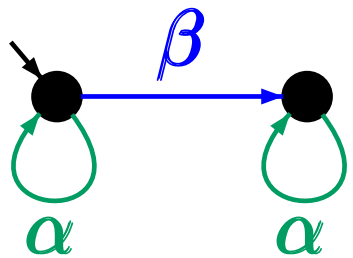
LTL3.4-15



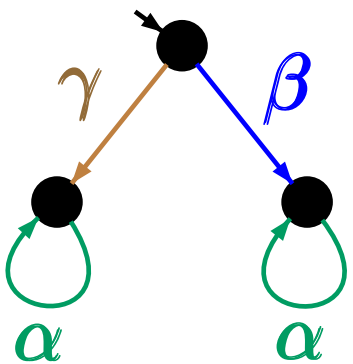
α and β are independent $\surd$

Correct or wrong?

LTL3.4-15



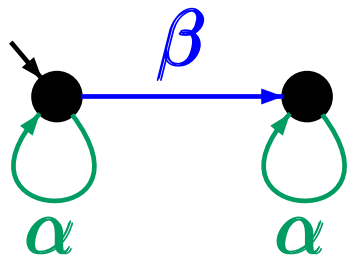
α and β are independent $\surd$



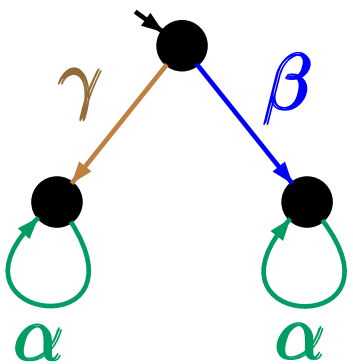
α and β are independent **?**

Correct or wrong?

LTL3.4-15



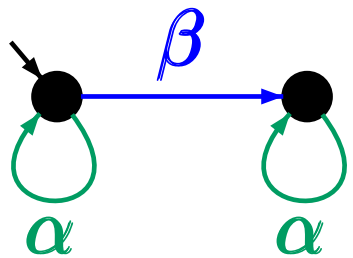
α and β are independent $\surd$



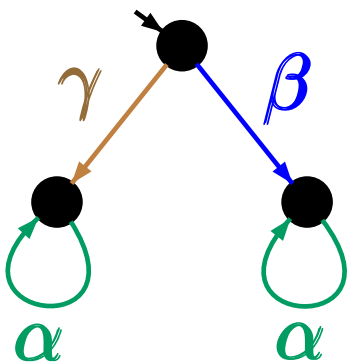
α and β are independent $\surd$

Correct or wrong?

LTL3.4-15



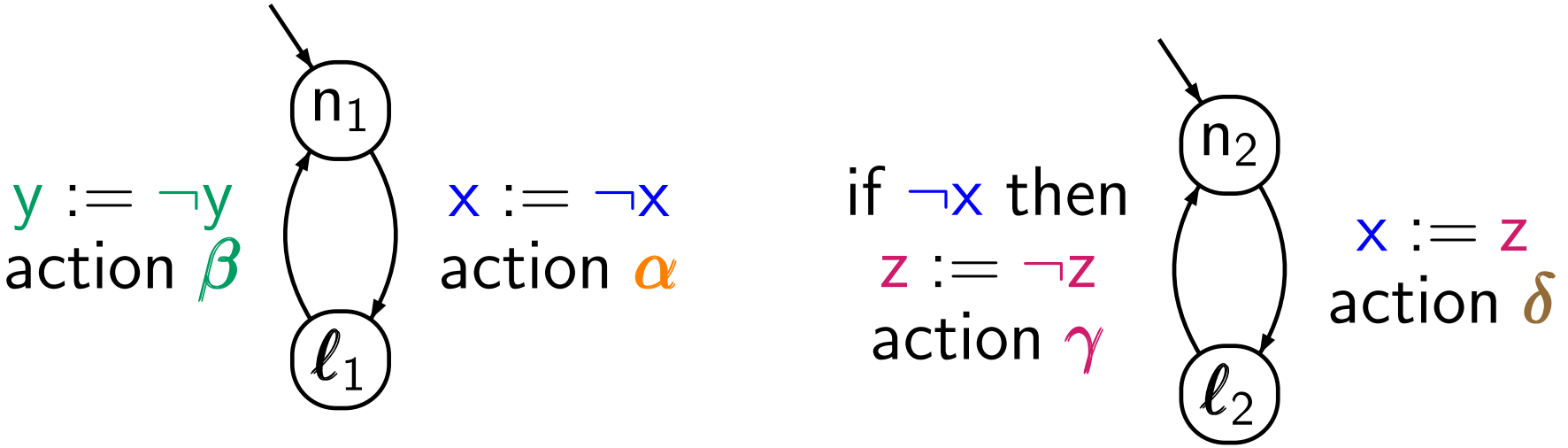
α and β are independent $\surd$



α and β are independent $\surd$

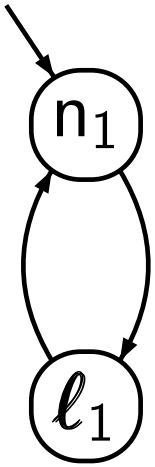
note: there is no state in which α and β are enabled

Independent or not?



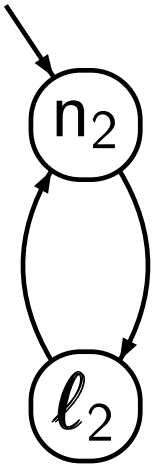
Independent or not?

y := ¬y
action β



x := ¬x
action α

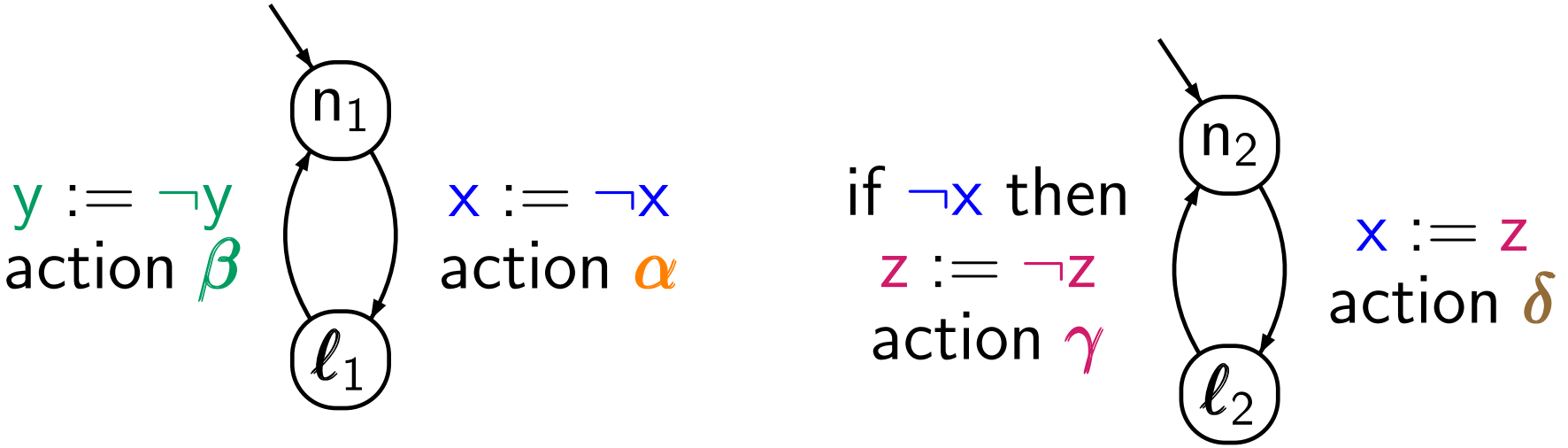
if ¬x then
z := ¬z
action γ



x := z
action δ

Are actions α, δ independent for $\mathcal{T}_{P_1 ||| P_2}$?

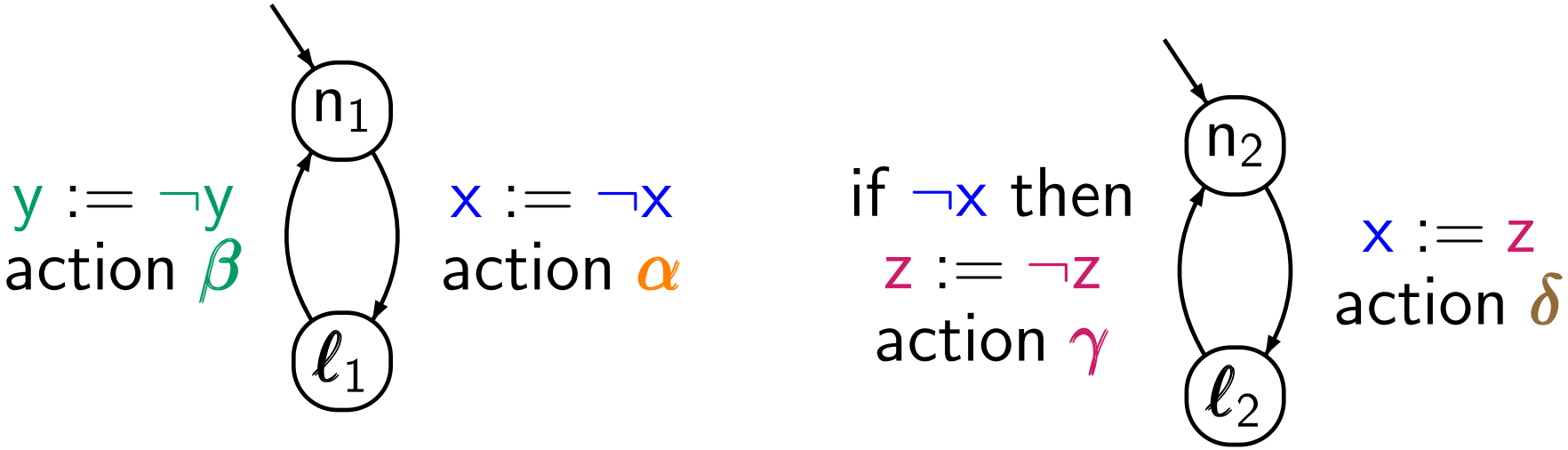
Independent or not?



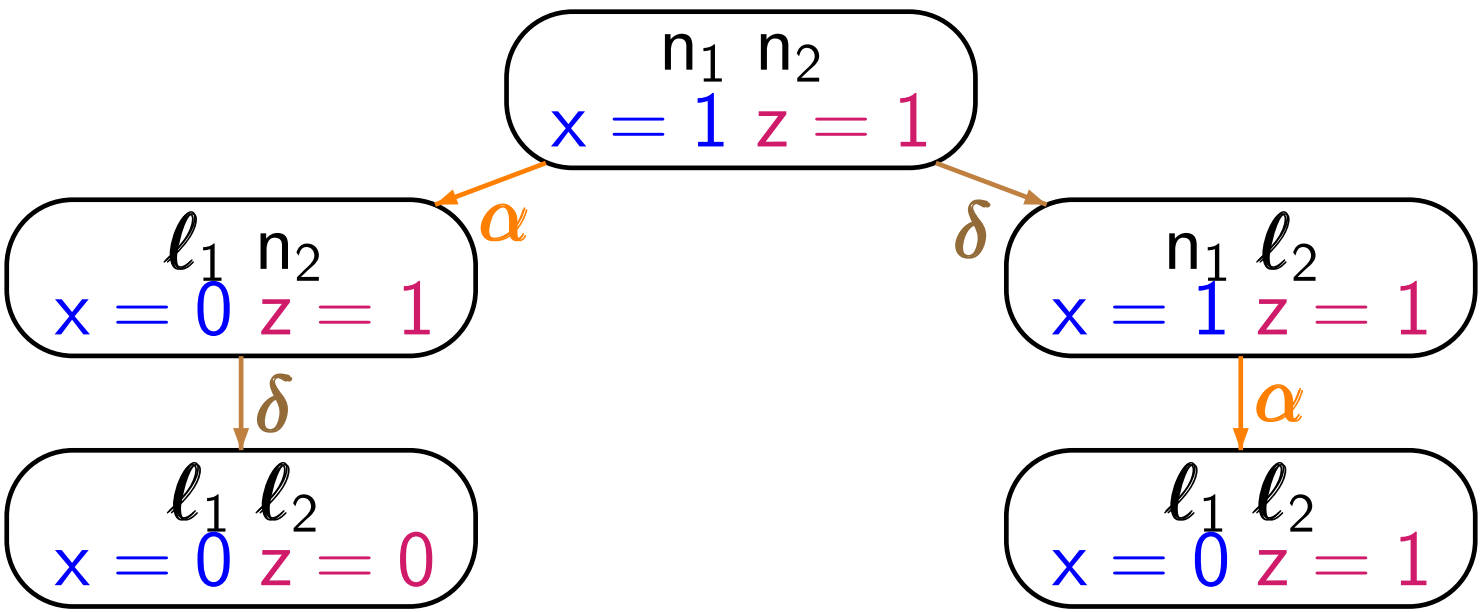
Are actions α, δ independent for $\mathcal{T}_{P_1 ||| P_2}$?

no

Independent or not?

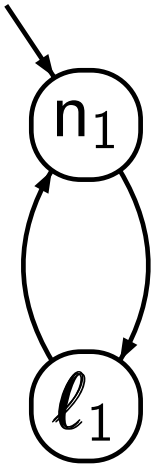


α, δ are dependent for $\mathcal{T}_{P_1 ||| P_2}$



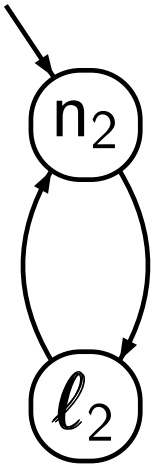
Independent or not?

y := ¬y
action β



x := ¬x
action α

if ¬x then
z := ¬z
action γ



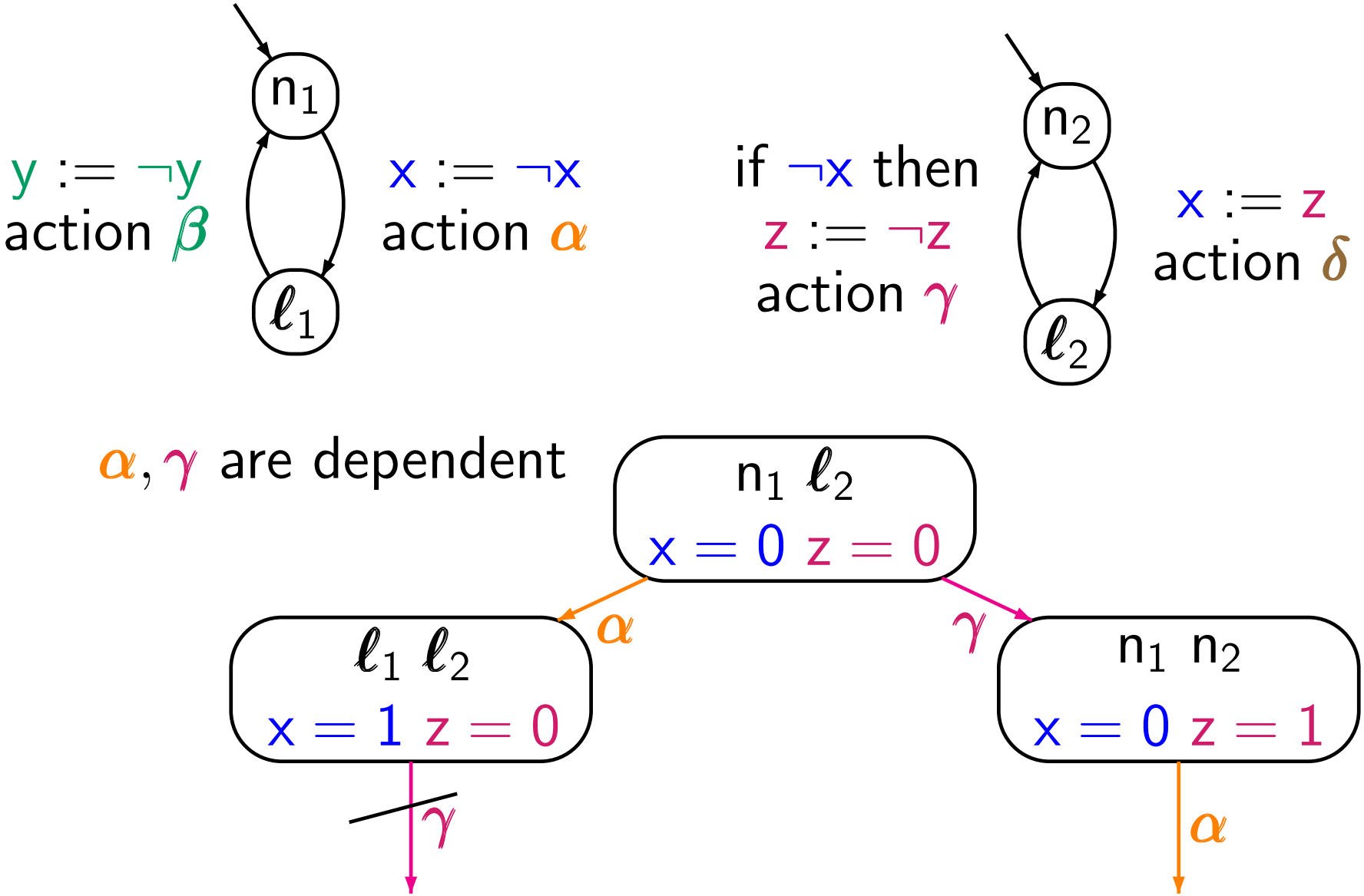
x := z
action δ

α, δ are dependent

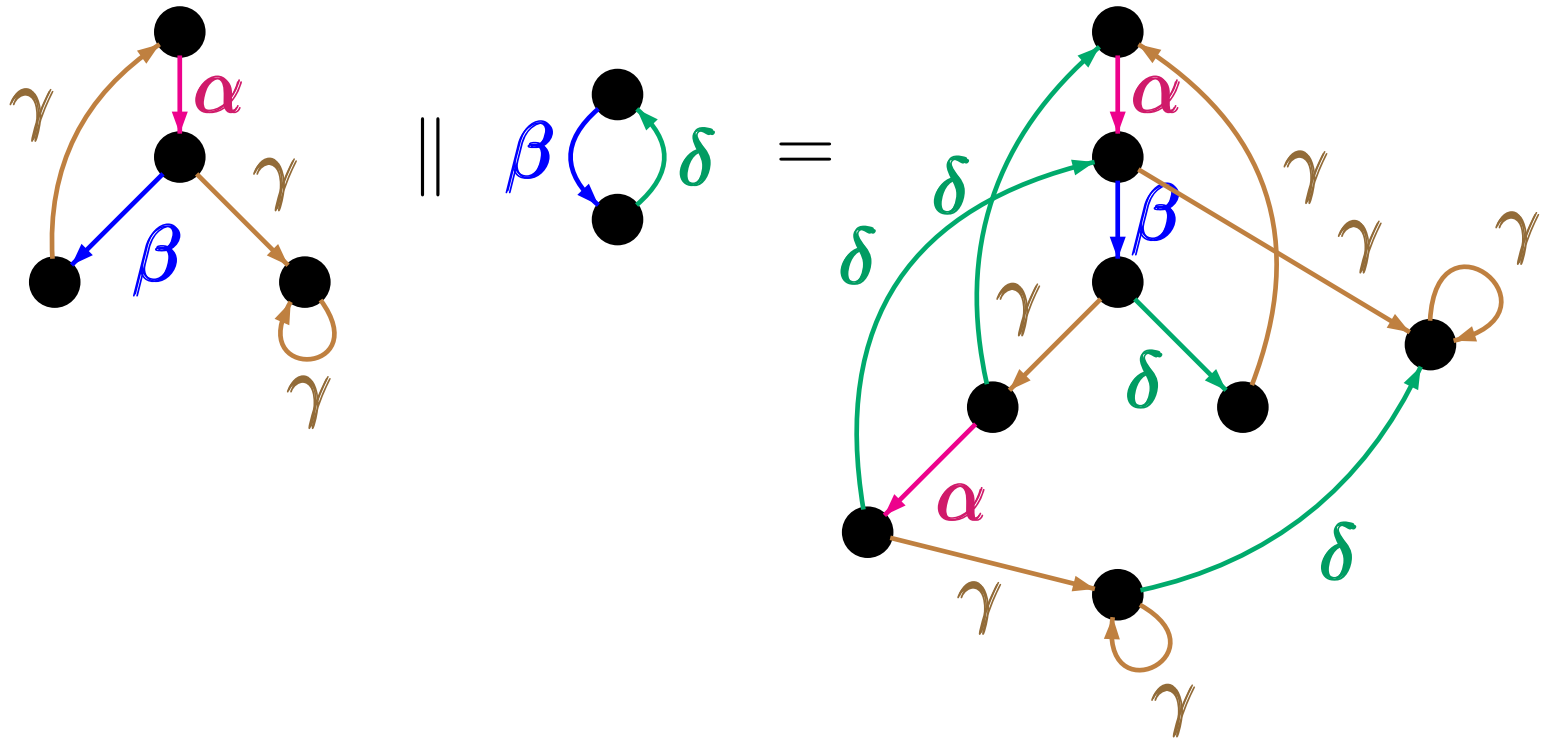
β, δ are independent,

as they access different variables

Independent or not?

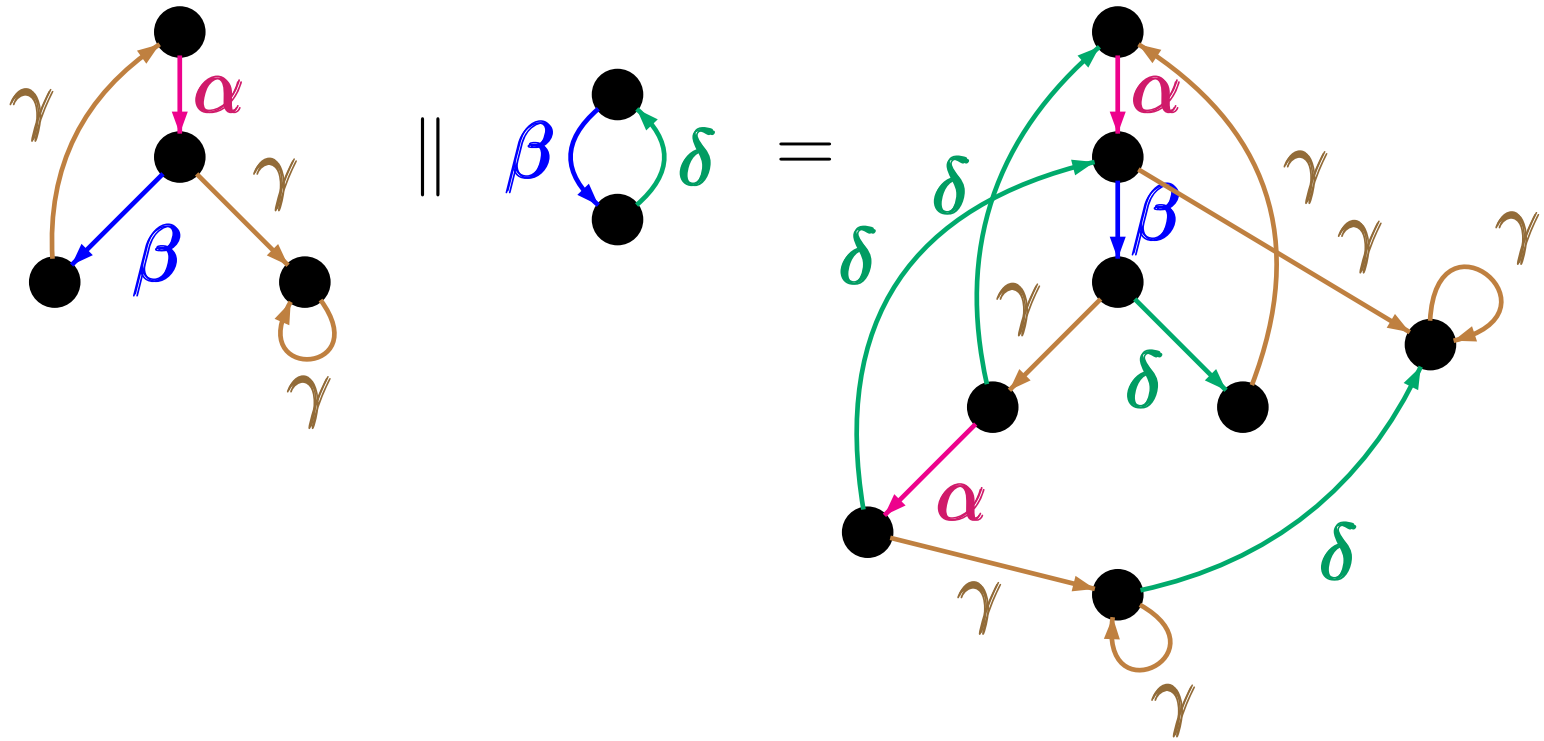


Independent or not?



TS that results by synchronization over the common action β

Independent or not?

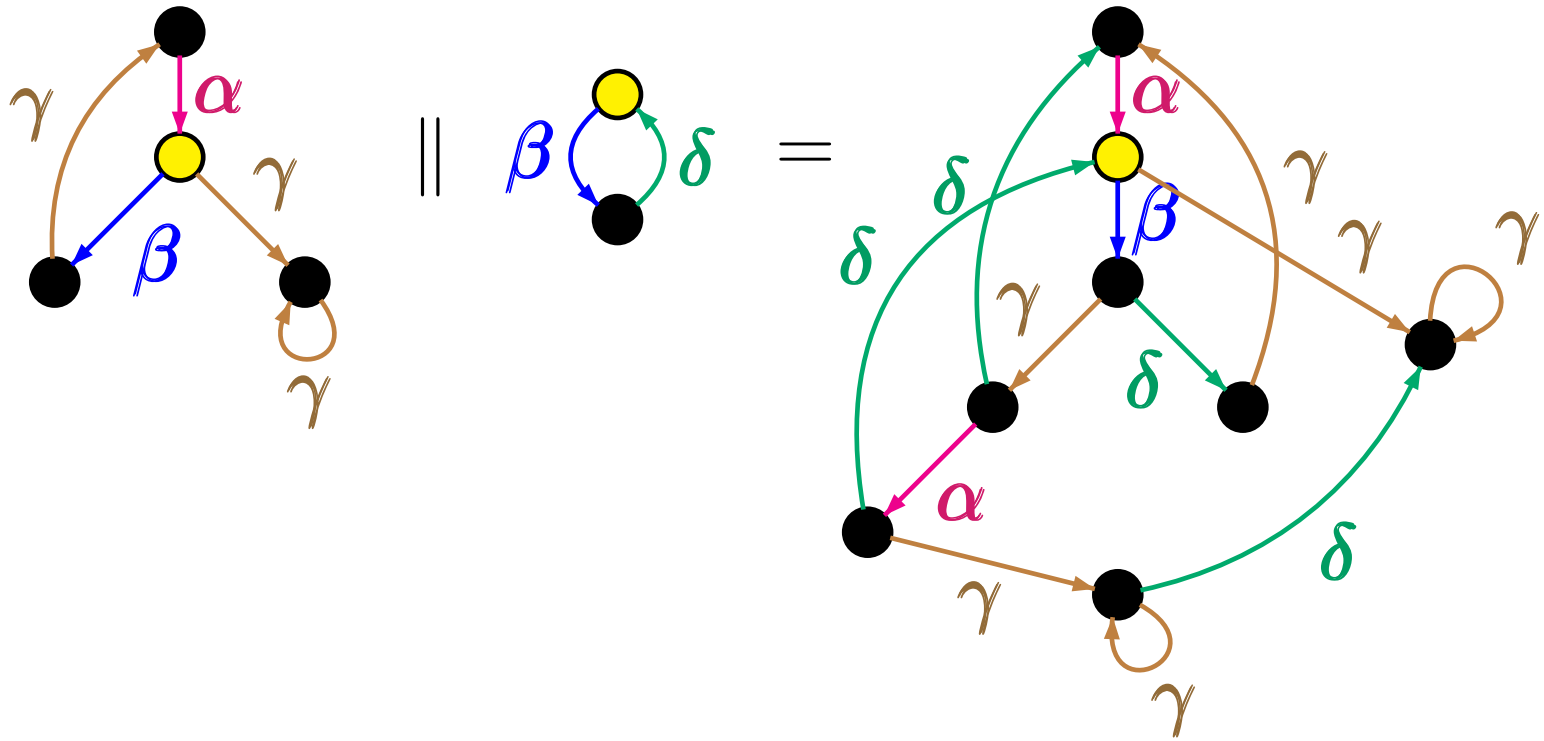


α, δ independent $\surd$

γ, δ independent $\surd$

β, γ dependent

Independent or not?



α, δ independent $\surd$

γ, δ independent $\surd$

β, γ dependent

Permutation of independent actions

Let α is independent from $\beta_1, \ldots, \beta_n$. I.e., for $1 \leq i \leq n$, the actions α and β_i are independent.

Then the action sequence

$$\beta_1 \beta_2 \ldots \beta_n \alpha$$

can be replaced with the action sequence

$$\alpha \beta_1 \beta_2 \ldots \beta_n$$

Permutation of independent actions

Let $\alpha \in Act(s_0)$ and

$$s_0 \xrightarrow{\beta_1} s_1 \xrightarrow{\beta_2} \ldots \xrightarrow{\beta_n} s_n \xrightarrow{\alpha} t$$

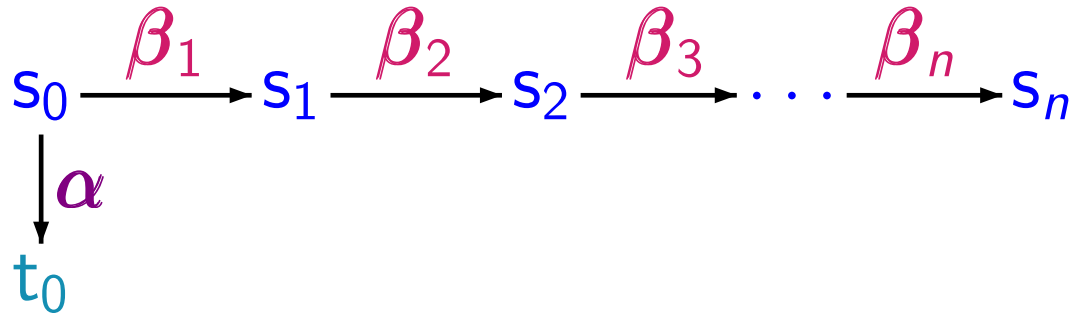
be a path fragment such that α is independent from $\beta_1, \ldots, \beta_n$.

Then there exists a path fragment

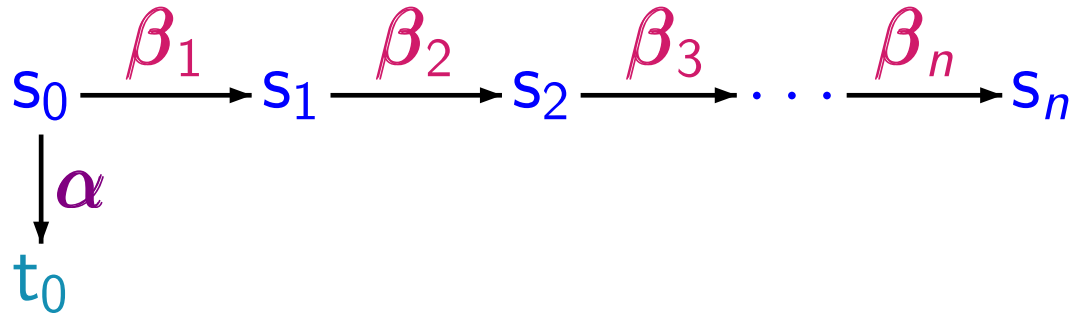
$$s_0 \xrightarrow{\alpha} t_0 \xrightarrow{\beta_1} t_1 \xrightarrow{\beta_2} \ldots \xrightarrow{\beta_n} t_n$$

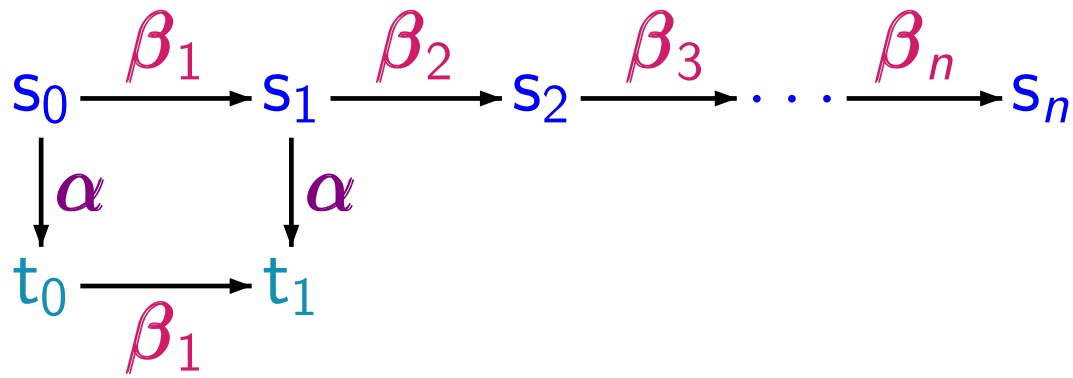
with $t_n = t$

α independent from $\beta_1, \ldots, \beta_n$

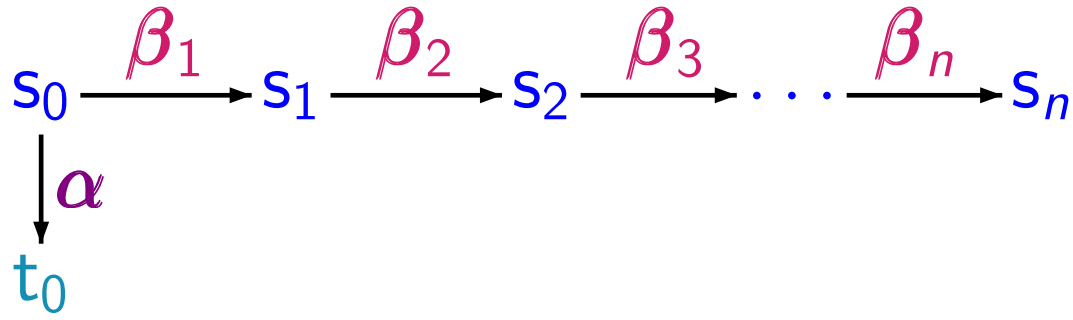


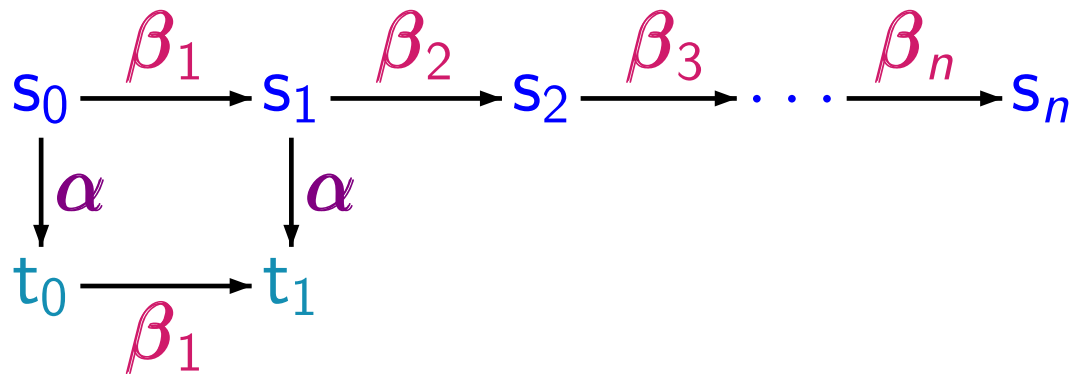
α independent from $\beta_1, \ldots, \beta_n$

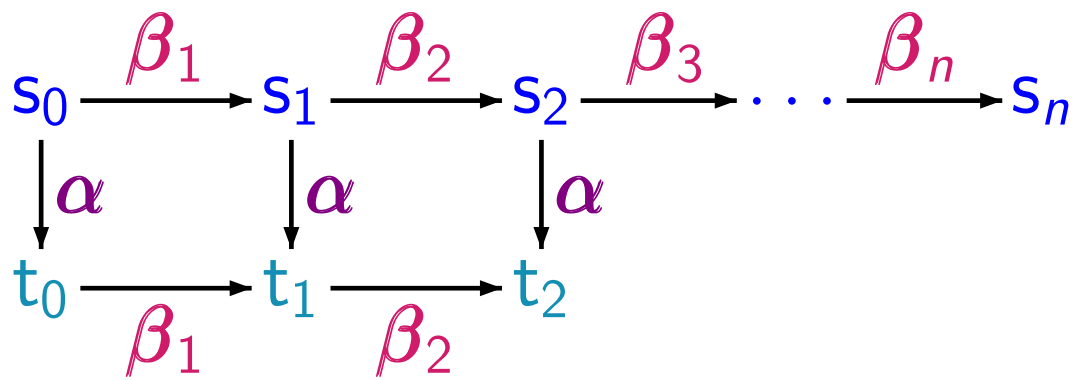




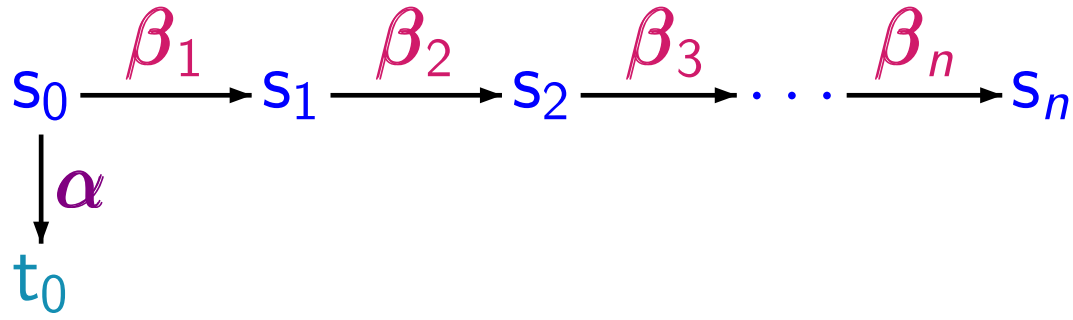
α independent from $\beta_1, \ldots, \beta_n$

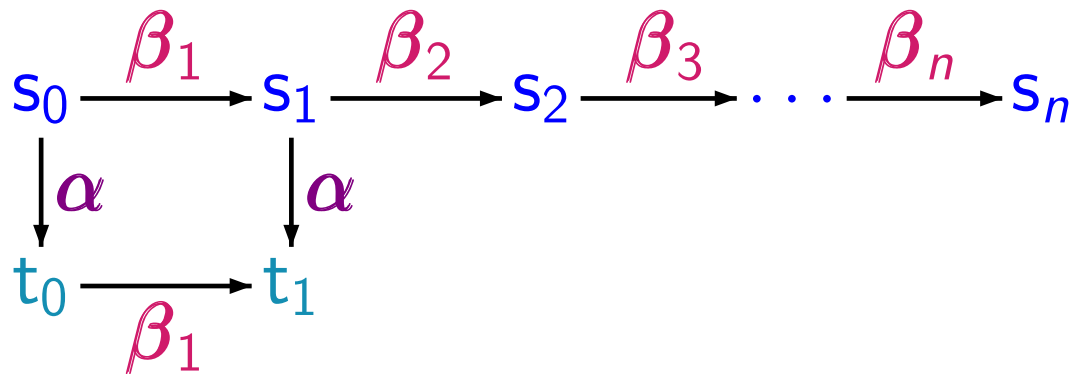


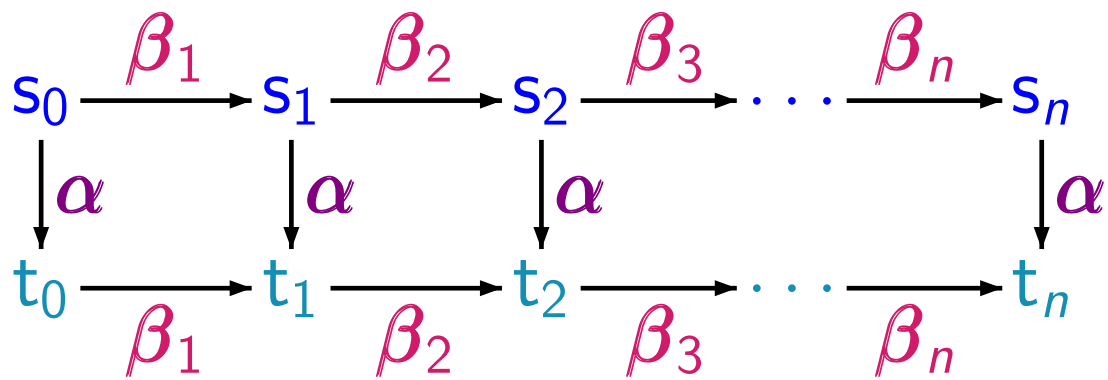




α independent from $\beta_1, \ldots, \beta_n$







The ample set method for LTL$_{\setminus\bigcirc}$

given: action-deterministic, finite transition system

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

goal: define "small" action-sets $ample(s) \subseteq Act(s)$ for all states $s \in S$ s.t.

$$\mathcal{T} \stackrel{\Delta}{=} \mathcal{T}_{red}$$

where $\mathcal{T}_{red} = (S', Act, \Longrightarrow, S_0, AP, L)$ results from $\mathcal{T}$ by an on-the-fly construction using the reduced transition relation

$$\frac{s \xrightarrow{\alpha} s' \ \wedge \ \alpha \in ample(s)}{s \stackrel{\alpha}{\Longrightarrow} s'}$$

$S' =$ reachable fragment w.r.t. $\Longrightarrow$

The ample set method for $\mathrm{LTL}_{\setminus\bigcirc}$

given: syntactic representation of the processes of an action-deterministic, finite transition system

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

goal: define "small" action-sets $ample(s) \subseteq Act(s)$ for all states $s \in S$ s.t. $\mathcal{T} \stackrel{\Delta}{=} \mathcal{T}_{red}$

Conditions for the ample sets

given: syntactic representation of the processes of an action-deterministic, finite transition system

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

goal: define "small" action-sets $ample(s) \subseteq Act(s)$ for all states $s \in S$ s.t. $\mathcal{T} \stackrel{\Delta}{=} \mathcal{T}_{red}$

provide *conditions* for the ample sets that

- are suitable for an on-the-fly construction of $\mathcal{T}_{red}$
- can be checked efficiently (without analyzing $\mathcal{T}$)
- ensure that $\mathcal{T} \stackrel{\Delta}{=} \mathcal{T}_{red}$

Stutter trace equivalence of $\mathcal{T}$ and $\mathcal{T}_{\text{red}}$

idea: the conditions for the ample sets

should ensure that

for each execution ρ in $\mathcal{T}$,

a stutter trace equivalent execution ρ_{red} in $\mathcal{T}_{\text{red}}$

can be constructed

Stutter trace equivalence of $\mathcal{T}$ and $\mathcal{T}_{\text{red}}$

idea: the conditions for the ample sets

should ensure that

for each execution ρ in $\mathcal{T}$,

a stutter trace equivalent execution ρ_{red} in $\mathcal{T}_{\text{red}}$

can be constructed by successively

- permutating the order independent actions
- adding independent stutter actions

Stutter trace equivalence of $\mathcal{T}$ and $\mathcal{T}_{\text{red}}$

idea: the conditions for the ample sets

should ensure that

for each execution ρ in $\mathcal{T}$,

a stutter trace equivalent execution ρ_{red} in $\mathcal{T}_{\text{red}}$

can be constructed by successively

- permutating the order independent actions
- adding independent stutter actions

execution ρ in $\mathcal{T}$ $\quad\rightsquigarrow\quad$ execution ρ_{red} in $\mathcal{T}_{\text{red}}$ s.t. $\rho \overset{\Delta}{=} \rho_{\text{red}}$

Stutter trace equivalence of $\mathcal{T}$ and $\mathcal{T}_{\text{red}}$

$$\text{execution } \rho \text{ in } \mathcal{T} \quad \rightsquigarrow \quad \begin{array}{c}\text{execution } \rho_{\text{red}} \text{ in } \mathcal{T}_{\text{red}} \\ \text{s.t. } \rho \overset{\Delta}{=} \rho_{\text{red}}\end{array}$$

by successively applying the following transformations:

Stutter trace equivalence of $\mathcal{T}$ and $\mathcal{T}_{\text{red}}$

execution ρ in $\mathcal{T}$ $\rightsquigarrow$ execution ρ_{red} in $\mathcal{T}_{\text{red}}$ s.t. $\rho \overset{\Delta}{=} \rho_{\text{red}}$

case 0: $\rho = s_0 \xrightarrow{\alpha} s_0' \longrightarrow \ldots$ with $\alpha \in \text{ample}(s_0)$

case 1: $\rho = s_0 \xrightarrow{\beta_1} \ldots \xrightarrow{\beta_n} \xrightarrow{\alpha} \longrightarrow \ldots$ with $\alpha \in \text{ample}(s_0)$, $\beta_i \notin \text{ample}(s_0)$

case 2: $\rho = s_0 \xrightarrow{\beta_1} \xrightarrow{\beta_2} \xrightarrow{\beta_3} \ldots$ with $\beta_i \notin \text{ample}(s_0)$

Stutter trace equivalence of $\mathcal{T}$ and $\mathcal{T}_{\text{red}}$

execution ρ in $\mathcal{T}$ $\rightsquigarrow$ execution ρ_{red} in $\mathcal{T}_{\text{red}}$ s.t. $\rho \overset{\Delta}{=} \rho_{\text{red}}$

case 0: $\rho = s_0 \xrightarrow{\alpha} s_0' \longrightarrow \ldots$ with $\alpha \in \text{ample}(s_0)$

$s_0 \overset{\alpha}{\Longrightarrow} s_0' \longrightarrow \ldots$

case 1: $\rho = s_0 \xrightarrow{\beta_1} \ldots \xrightarrow{\beta_n} \xrightarrow{\alpha} \longrightarrow \ldots$ with $\alpha \in \text{ample}(s_0)$ $\beta_i \notin \text{ample}(s_0)$

case 2: $\rho = s_0 \xrightarrow{\beta_1} \xrightarrow{\beta_2} \xrightarrow{\beta_3} \ldots$ with $\beta_i \notin \text{ample}(s_0)$

Stutter trace equivalence of $\mathcal{T}$ and $\mathcal{T}_{\text{red}}$

execution ρ in $\mathcal{T}$ $\rightsquigarrow$ execution ρ_{red} in $\mathcal{T}_{\text{red}}$ s.t. $\rho \overset{\Delta}{=} \rho_{\text{red}}$

case 0:	$\rho = s_0 \xrightarrow{\alpha} s_0' \longrightarrow \ldots$ with $\alpha \in \text{ample}(s_0)$ $s_0 \overset{\alpha}{\Longrightarrow} s_0' \longrightarrow \ldots$
case 1:	$\rho = s_0 \xrightarrow{\beta_1} \ldots \xrightarrow{\beta_n} \xrightarrow{\alpha} \longrightarrow \ldots$ with $\alpha \in \text{ample}(s_0)$, $\beta_i \notin \text{ample}(s_0)$ $s_0 \overset{\alpha}{\Longrightarrow} \xrightarrow{\beta_1} \ldots \xrightarrow{\beta_n} \longrightarrow \ldots$
case 2:	$\rho = s_0 \xrightarrow{\beta_1} \xrightarrow{\beta_2} \xrightarrow{\beta_3} \ldots$ with $\beta_i \notin \text{ample}(s_0)$

Stutter trace equivalence of $\mathcal{T}$ and $\mathcal{T}_{\text{red}}$

execution ρ in $\mathcal{T}$ $\rightsquigarrow$ execution ρ_{red} in $\mathcal{T}_{\text{red}}$ s.t. $\rho \overset{\Delta}{=} \rho_{\text{red}}$

case 0: $\rho =$	$s_0 \xrightarrow{\alpha} s_0' \longrightarrow \ldots$ with $\alpha \in \text{ample}(s_0)$ $s_0 \overset{\alpha}{\Longrightarrow} s_0' \longrightarrow \ldots$
case 1: $\rho =$	$s_0 \xrightarrow{\beta_1} \ldots \xrightarrow{\beta_n} \xrightarrow{\alpha} \longrightarrow \ldots$ with $\alpha \in \text{ample}(s_0)$, $\beta_i \notin \text{ample}(s_0)$ $s_0 \overset{\alpha}{\Longrightarrow} \xrightarrow{\beta_1} \ldots \xrightarrow{\beta_n} \longrightarrow \ldots$
case 2: $\rho =$	$s_0 \xrightarrow{\beta_1} \xrightarrow{\beta_2} \xrightarrow{\beta_3} \ldots$ with $\beta_i \notin \text{ample}(s_0)$ $s_0 \overset{\alpha}{\Longrightarrow} \xrightarrow{\beta_1} \xrightarrow{\beta_2} \xrightarrow{\beta_3} \ldots$ for some $\alpha \in \text{ample}(s_0)$

Stutter trace equivalence of $\mathcal{T}$ and $\mathcal{T}_{\text{red}}$

$$\text{execution } \rho \text{ in } \mathcal{T} \quad \rightsquigarrow \quad \begin{array}{c}\text{execution } \rho_{\text{red}} \text{ in } \mathcal{T}_{\text{red}} \\ \text{s.t. } \rho \overset{\Delta}{=} \rho_{\text{red}}\end{array}$$

ρ_{red} results by an infinite sequence application of *cases 0*, *1* and *2*, i.e.,

$$\rho \rightsquigarrow \rho_1 \rightsquigarrow \rho_2 \rightsquigarrow \rho_3 \rightsquigarrow \ldots$$

Stutter trace equivalence of $\mathcal{T}$ and $\mathcal{T}_{\text{red}}$

$$\text{execution } \rho \text{ in } \mathcal{T} \quad \rightsquigarrow \quad \begin{array}{c}\text{execution } \rho_{\text{red}} \text{ in } \mathcal{T}_{\text{red}} \\ \text{s.t. } \rho \overset{\Delta}{=} \rho_{\text{red}}\end{array}$$

ρ_{red} results by an infinite sequence application of *cases 0*, *1* and *2*, i.e.,

$$\rho \rightsquigarrow \rho_1 \rightsquigarrow \rho_2 \rightsquigarrow \rho_3 \rightsquigarrow \ldots$$

where for $i < j$ the executions ρ_j and ρ_i have a common prefix of length i which is a path fragment in $\mathcal{T}_{\text{red}}$

Stutter trace equivalence of $\mathcal{T}$ and $\mathcal{T}_{\text{red}}$

$$\text{execution } \rho \text{ in } \mathcal{T} \quad \rightsquigarrow \quad \begin{array}{c}\text{execution } \rho_{\text{red}} \text{ in } \mathcal{T}_{\text{red}} \\ \text{s.t. } \rho \overset{\Delta}{=} \rho_{\text{red}}\end{array}$$

ρ_{red} results by an infinite sequence application of *cases 0*, *1* and *2*, i.e.,

$$\rho \rightsquigarrow \rho_1 \rightsquigarrow \rho_2 \rightsquigarrow \rho_3 \rightsquigarrow \ldots$$

where for $i < j$ the executions ρ_j and ρ_i have a common prefix of length i which is a path fragment in $\mathcal{T}_{\text{red}}$,i.e., ρ_i has the form

$$\rho_i = \underbrace{s_0 \Rightarrow s_1 \Rightarrow \ldots \Rightarrow s_i}_{\text{in } \mathcal{T}_{\text{red}}} \underbrace{\rightarrow s_{i+1} \rightarrow s_{i+2} \rightarrow \ldots}_{\text{in } \mathcal{T}}$$

Stutter trace equivalence of $\mathcal{T}$ and $\mathcal{T}_{\text{red}}$

$$\text{execution } \rho \text{ in } \mathcal{T} \quad \rightsquigarrow \quad \begin{array}{c}\text{execution } \rho_{\text{red}} \text{ in } \mathcal{T}_{\text{red}} \\ \text{s.t. } \rho \overset{\Delta}{=} \rho_{\text{red}}\end{array}$$

ρ_{red} results by an infinite sequence application of *cases 0*, *1* and *2*, i.e.,

$$\rho \rightsquigarrow \rho_1 \rightsquigarrow \rho_2 \rightsquigarrow \rho_3 \rightsquigarrow \ldots$$

where

$$\rho_i = s_0 \Rightarrow s_1 \Rightarrow \ldots \Rightarrow s_i \rightarrow s_{i+1} \rightarrow s_{i+2} \rightarrow s_{i+3} \rightarrow \ldots$$

$$\rho_{i+1} = s_0 \Rightarrow s_1 \Rightarrow \ldots \Rightarrow s_i \Rightarrow s_{i+1} \rightarrow s_{i+2} \rightarrow s_{i+3} \rightarrow \ldots$$

$$\rho_{i+2} = s_0 \Rightarrow s_1 \Rightarrow \ldots \Rightarrow s_i \Rightarrow s_{i+1} \Rightarrow s_{i+2} \rightarrow s_{i+3} \rightarrow \ldots$$

Transformation $\rho \rightsquigarrow \rho_1$

case 0: $\rho = \quad s_0 \xrightarrow{\alpha} s_0' \longrightarrow \ldots$ with $\alpha \in \text{ample}(s_0)$

---

case 1: $\rho = \quad s_0 \xrightarrow{\beta_1} \ldots \xrightarrow{\beta_n} \xrightarrow{\alpha} \longrightarrow \ldots$ with $\alpha \in \text{ample}(s_0)$

$\beta_i \notin \text{ample}(s_0)$

---

case 2: $\rho = \quad s_0 \xrightarrow{\beta_1} \xrightarrow{\beta_2} \xrightarrow{\beta_3} \ldots$ with $\beta_i \notin \text{ample}(s_0)$

Transformation $\rho \rightsquigarrow \rho_1$

case 0: $\rho = s_0 \xrightarrow{\alpha} s_0' \longrightarrow \ldots$ with $\alpha \in \text{ample}(s_0)$

$\rho_1 = s_0 \overset{\alpha}{\Longrightarrow} s_0' \longrightarrow \ldots$

---

case 1: $\rho = s_0 \xrightarrow{\beta_1} \ldots \xrightarrow{\beta_n} \xrightarrow{\alpha} \longrightarrow \ldots$ with $\alpha \in \text{ample}(s_0)$

$\beta_i \notin \text{ample}(s_0)$

---

case 2: $\rho = s_0 \xrightarrow{\beta_1} \xrightarrow{\beta_2} \xrightarrow{\beta_3} \ldots$ with $\beta_i \notin \text{ample}(s_0)$

Transformation $\rho \rightsquigarrow \rho_1$

case 0: $\rho = s_0 \xrightarrow{\alpha} s_0' \longrightarrow \ldots$ with $\alpha \in \mathrm{ample}(s_0)$

$\rho_1 = s_0 \overset{\alpha}{\Longrightarrow} s_0' \longrightarrow \ldots$

---

case 1: $\rho = s_0 \xrightarrow{\beta_1} \ldots \xrightarrow{\beta_n} \xrightarrow{\alpha} \longrightarrow \ldots$ with $\alpha \in \mathrm{ample}(s_0)$, $\beta_i \notin \mathrm{ample}(s_0)$

$\rho_1 = s_0 \overset{\alpha}{\Longrightarrow} s_0' \xrightarrow{\beta_1} \ldots \xrightarrow{\beta_n} \longrightarrow \ldots$

---

case 2: $\rho = s_0 \xrightarrow{\beta_1} \xrightarrow{\beta_2} \xrightarrow{\beta_3} \ldots$ with $\beta_i \notin \mathrm{ample}(s_0)$

Transformation $\rho \rightsquigarrow \rho_1$

case 0: $\rho = s_0 \xrightarrow{\alpha} s_0' \longrightarrow \ldots$ with $\alpha \in \mathrm{ample}(s_0)$

$\rho_1 = s_0 \overset{\alpha}{\Rightarrow} s_0' \longrightarrow \ldots$

---

case 1: $\rho = s_0 \xrightarrow{\beta_1} \ldots \xrightarrow{\beta_n} \xrightarrow{\alpha} \longrightarrow \ldots$ with $\alpha \in \mathrm{ample}(s_0)$, $\beta_i \notin \mathrm{ample}(s_0)$

$\rho_1 = s_0 \overset{\alpha}{\Rightarrow} s_0' \xrightarrow{\beta_1} \ldots \xrightarrow{\beta_n} \longrightarrow \ldots$

---

case 2: $\rho = s_0 \xrightarrow{\beta_1} \xrightarrow{\beta_2} \xrightarrow{\beta_3} \ldots$ with $\beta_i \notin \mathrm{ample}(s_0)$

$\rho_1 = s_0 \overset{\alpha}{\Rightarrow} s_0' \xrightarrow{\beta_1} \xrightarrow{\beta_2} \xrightarrow{\beta_3} \ldots$ for some $\alpha \in \mathrm{ample}(s_0)$

Transformation $\rho \rightsquigarrow \rho_1$

case 0: $\rho = s_0 \xrightarrow{\alpha} s_0' \longrightarrow \ldots$ with $\alpha \in \text{ample}(s_0)$

$\rho_1 = s_0 \overset{\alpha}{\Longrightarrow} s_0' \longrightarrow \ldots$

---

case 1: $\rho = s_0 \xrightarrow{\beta_1} \ldots \xrightarrow{\beta_n} \xrightarrow{\alpha} \longrightarrow \ldots$ with $\alpha \in \text{ample}(s_0)$, $\beta_i \notin \text{ample}(s_0)$

$\rho_1 = s_0 \overset{\alpha}{\Longrightarrow} s_0' \xrightarrow{\beta_1} \ldots \xrightarrow{\beta_n} \longrightarrow \ldots$

---

case 2: $\rho = s_0 \xrightarrow{\beta_1} \xrightarrow{\beta_2} \xrightarrow{\beta_3} \ldots$ with $\beta_i \notin \text{ample}(s_0)$

$\rho_1 = s_0 \overset{\alpha}{\Longrightarrow} s_0' \xrightarrow{\beta_1} \xrightarrow{\beta_2} \xrightarrow{\beta_3} \ldots$ for some $\alpha \in \text{ample}(s_0)$

for the transformation $\rho_1 \rightsquigarrow \rho_2$:

apply case 0,1 or 2 to the suffix starting in state s_0'

Conditions for ample sets

LTL3.4-A12

(A1) nonemptiness condition

$$\emptyset \neq \mathsf{ample}(s) \subseteq \mathit{Act}(s)$$

Conditions for ample sets

LTL3.4-A12

(A1) nonemptiness condition

$$\emptyset \neq \mathsf{ample}(s) \subseteq \mathit{Act}(s)$$

(A2) dependency condition

Conditions for ample sets

LTL3.4-A12

(A1) nonemptiness condition

$$\emptyset \neq \mathsf{ample(s)} \subseteq \mathit{Act}(\mathsf{s})$$

(A2) dependency condition

for each execution fragment in $\boldsymbol{\mathcal{T}}$

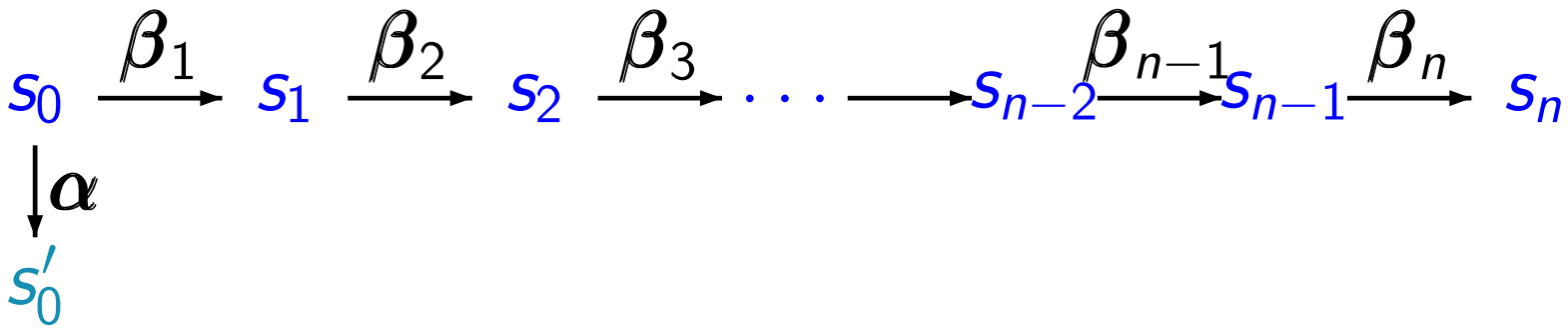
$$\mathsf{s} \xrightarrow{\beta_1} \xrightarrow{\beta_2} \ldots \xrightarrow{\beta_{i-1}} \xrightarrow{\beta_i} \xrightarrow{\beta_{i+1}} \ldots \xrightarrow{\beta_{n-1}} \xrightarrow{\beta_n}$$

such that β_n is dependent from ample(s)
there is some i $<$ n with

$$\beta_{\mathsf{i}} \in \mathsf{ample(s)}$$

Condition (A2)

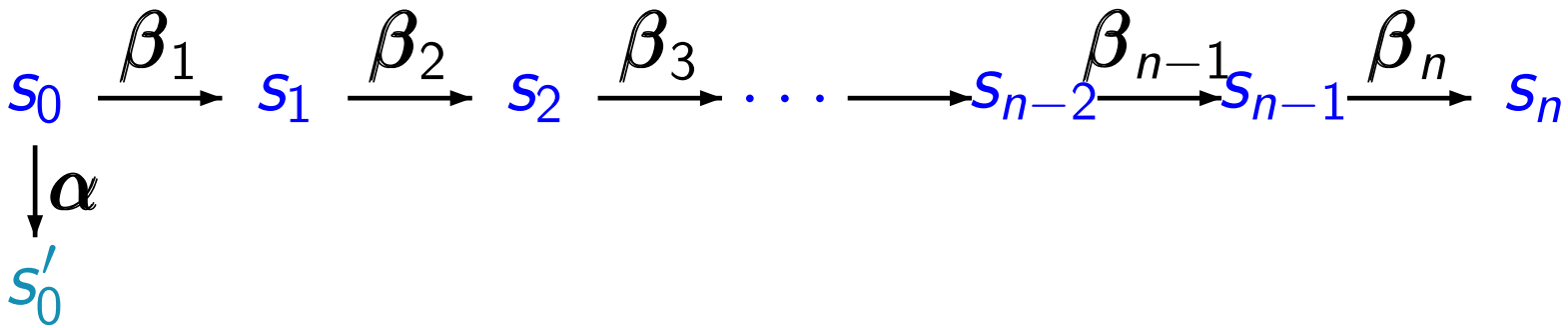
suppose $\alpha \in \text{ample}(s_0)$, $\beta_i \notin \text{ample}(s_0)$



Condition (A2)

suppose $\alpha \in \text{ample}(s_0)$, $\beta_i \notin \text{ample}(s_0)$

$\overset{\text{(A2)}}{\Longrightarrow}$ α, β_i independent

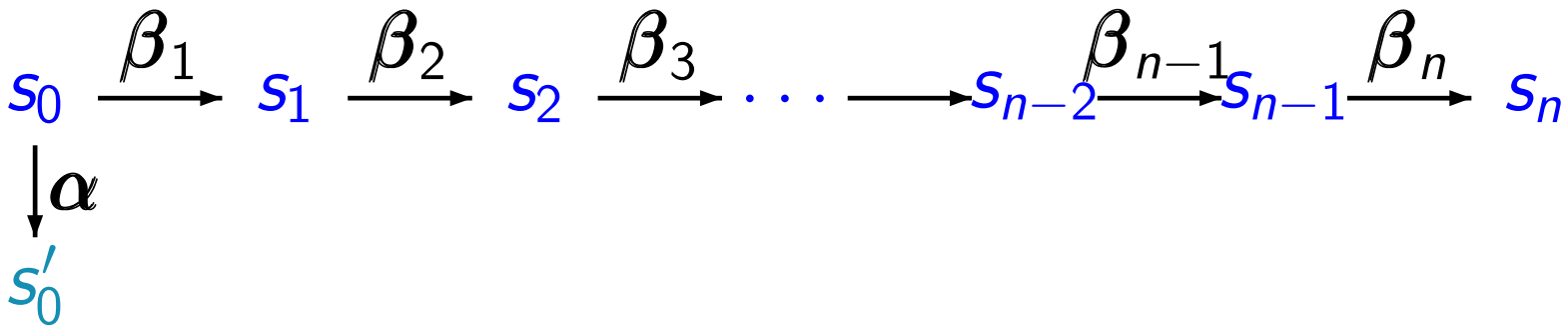


Condition (A2)

suppose $\alpha \in \text{ample}(s_0)$, $\beta_i \notin \text{ample}(s_0)$

$\stackrel{\text{(A2)}}{\Longrightarrow}$ α, β_i independent

$\Longrightarrow$ $\alpha \in Act(s_i)$ for $i = 0, 1, 2, \ldots$

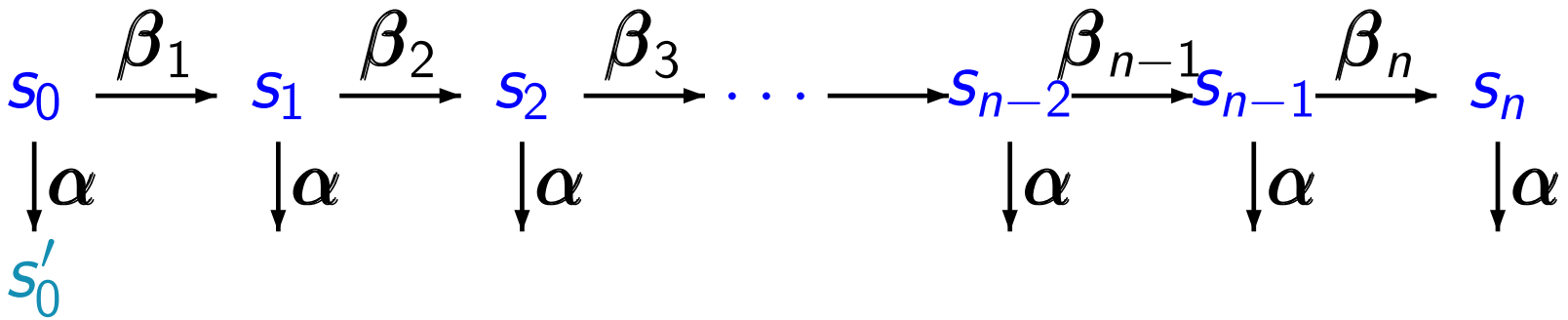


Condition (A2)

suppose $\alpha \in \text{ample}(s_0)$, $\beta_i \notin \text{ample}(s_0)$

$\stackrel{\text{(A2)}}{\Longrightarrow}$ α, β_i independent

$\Longrightarrow$ $\alpha \in Act(s_i)$ for $i = 0, 1, 2, \ldots$

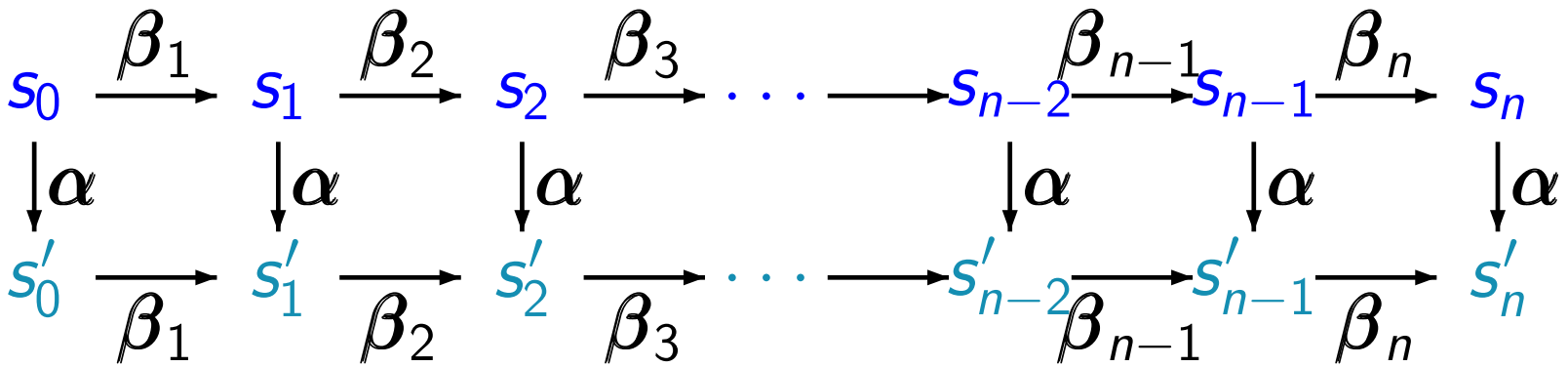


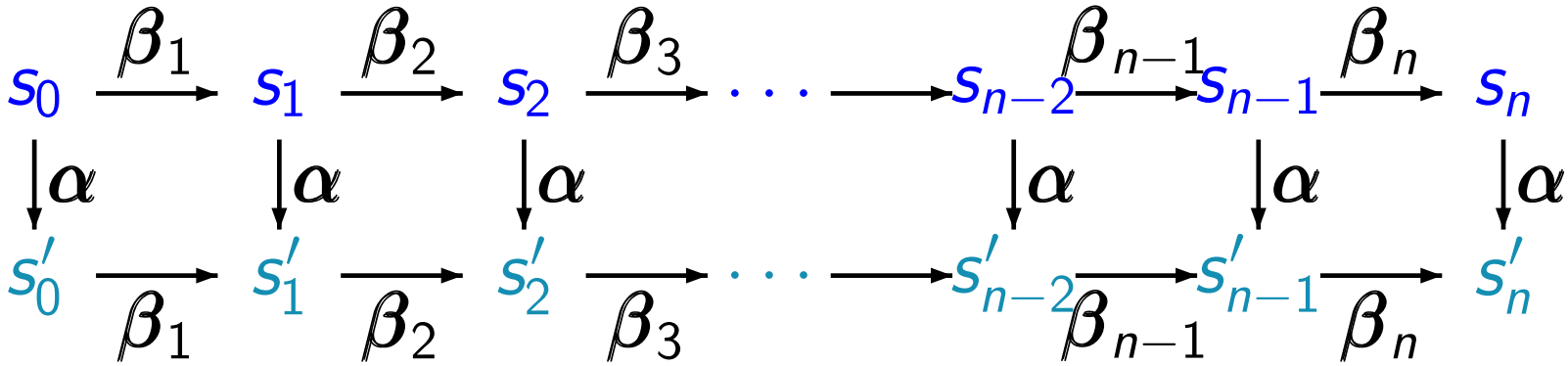
Condition (A2)

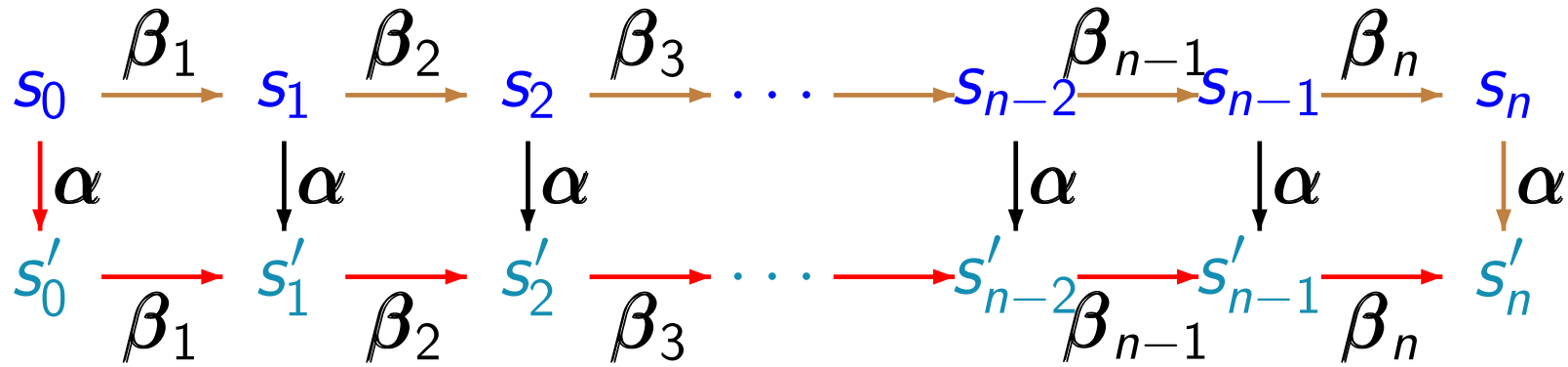
suppose $\alpha \in \text{ample}(s_0)$, $\beta_i \notin \text{ample}(s_0)$

$\overset{\text{(A2)}}{\Longrightarrow}$ α, β_i independent

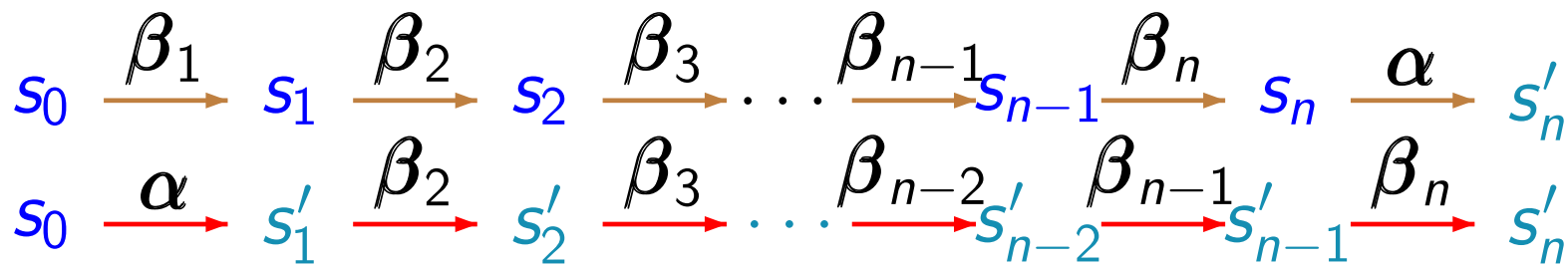
$\Longrightarrow$ $\alpha \in Act(s_i)$ for $i = 0, 1, 2, \ldots$

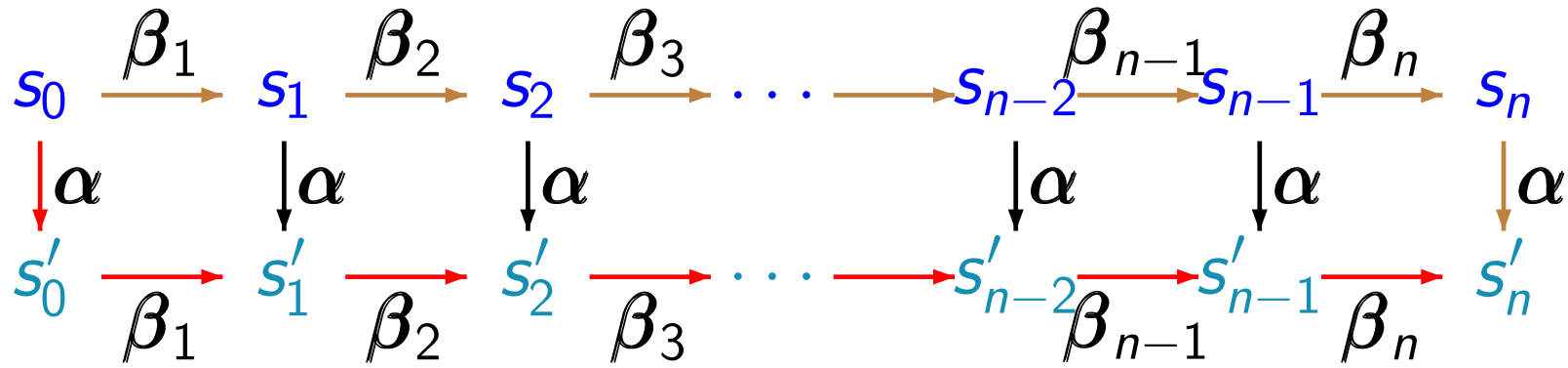



s_0 β_1 s_1 β_2 s_2 β_3 $\cdots$ s_{n-2} β_{n-1} s_{n-1} β_n s_n
α α α α α α
s'_0 β_1 s'_1 β_2 s'_2 β_3 $\cdots$ s'_{n-2} β_{n-1} s'_{n-1} β_n s'_n

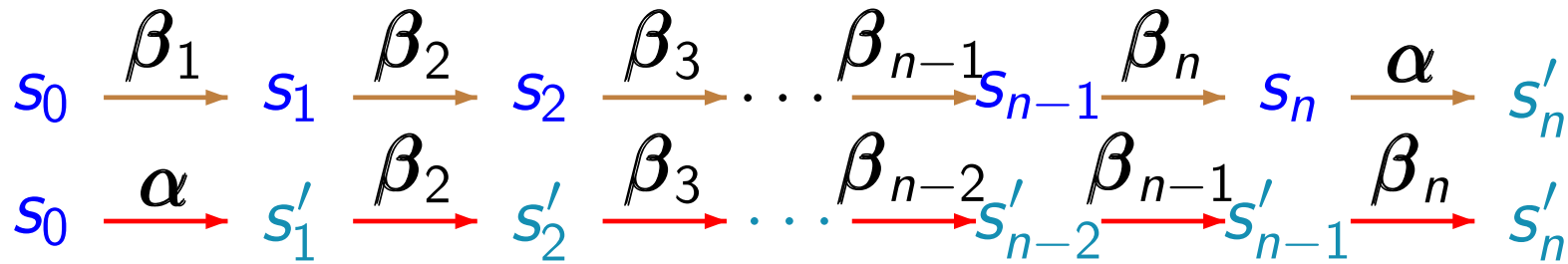

s_0 β_1 s_1 β_2 s_2 β_3 $\cdots$ s_{n-2} β_{n-1} s_{n-1} β_n s_n
α α α α α α
s'_0 β_1 s'_1 β_2 s'_2 β_3 $\cdots$ s'_{n-2} β_{n-1} s'_{n-1} β_n s'_n

case 1:

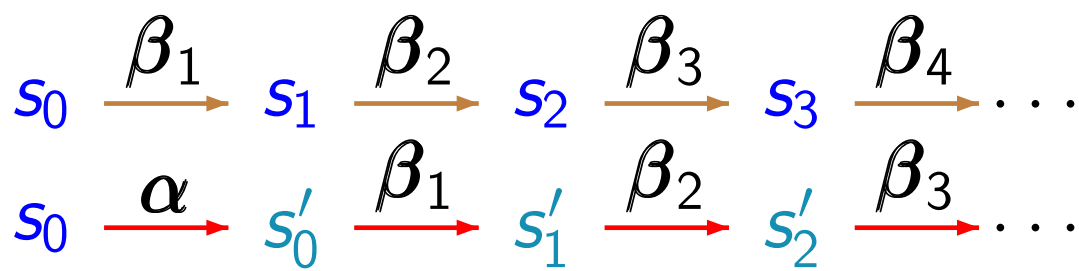

s_0 β_1 s_1 β_2 s_2 β_3 $\cdots$ β_{n-1} s_{n-1} β_n s_n α s'_n
s_0 α s'_1 β_2 s'_2 β_3 $\cdots$ β_{n-2} s'_{n-2} β_{n-1} s'_{n-1} β_n s'_n


s_0 β_1 s_1 β_2 s_2 β_3 $\cdots$ s_{n-2} β_{n-1} s_{n-1} β_n s_n
α α α α α α
s'_0 β_1 s'_1 β_2 s'_2 β_3 $\cdots$ s'_{n-2} β_{n-1} s'_{n-1} β_n s'_n

case 1:


s_0 β_1 s_1 β_2 s_2 β_3 $\cdots$ β_{n-1} s_{n-1} β_n s_n α s'_n
s_0 α s'_1 β_2 s'_2 β_3 $\cdots$ β_{n-2} s'_{n-2} β_{n-1} s'_{n-1} β_n s'_n

case 2:


s_0 β_1 s_1 β_2 s_2 β_3 s_3 β_4 $\cdots$
s_0 α s'_0 β_1 s'_1 β_2 s'_2 β_3 $\cdots$

Conditions for ample sets

(A1) nonemptiness condition

$\emptyset \neq \mathsf{ample(s)} \subseteq \mathit{Act}(\mathsf{s})$

(A2) dependency condition

for each execution fragment in $\boldsymbol{\mathcal{T}}$

$$\mathsf{s} \xrightarrow{\beta_1}\xrightarrow{\beta_2} \ldots \xrightarrow{\beta_{i-1}} \xrightarrow{\beta_i} \xrightarrow{\beta_{i+1}} \ldots \xrightarrow{\beta_{n-1}} \xrightarrow{\beta_n}$$

such that β_n is *dependent* from ample(s) there is some i $<$ n with $\beta_{\mathsf{i}} \in$ ample(s)

Conditions for ample sets

(A1) nonemptiness condition

$\emptyset \neq \mathsf{ample(s)} \subseteq \mathit{Act}(\mathsf{s})$

(A2) dependency condition

for each execution fragment in $\boldsymbol{\mathcal{T}}$

$$\mathsf{s} \xrightarrow{\beta_1}\xrightarrow{\beta_2} \ldots \xrightarrow{\beta_{i-1}} \xrightarrow{\beta_i} \xrightarrow{\beta_{i+1}} \ldots \xrightarrow{\beta_{n-1}} \xrightarrow{\beta_n}$$

such that β_n is *dependent* from ample(s) there is some i $<$ n with $\beta_{\mathsf{i}} \in$ ample(s)

(A3) stutter condition

Conditions for ample sets

(A1) nonemptiness condition

$\emptyset \neq \mathsf{ample(s)} \subseteq \mathit{Act}(\mathsf{s})$

(A2) dependency condition

for each execution fragment in $\boldsymbol{\mathcal{T}}$

$$\mathsf{s} \xrightarrow{\beta_1}\xrightarrow{\beta_2} \ldots \xrightarrow{\beta_{i-1}} \xrightarrow{\beta_i} \xrightarrow{\beta_{i+1}} \ldots \xrightarrow{\beta_{n-1}} \xrightarrow{\beta_n}$$

such that β_n is *dependent* from ample(s) there is some i $<$ n with $\beta_{\mathsf{i}} \in$ ample(s)

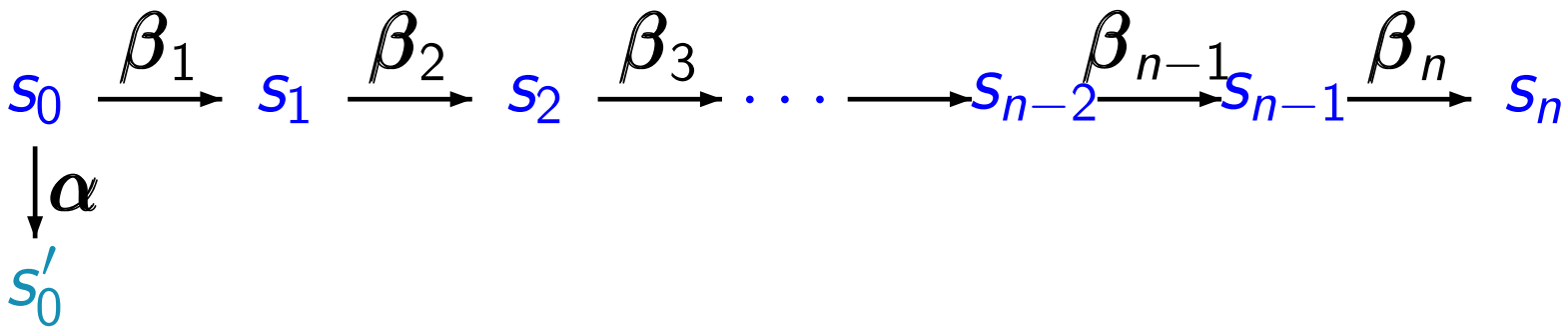
(A3) stutter condition

if ample(s) $\neq \mathit{Act}(\mathsf{s})$ then all actions in ample(s) are stutter actions

Conditions (A2) and (A3)

Suppose

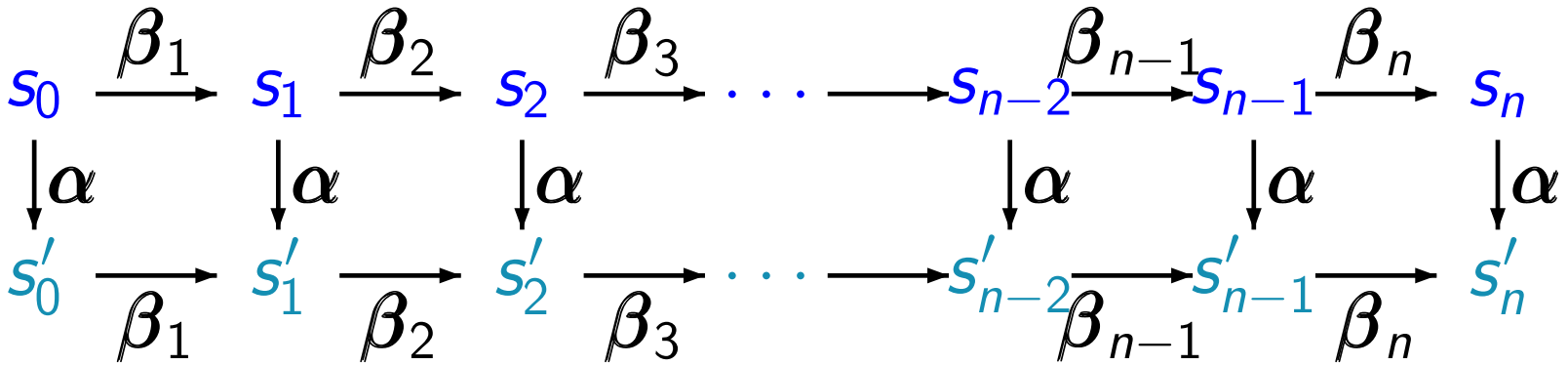
- $\alpha \in \text{ample}(s_0)$, $\beta_i \notin \text{ample}(s_0)$
- α stutter action



Conditions (A2) and (A3)

Suppose

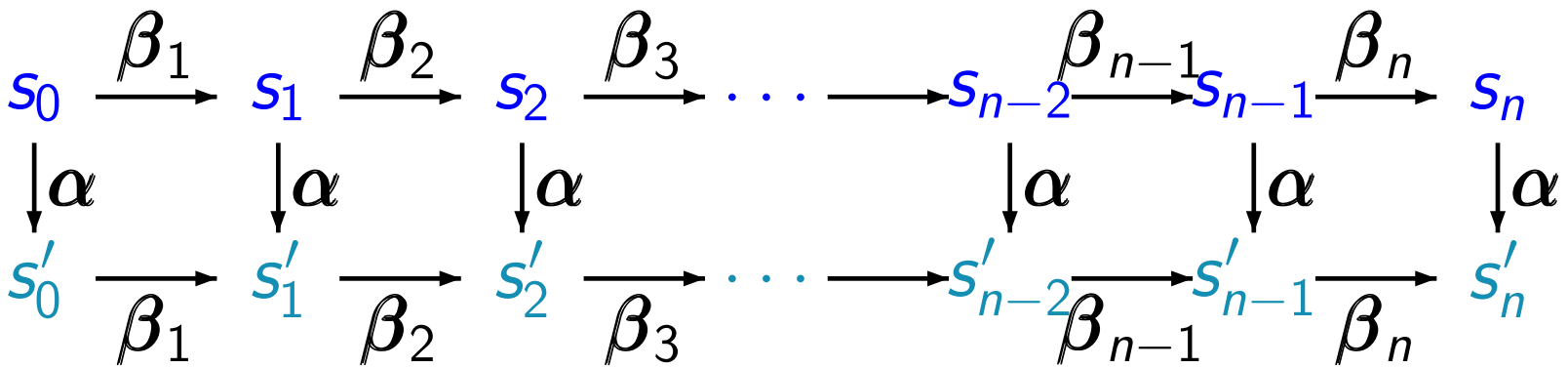
- $\alpha \in \text{ample}(s_0)$, $\beta_i \notin \text{ample}(s_0)$
- α stutter action



Conditions (A2) and (A3)

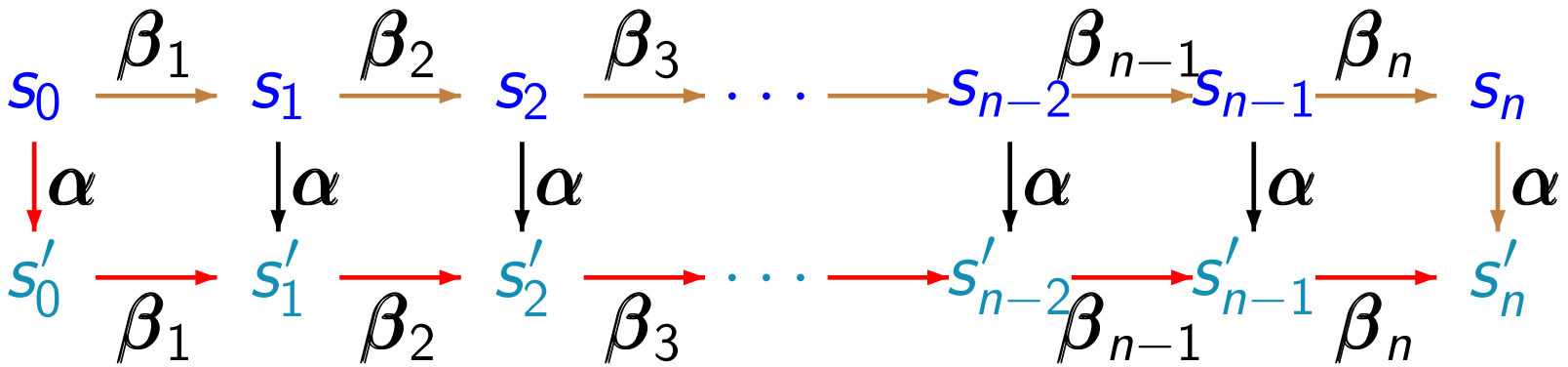
Suppose

- $\alpha \in \text{ample}(s_0)$, $\beta_i \notin \text{ample}(s_0)$
- α stutter action $\Rightarrow L(s_i) = L(s'_i)$, $i = 0, 1, 2, \ldots$

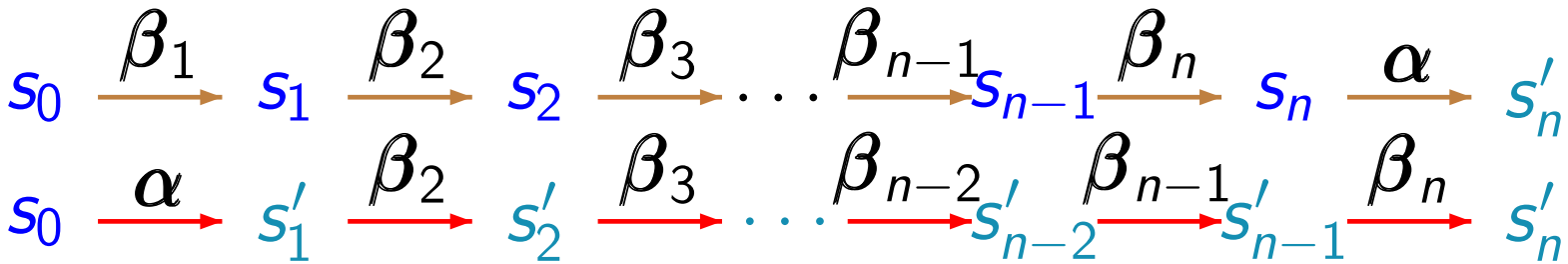


Conditions (A2) and (A3)

- $\alpha \in \text{ample}(s_0)$, $\beta_i \notin \text{ample}(s_0)$
- α stutter action $\Rightarrow L(s_i) = L(s'_i)$, $i = 0, 1, 2, \ldots$

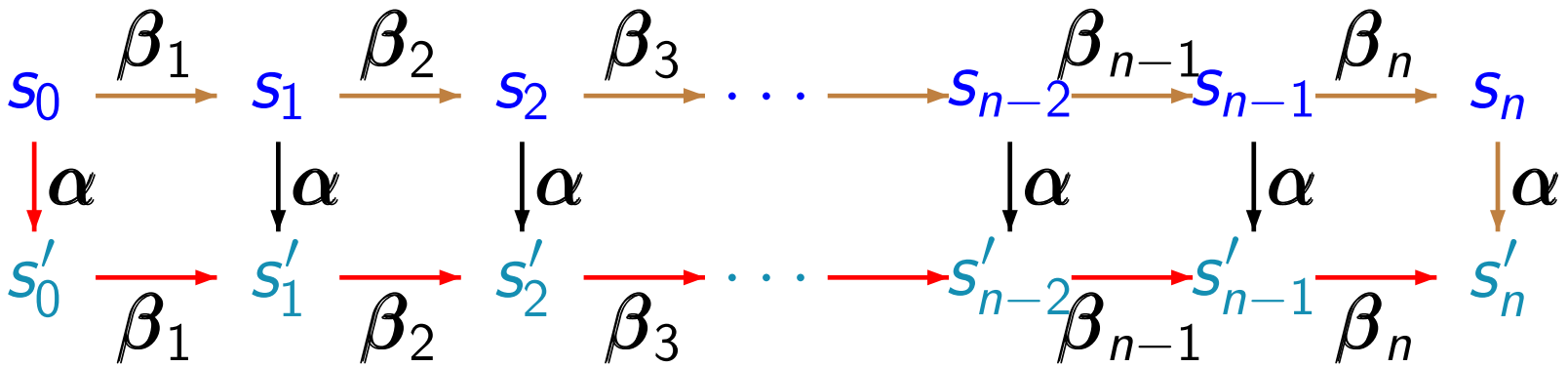


case 1:

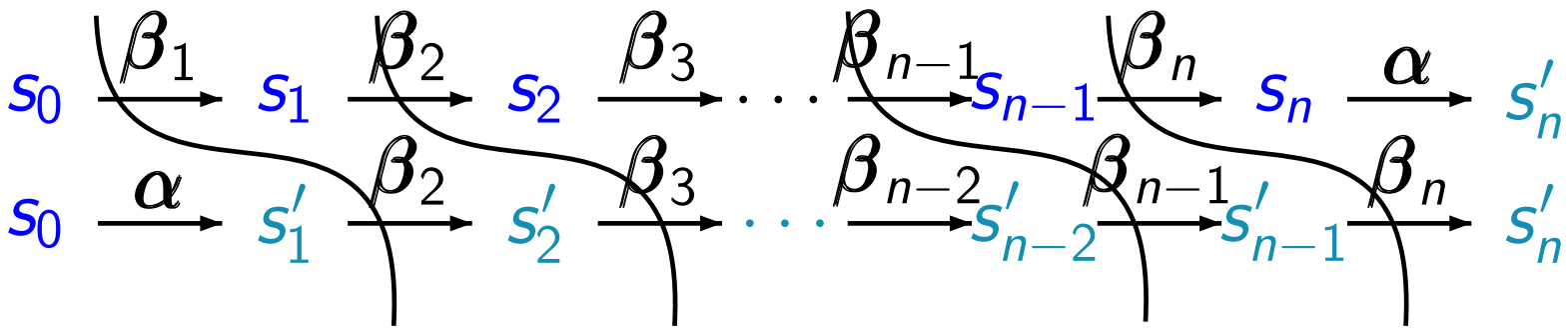


Conditions (A2) and (A3)

- $\alpha \in \text{ample}(s_0)$, $\beta_i \notin \text{ample}(s_0)$
- α stutter action $\Rightarrow L(s_i) = L(s'_i)$, $i = 0, 1, 2, \ldots$

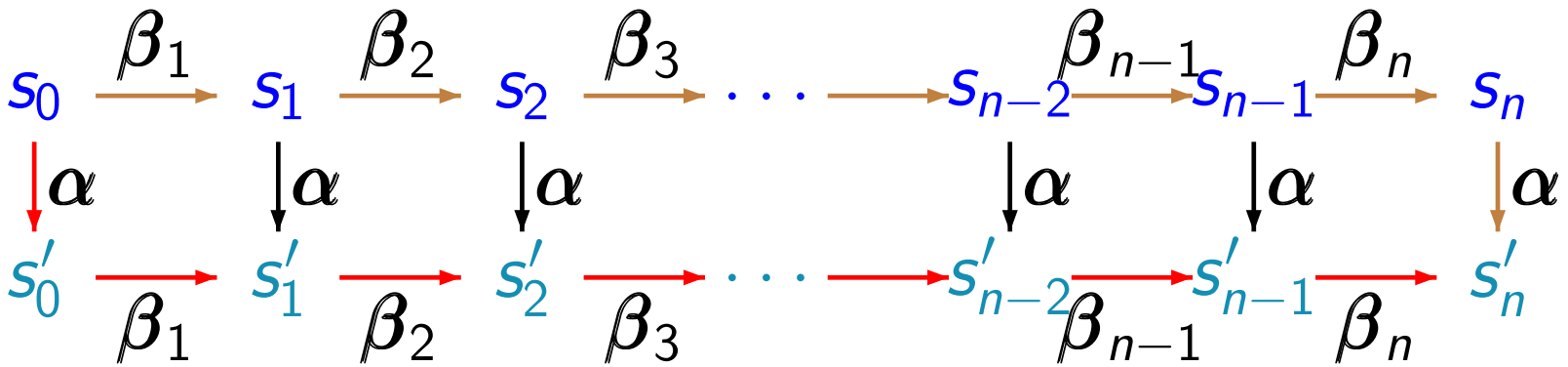


case 1:

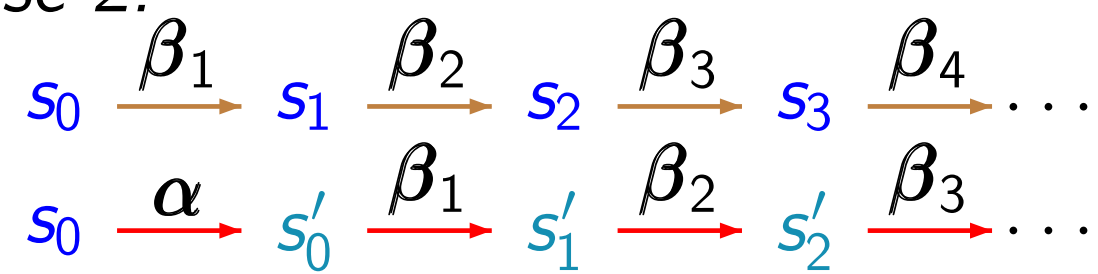


Conditions (A2) and (A3)

- $\alpha \in \text{ample}(s_0)$, $\beta_i \notin \text{ample}(s_0)$
- α stutter action $\Rightarrow L(s_i) = L(s'_i)$, $i = 0, 1, 2, \ldots$

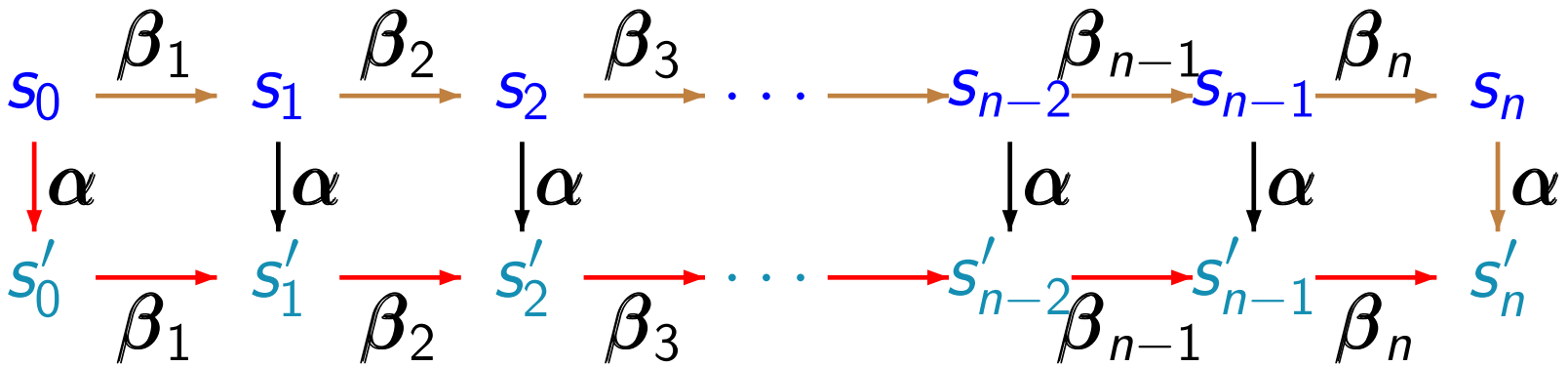


case 2:

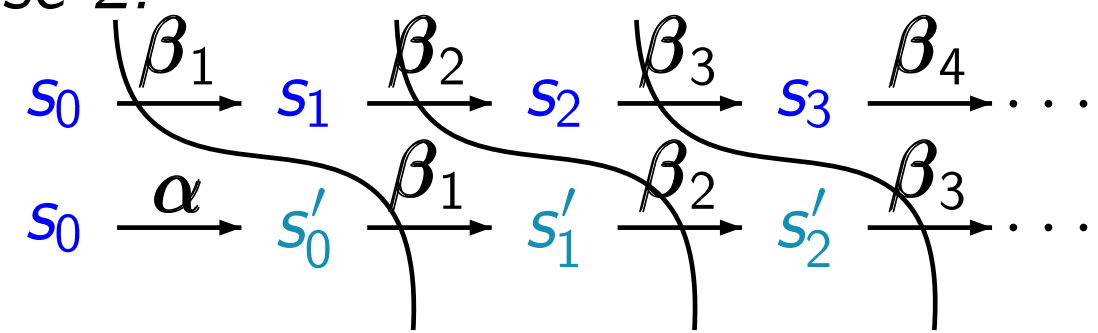


Conditions (A2) and (A3)

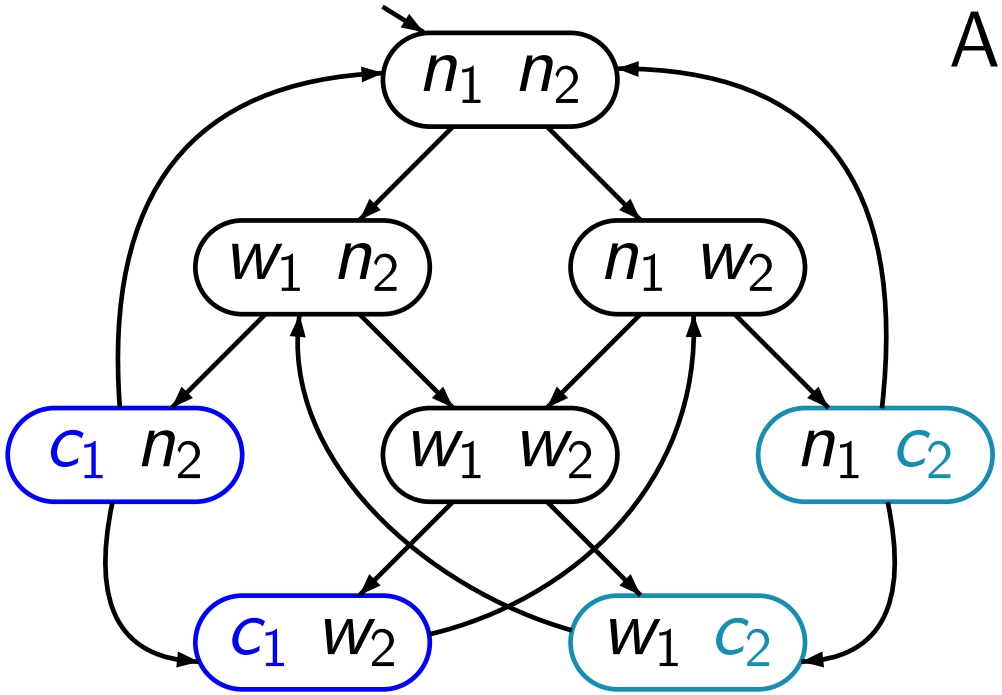
- $\alpha \in \text{ample}(s_0)$, $\beta_i \notin \text{ample}(s_0)$
- α stutter action $\Rightarrow L(s_i) = L(s'_i)$, $i = 0, 1, 2, \ldots$



case 2:

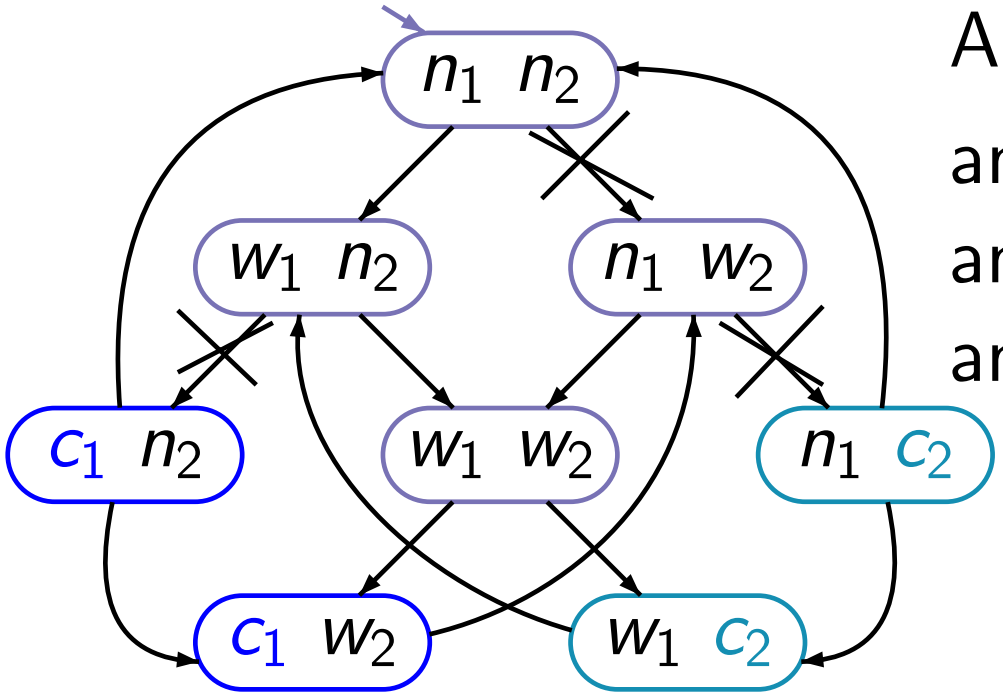


Ample sets for MUTEX



$$\mathrm{AP} = \{c_1, c_2\}$$

Ample sets for MUTEX



$$\mathrm{AP} = \{c_1, c_2\}$$

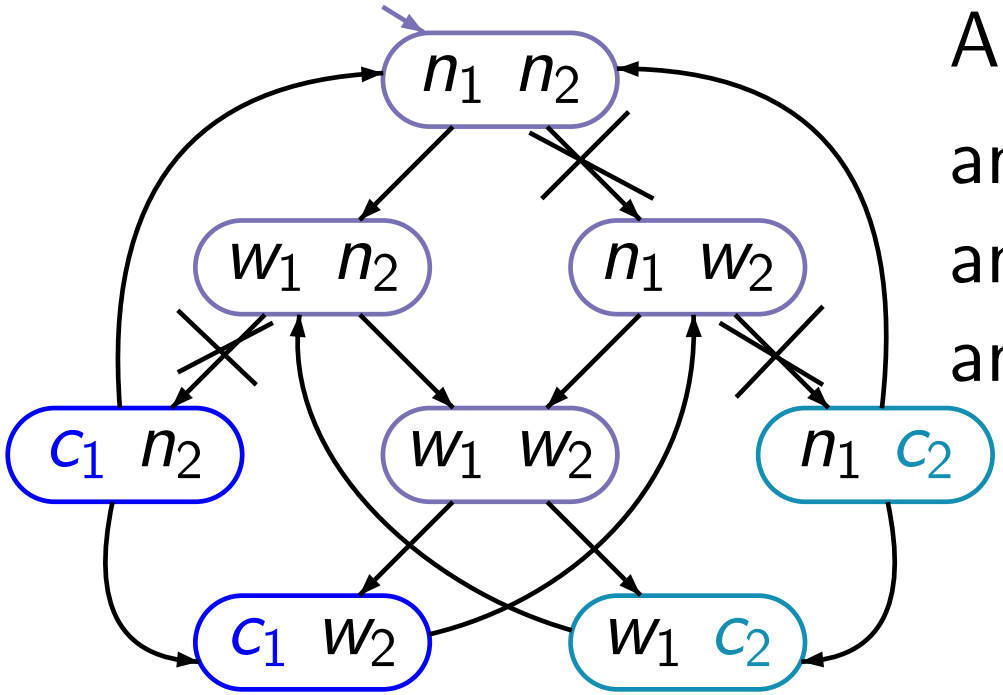
$$\mathrm{ample}(n_1, n_2) = \{\mathrm{request}_1\}$$

$$\mathrm{ample}(w_1, n_2) = \{\mathrm{request}_2\}$$

$$\mathrm{ample}(w_1, w_2) = \{\mathrm{enter}_1, \mathrm{enter}_2\}$$

$$\ldots$$

Ample sets for MUTEX



$\mathrm{AP} = \{c_1, c_2\}$

$\mathrm{ample}(n_1, n_2) = \{\mathrm{request}_1\}$

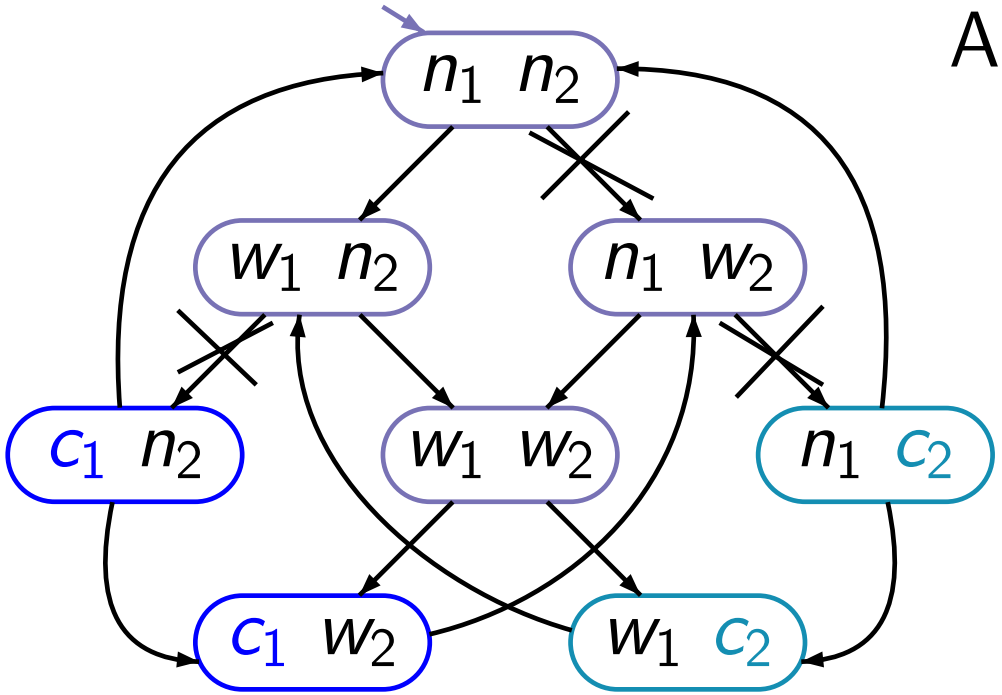
$\mathrm{ample}(w_1, n_2) = \{\mathrm{request}_2\}$

$\mathrm{ample}(w_1, w_2) = \{\mathrm{enter}_1, \mathrm{enter}_2\}$

$\ldots$

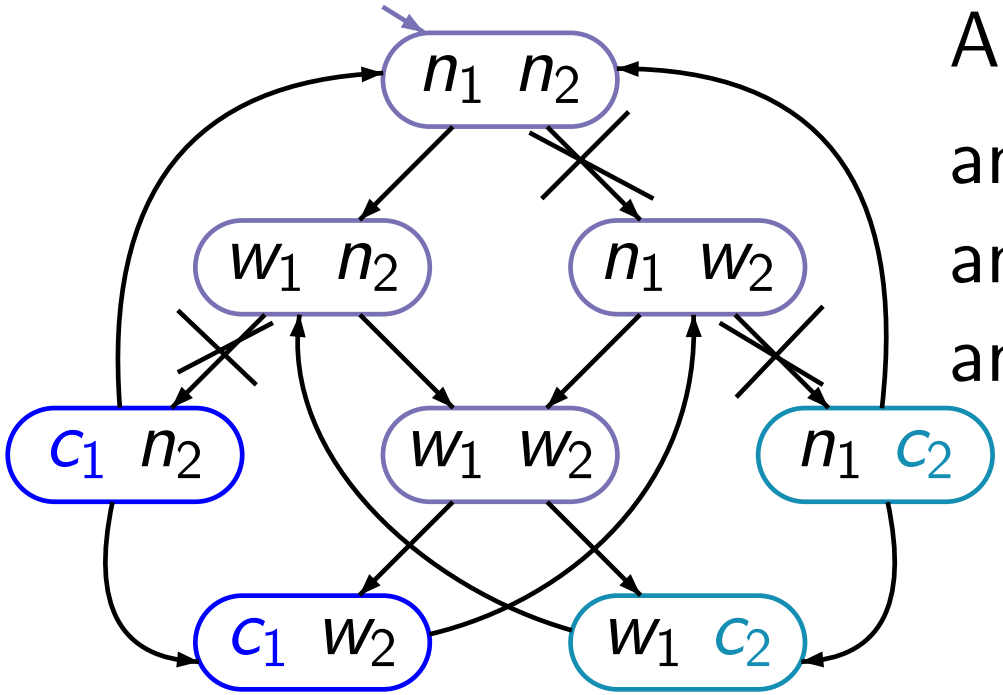
(A1), (A2), (A3) are satisfied

Ample sets for MUTEX



$$\mathrm{AP} = \{c_1, c_2\}$$

Ample sets for MUTEX



$$\mathrm{AP} = \{c_1, c_2\}$$

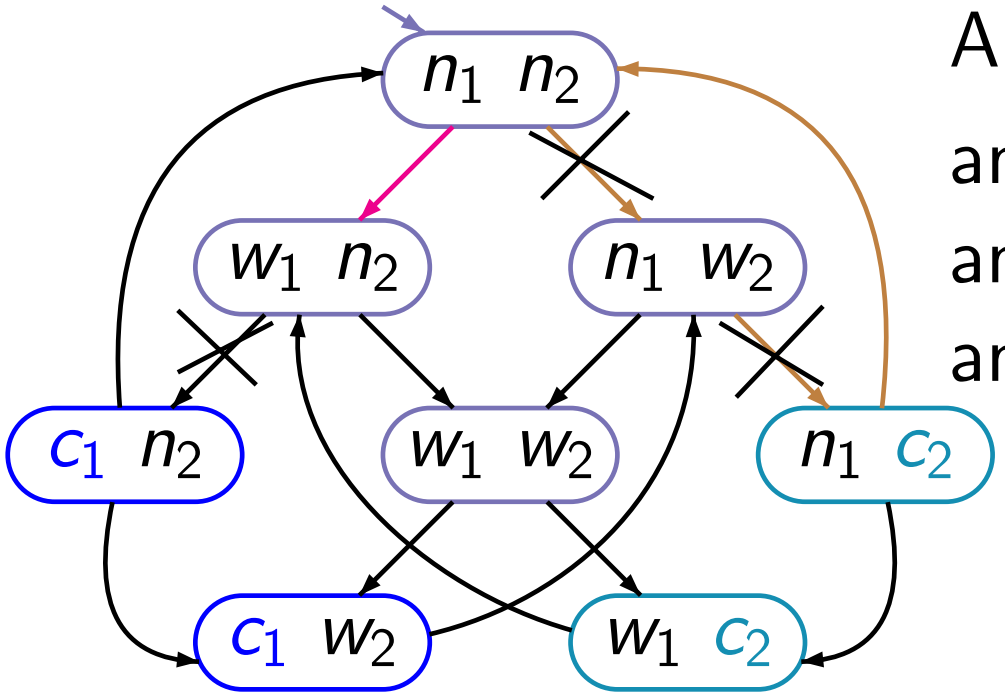
$$\mathrm{ample}(n_1, n_2) = \{\mathrm{request}_1\}$$

$$\mathrm{ample}(w_1, n_2) = \{\mathrm{request}_2\}$$

$$\mathrm{ample}(w_1, w_2) = \{\mathrm{enter}_1, \mathrm{enter}_2\}$$

$$\ldots$$

Ample sets for MUTEX



$\mathrm{AP} = \{c_1, c_2\}$

$\mathrm{ample}(n_1, n_2) = \{\mathrm{request}_1\}$

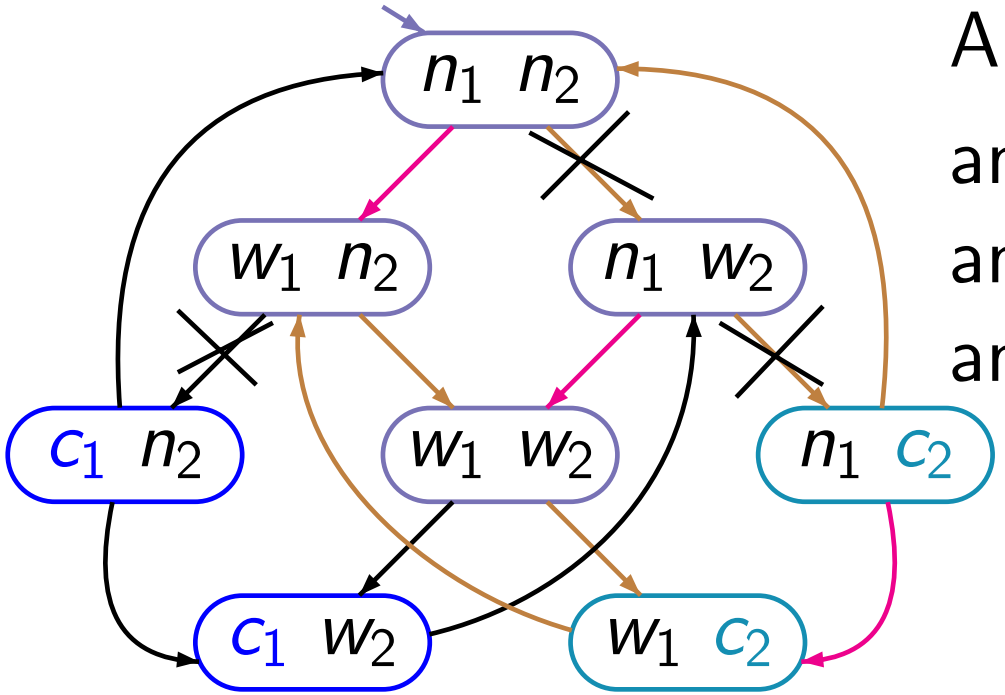
$\mathrm{ample}(w_1, n_2) = \{\mathrm{request}_2\}$

$\mathrm{ample}(w_1, w_2) = \{\mathrm{enter}_1, \mathrm{enter}_2\}$

$\ldots$

$$n_1 n_2 \xrightarrow{\mathrm{request}_2} n_1 w_2 \xrightarrow{\mathrm{enter}_2} n_1 c_2 \xrightarrow{\mathrm{release}_2} n_1 n_2 \xrightarrow{\mathrm{request}_1} w_1 n_2$$

Ample sets for MUTEX



$$\mathrm{AP} = \{c_1, c_2\}$$

$$\mathrm{ample}(n_1, n_2) = \{\mathrm{request}_1\}$$

$$\mathrm{ample}(w_1, n_2) = \{\mathrm{request}_2\}$$

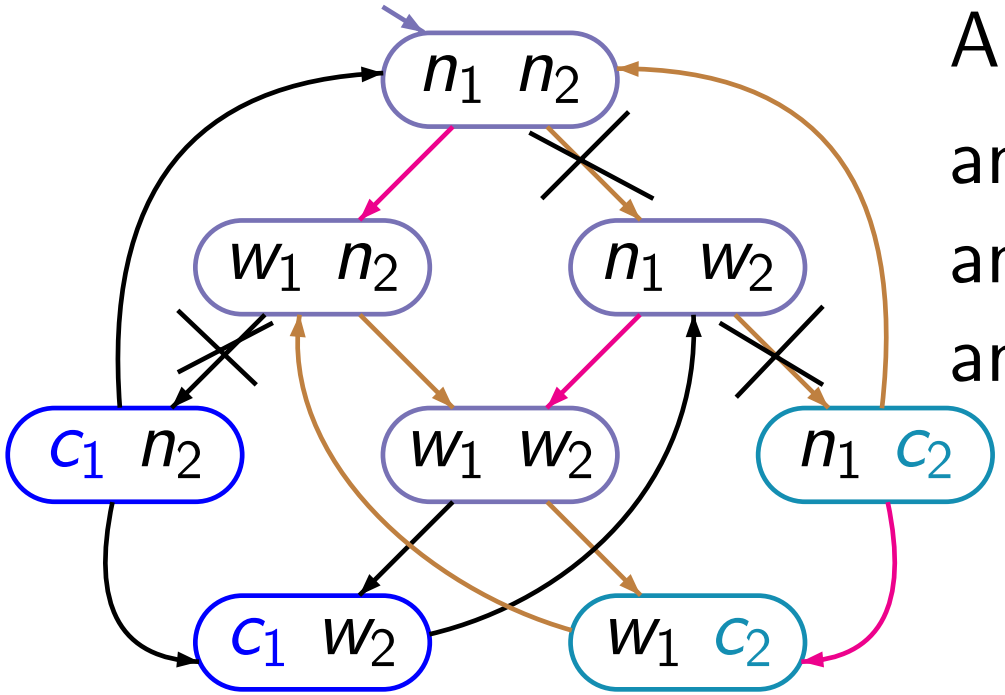
$$\mathrm{ample}(w_1, w_2) = \{\mathrm{enter}_1, \mathrm{enter}_2\}$$

$$\ldots$$

$$n_1 n_2 \xrightarrow{\mathrm{request}_2} n_1 w_2 \xrightarrow{\mathrm{enter}_2} n_1 c_2 \xrightarrow{\mathrm{release}_2} n_1 n_2 \xrightarrow{\mathrm{request}_1} w_1 n_2$$

$$n_1 n_2 \xrightarrow{\mathrm{request}_2} n_1 w_2 \xrightarrow{\mathrm{enter}_2} n_1 c_2 \xrightarrow{\mathrm{request}_1} w_1 c_2 \xrightarrow{\mathrm{release}_2} w_1 n_2$$

Ample sets for MUTEX



$$AP = \{c_1, c_2\}$$

$$\text{ample}(n_1, n_2) = \{\text{request}_1\}$$

$$\text{ample}(w_1, n_2) = \{\text{request}_2\}$$

$$\text{ample}(w_1, w_2) = \{\text{enter}_1, \text{enter}_2\}$$

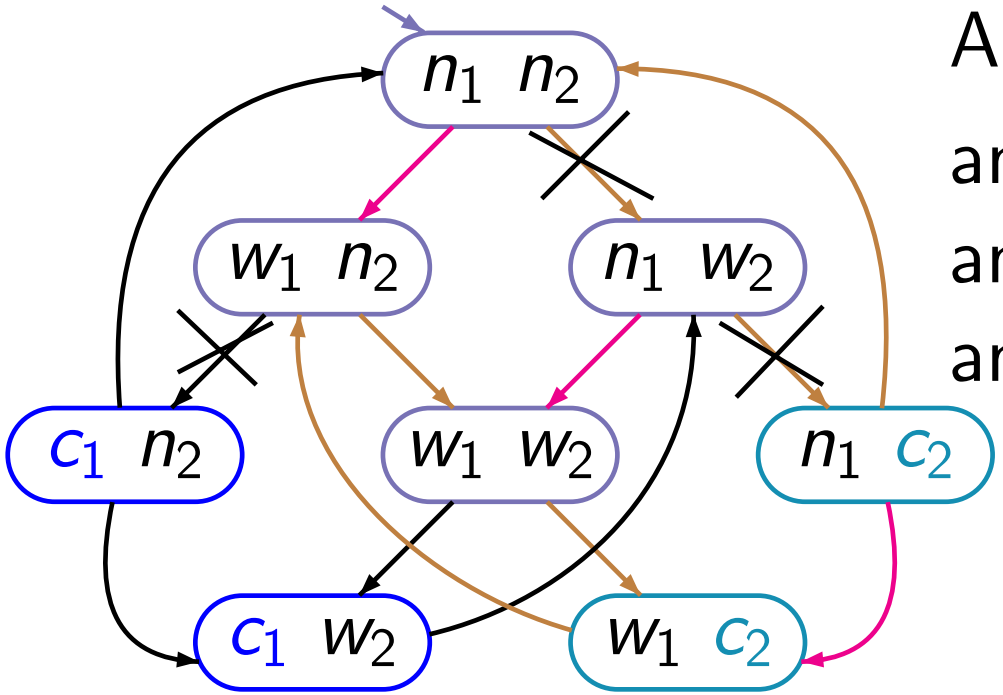
$$\ldots$$

$$n_1 n_2 \xrightarrow{\text{request}_2} n_1 w_2 \xrightarrow{\text{enter}_2} n_1 c_2 \xrightarrow{\text{release}_2} n_1 n_2 \xrightarrow{\text{request}_1} w_1 n_2$$

$$n_1 n_2 \xrightarrow{\text{request}_2} n_1 w_2 \xrightarrow{\text{enter}_2} n_1 c_2 \xrightarrow{\text{request}_1} w_1 c_2 \xrightarrow{\text{release}_2} w_1 n_2$$

$$n_1 n_2 \xrightarrow{\text{request}_2} n_1 w_2 \xrightarrow{\text{request}_1} w_1 w_2 \xrightarrow{\text{enter}_2} w_1 c_2 \xrightarrow{\text{release}_2} w_1 n_2$$

Ample sets for MUTEX



$$AP = \{c_1, c_2\}$$

$$\text{ample}(n_1, n_2) = \{\text{request}_1\}$$

$$\text{ample}(w_1, n_2) = \{\text{request}_2\}$$

$$\text{ample}(w_1, w_2) = \{\text{enter}_1, \text{enter}_2\}$$

$$\ldots$$

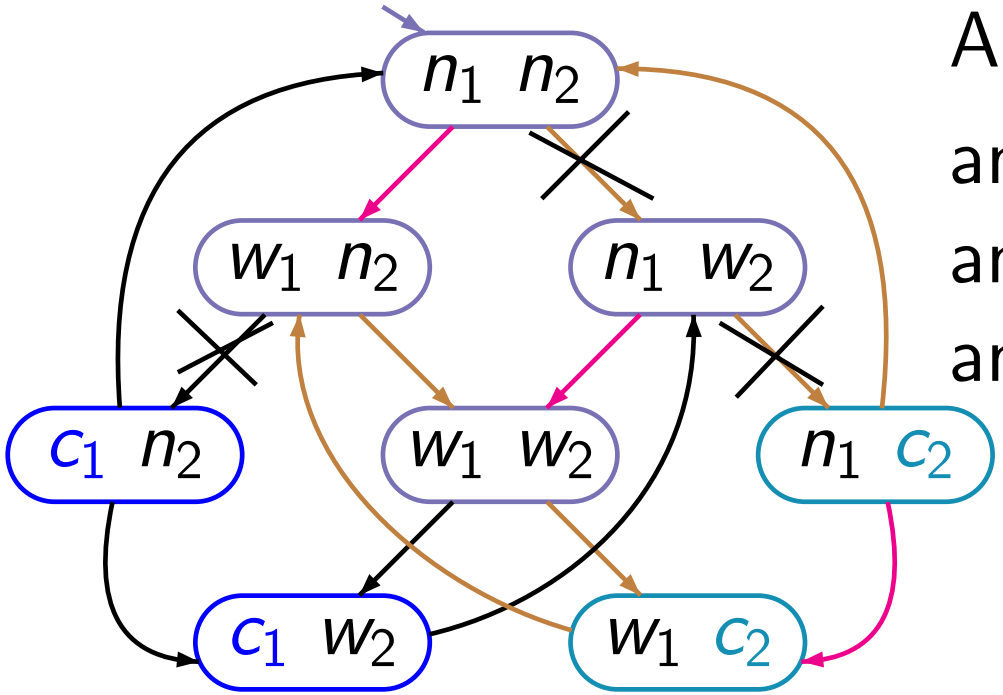
$$n_1 n_2 \xrightarrow{\text{request}_2} n_1 w_2 \xrightarrow{\text{enter}_2} n_1 c_2 \xrightarrow{\text{release}_2} n_1 n_2 \xrightarrow{\text{request}_1} w_1 n_2$$

$$n_1 n_2 \xrightarrow{\text{request}_2} n_1 w_2 \xrightarrow{\text{enter}_2} n_1 c_2 \xrightarrow{\text{request}_1} w_1 c_2 \xrightarrow{\text{release}_2} w_1 n_2$$

$$n_1 n_2 \xrightarrow{\text{request}_2} n_1 w_2 \xrightarrow{\text{request}_1} w_1 w_2 \xrightarrow{\text{enter}_2} w_1 c_2 \xrightarrow{\text{release}_2} w_1 n_2$$

$$n_1 n_2 \xrightarrow{\text{request}_1} w_1 n_2 \xrightarrow{\text{request}_2} w_1 w_2 \xrightarrow{\text{enter}_2} w_1 c_2 \xrightarrow{\text{release}_2} w_1 n_2$$

Ample sets for MUTEX



$$AP = \{c_1, c_2\}$$

$$\text{ample}(n_1, n_2) = \{\text{request}_1\}$$

$$\text{ample}(w_1, n_2) = \{\text{request}_2\}$$

$$\text{ample}(w_1, w_2) = \{\text{enter}_1, \text{enter}_2\}$$

$$\ldots$$

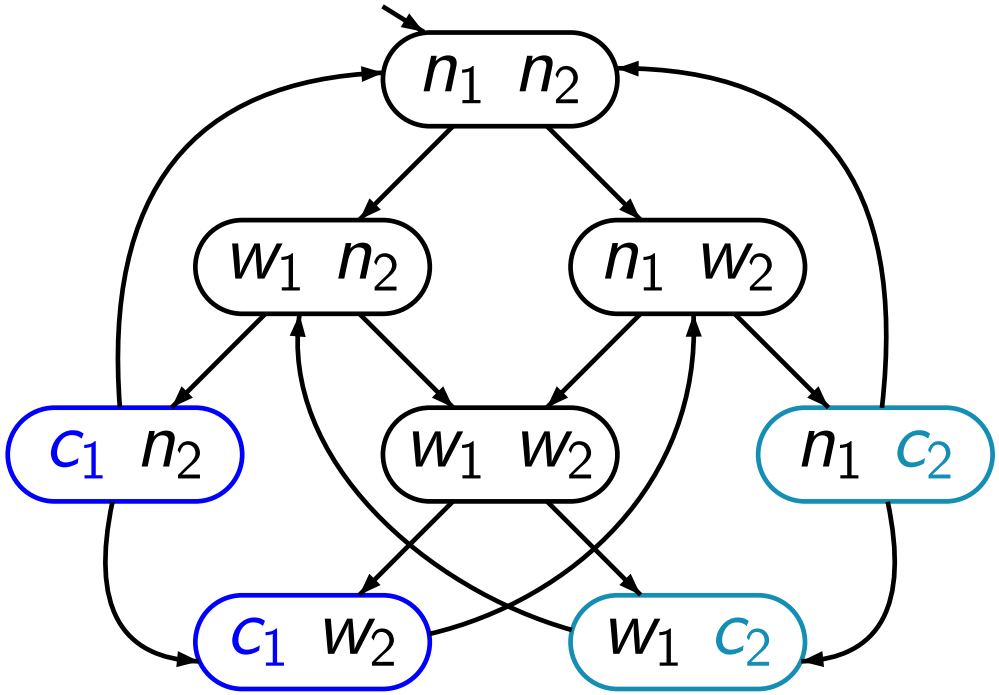
$$n_1 n_2 \xrightarrow{\text{request}_2} n_1 w_2 \xrightarrow{\text{enter}_2} n_1 c_2 \xrightarrow{\text{release}_2} n_1 n_2 \xrightarrow{\text{request}_1} w_1 n_2$$

$$n_1 n_2 \xrightarrow{\text{request}_2} n_1 w_2 \xrightarrow{\text{enter}_2} n_1 c_2 \xrightarrow{\text{request}_1} w_1 c_2 \xrightarrow{\text{release}_2} w_1 n_2$$

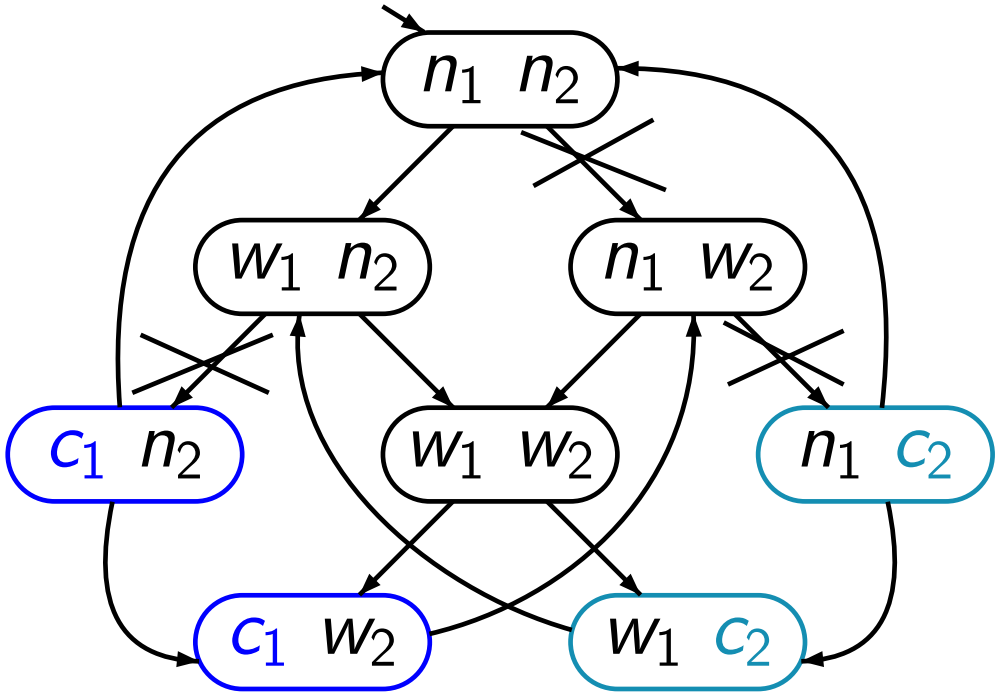
$$n_1 n_2 \xrightarrow{\text{request}_2} n_1 w_2 \xrightarrow{\text{request}_1} w_1 w_2 \xrightarrow{\text{enter}_2} w_1 c_2 \xrightarrow{\text{release}_2} w_1 n_2$$

$$n_1 n_2 \xrightarrow{\text{request}_1} w_1 n_2 \xrightarrow{\text{request}_2} w_1 w_2 \xrightarrow{\text{enter}_2} w_1 c_2 \xrightarrow{\text{release}_2} w_1 n_2$$

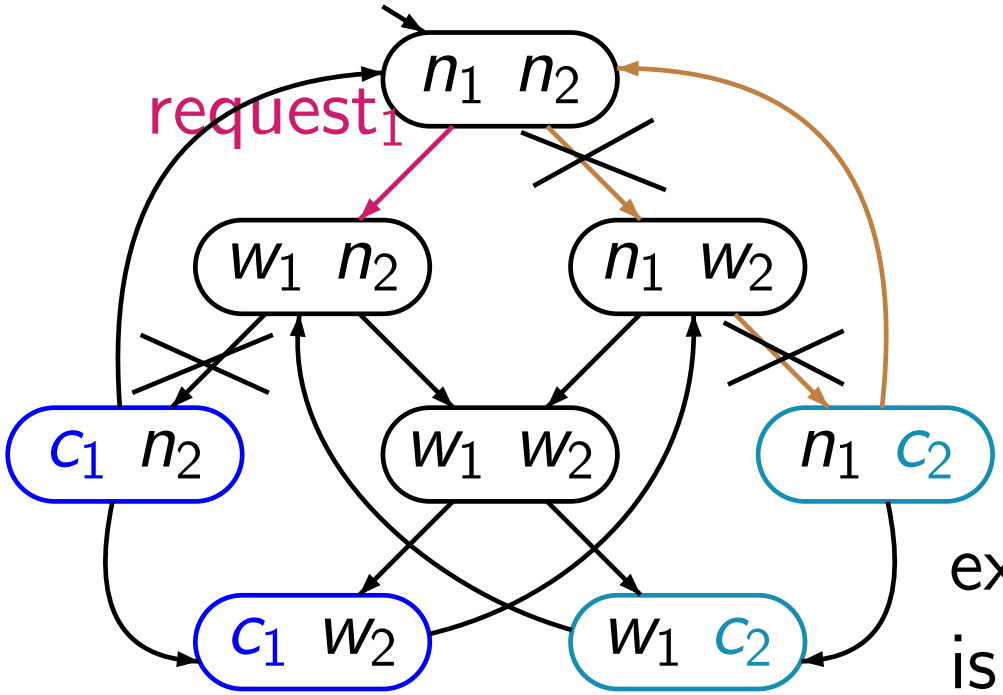
Example: case 2



Example: case 2



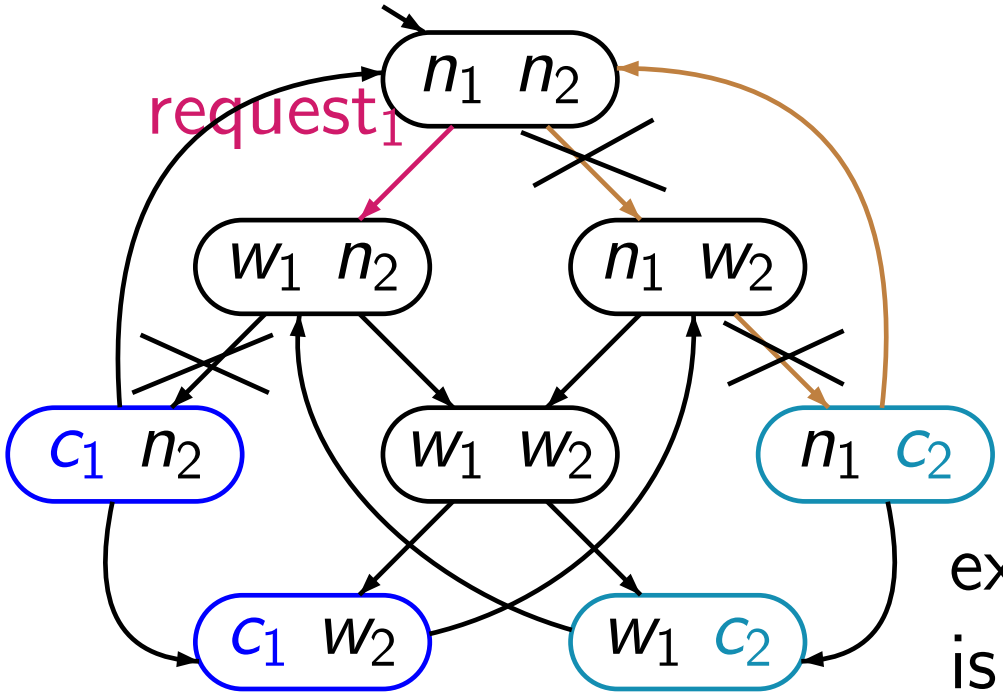
Example: case 2



execution where request$_1$ is not executed

$$n_1 n_2 \xrightarrow{\text{requ}_2} n_1 w_2 \xrightarrow{\text{enter}_2} n_1 c_2 \xrightarrow{\text{release}_2} n_1 n_2 \xrightarrow{\text{requ}_2} \ldots$$

Example: case 2

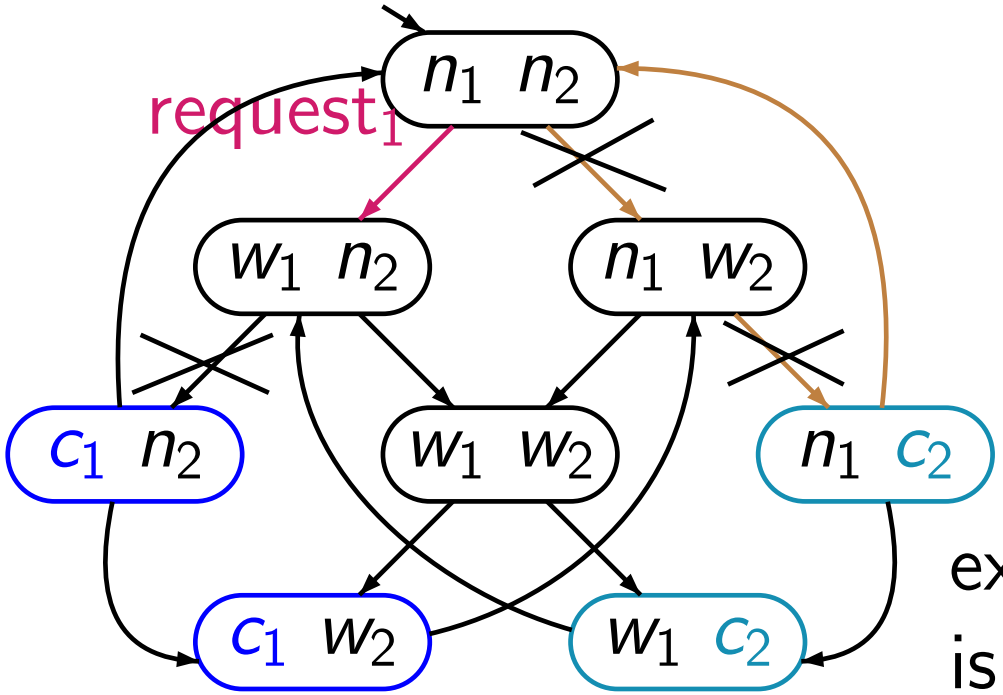


execution where request$_1$ is not executed

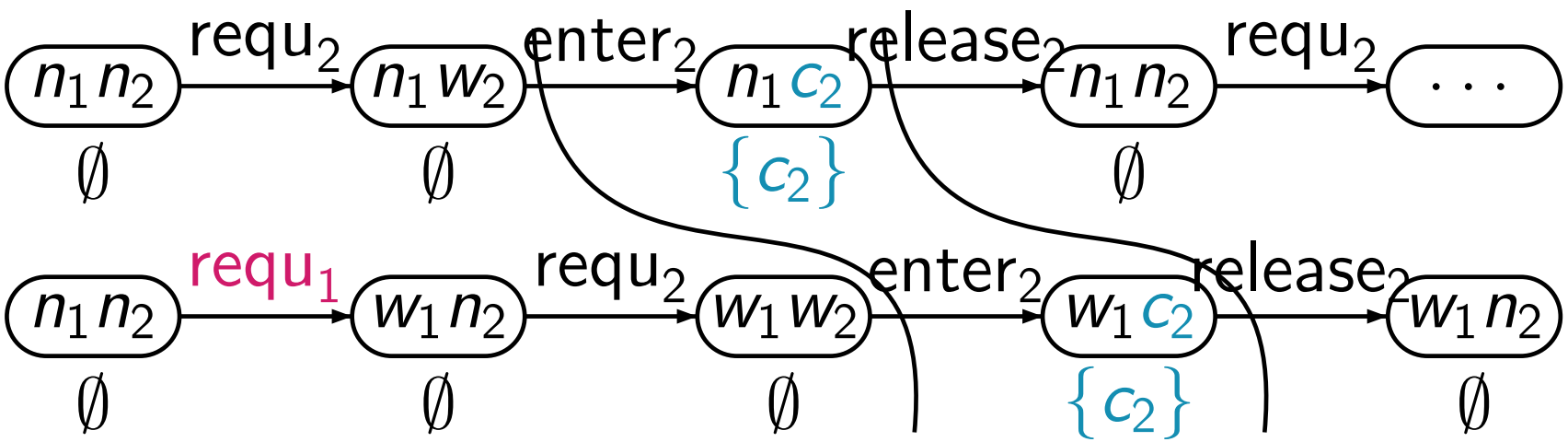
$$n_1 n_2 \xrightarrow{\text{requ}_2} n_1 w_2 \xrightarrow{\text{enter}_2} n_1 c_2 \xrightarrow{\text{release}_2} n_1 n_2 \xrightarrow{\text{requ}_2} \ldots$$

$$n_1 n_2 \xrightarrow{\text{requ}_1} w_1 n_2 \xrightarrow{\text{requ}_2} w_1 w_2 \xrightarrow{\text{enter}_2} w_1 c_2 \xrightarrow{\text{release}_2} w_1 n_2$$

Example: case 2



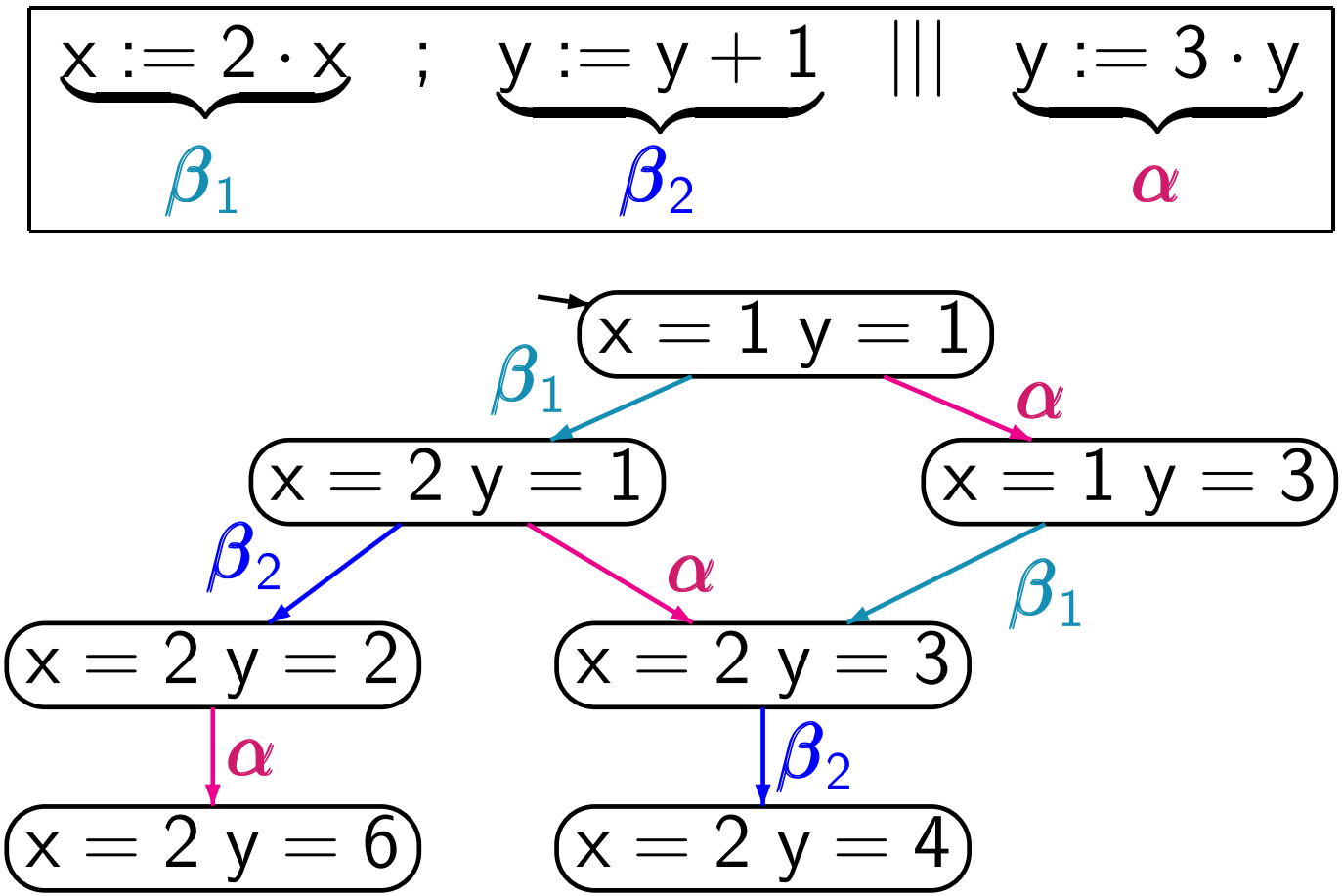
execution where request$_1$ is not executed



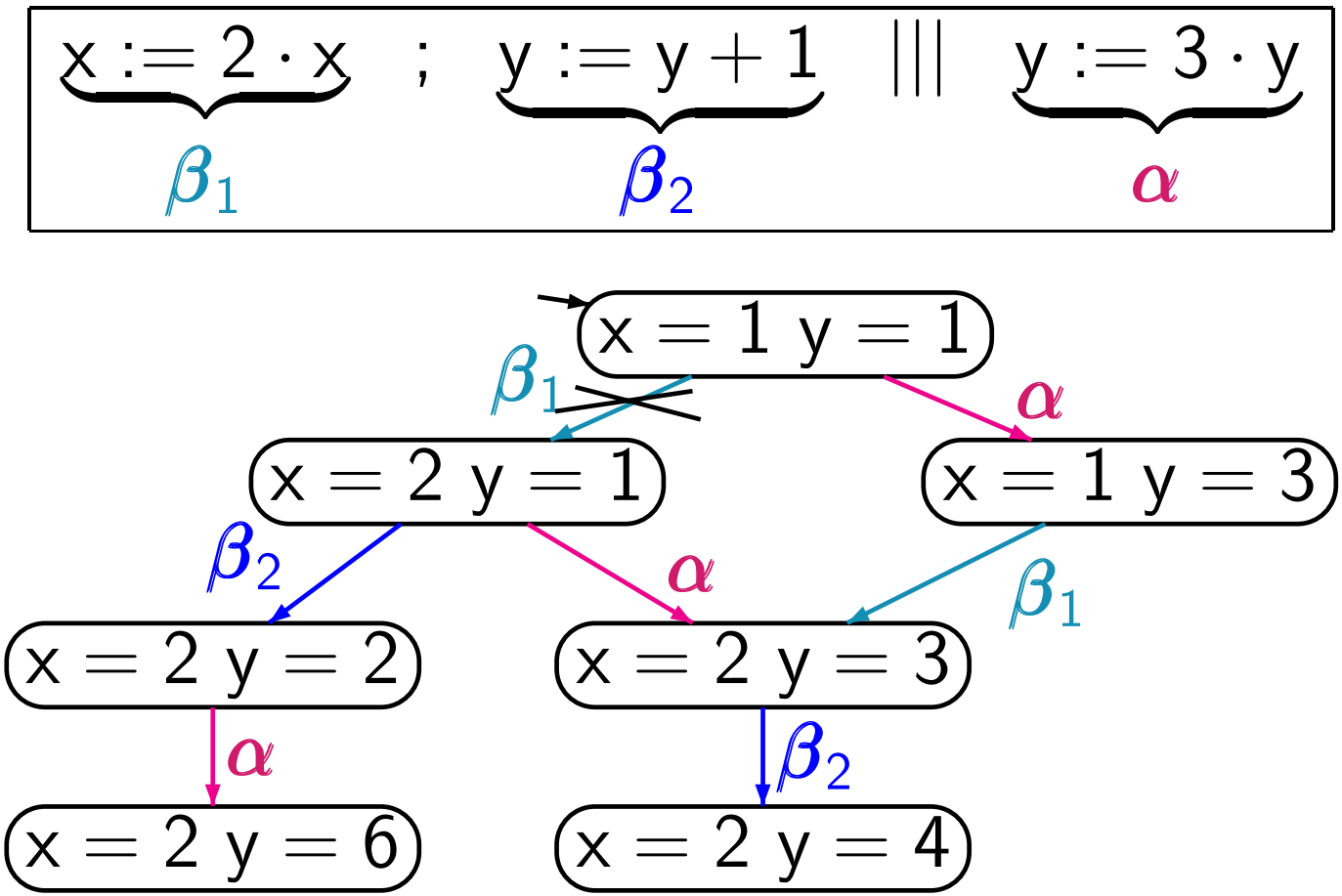
Example

$$\underbrace{x := 2 \cdot x}_{\beta_1} \;;\; \underbrace{y := y + 1}_{\beta_2} \;|||\; \underbrace{y := 3 \cdot y}_{\alpha}$$

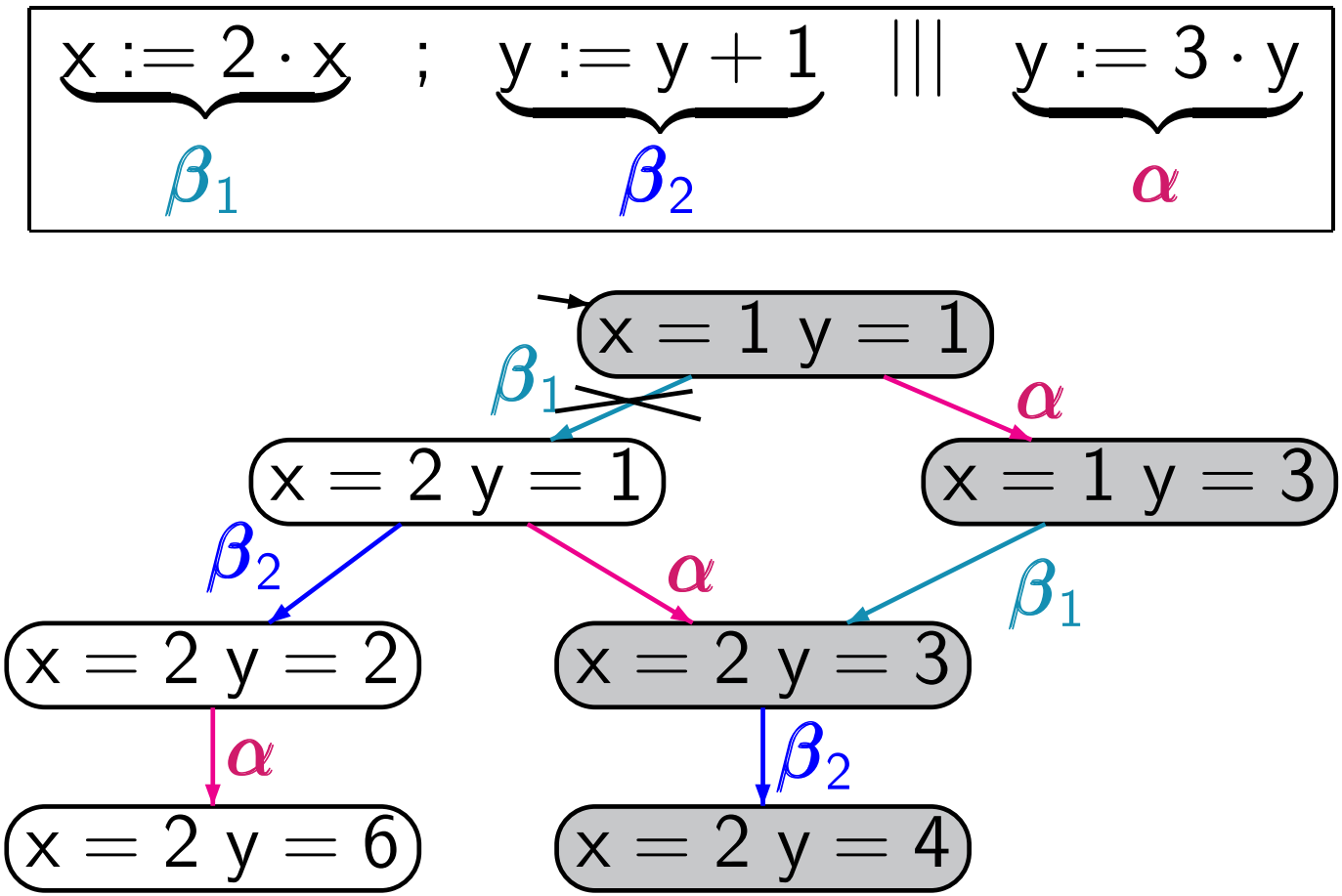
Example



Example

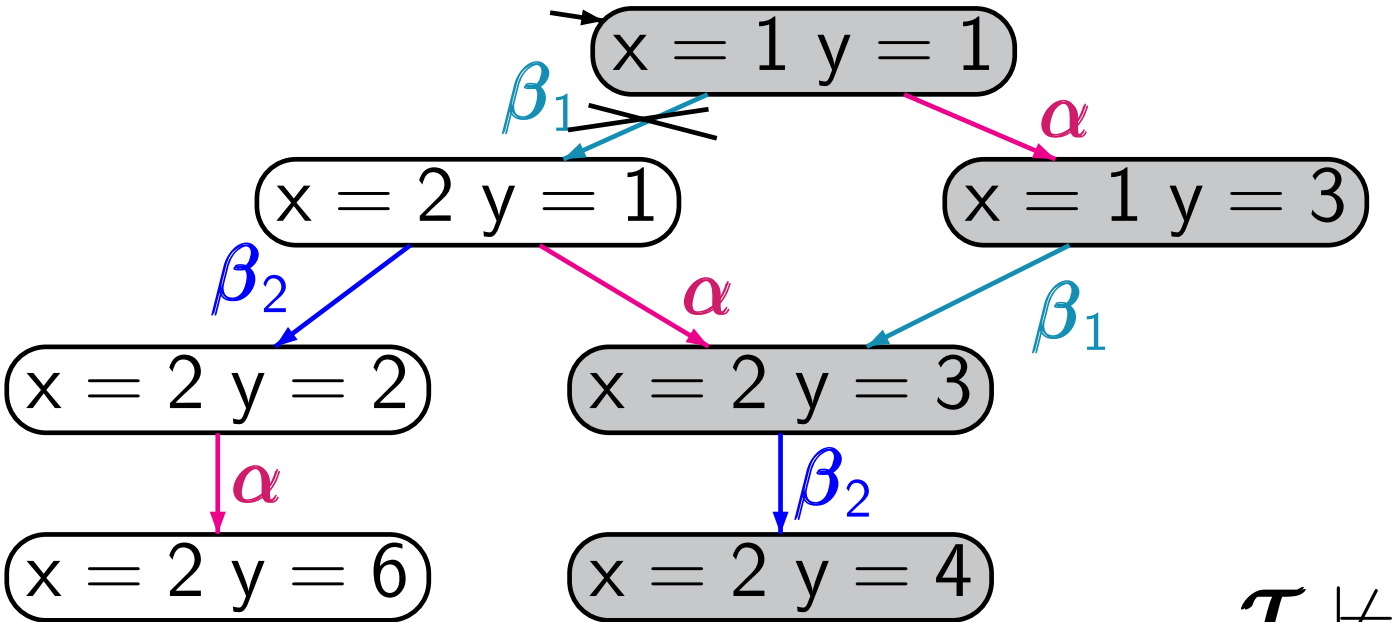


Example



Example

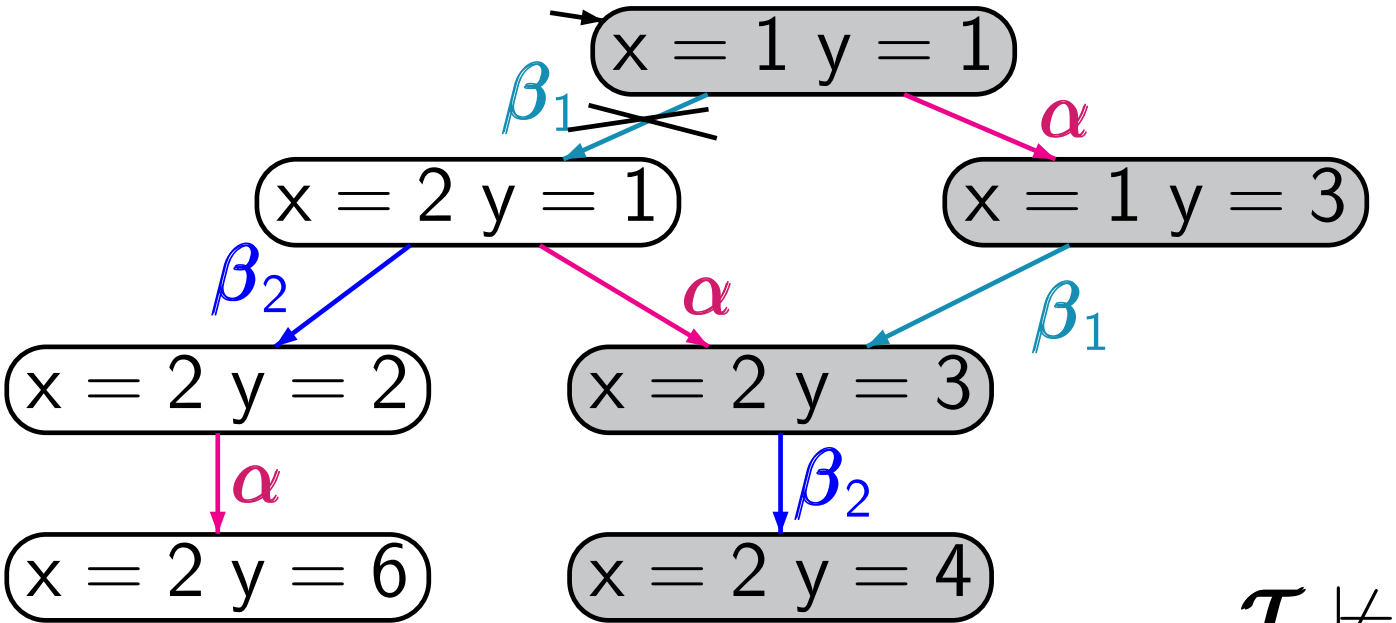
$$\underbrace{x := 2 \cdot x}_{\beta_1} \;;\; \underbrace{y := y + 1}_{\beta_2} \;|||\; \underbrace{y := 3 \cdot y}_{\alpha}$$



$$\mathcal{T} \not\models \Box(y \neq 6)$$
$$\mathcal{T}_{\text{red}} \models \Box(y \neq 6)$$

Example

$$\underbrace{x := 2 \cdot x}_{\beta_1} \;;\; \underbrace{y := y + 1}_{\beta_2} \;|||\; \underbrace{y := 3 \cdot y}_{\alpha}$$



$$\mathcal{T} \not\models \Box(y \neq 6)$$
$$\mathcal{T}_{\text{red}} \models \Box(y \neq 6)$$

(A2) violated as β_2, α dependent

Correct or wrong?

Let $\mathcal{T}_1, \mathcal{T}_2$ be TS with disjoint action-sets Act_1 and Act_2, respectively, and

$$\mathcal{T} = \mathcal{T}_1 \,|||\, \mathcal{T}_2$$

remind: $|||$ denotes full interleaving

Correct or wrong?

Let $\mathcal{T}_1, \mathcal{T}_2$ be TS with disjoint action-sets Act_1 and Act_2, respectively, and

$$\mathcal{T} = \mathcal{T}_1 \,|||\, \mathcal{T}_2$$

Then, the ample sets given by

$$\text{ample}(\langle s_1, s_2 \rangle) = \begin{cases} Act_1(s_1) \text{: if every act. in } Act_1(s_1) \\ \qquad\qquad \text{is stutter } \cancel{a} \text{ action} \\ Act_1(s_1) \cup Act_2(s_2) \text{: else} \end{cases}$$

satisfy $(\mathrm{A2})$ and $(\mathrm{A3})$.

Correct or wrong?

Let $\mathcal{T}_1, \mathcal{T}_2$ be TS with disjoint action-sets Act_1 and Act_2, respectively, and

$$\mathcal{T} = \mathcal{T}_1 \,|||\, \mathcal{T}_2$$

Then, the ample sets given by

$$\text{ample}(\langle s_1, s_2 \rangle) = \begin{cases} Act_1(s_1) \text{: if every act. in } Act_1(s_1) \\ \qquad\qquad \text{is stutter a action} \\ Act_1(s_1) \cup Act_2(s_2) \text{: else} \end{cases}$$

satisfy $(\mathrm{A2})$ and $(\mathrm{A3})$.

correct.

Note: all actions $\alpha \in Act_1$ and $\beta \in Act_2$ are independent in $\mathcal{T}$.

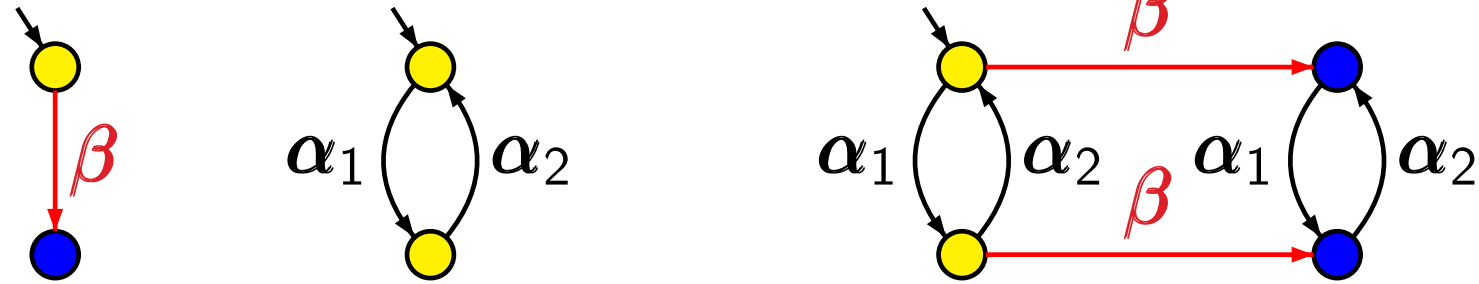
Conditions (A1), (A2), (A3) are not sufficient

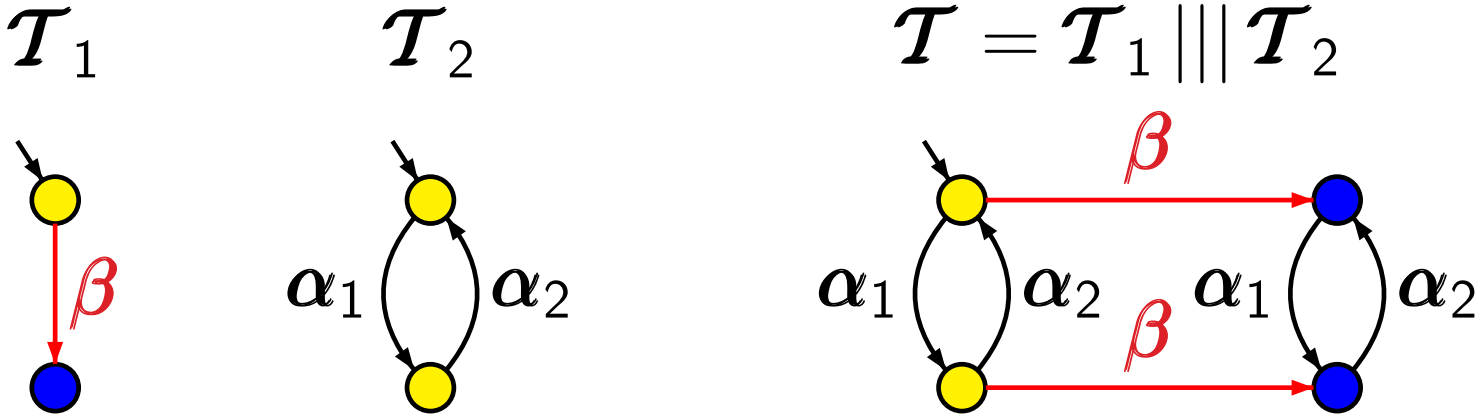
There exists a finite, action-deterministic transition system $\mathcal{T}$ and ample sets for $\mathcal{T}$ such that

$$\mathcal{T} \not\stackrel{\Delta}{=} \mathcal{T}_{\text{red}}$$

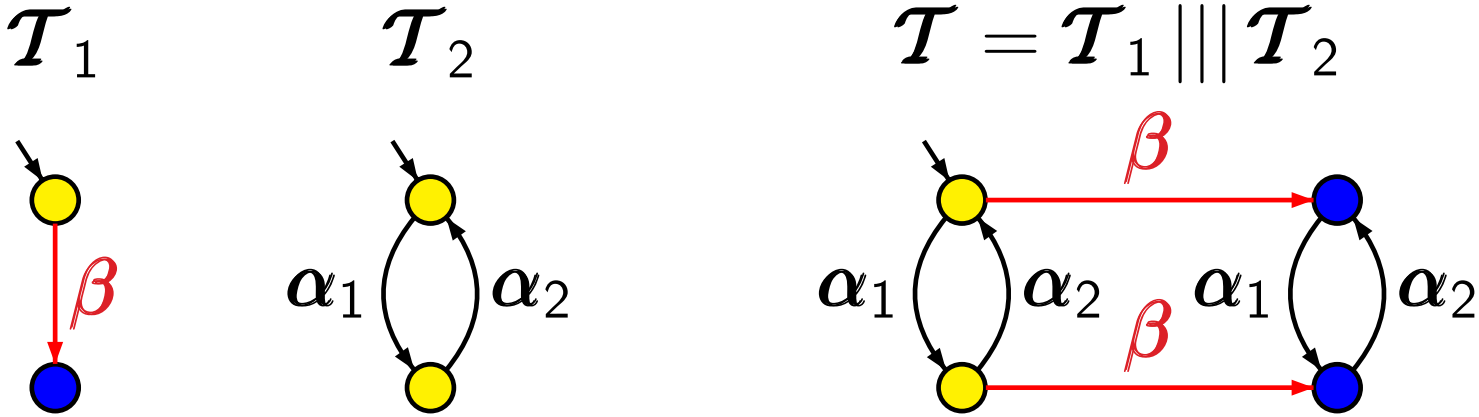
remind: $\stackrel{\Delta}{=}$ stutter trace equivalence

$\mathcal{T}_1$ $\mathcal{T}_2$ $\mathcal{T} = \mathcal{T}_1 ||| \mathcal{T}_2$





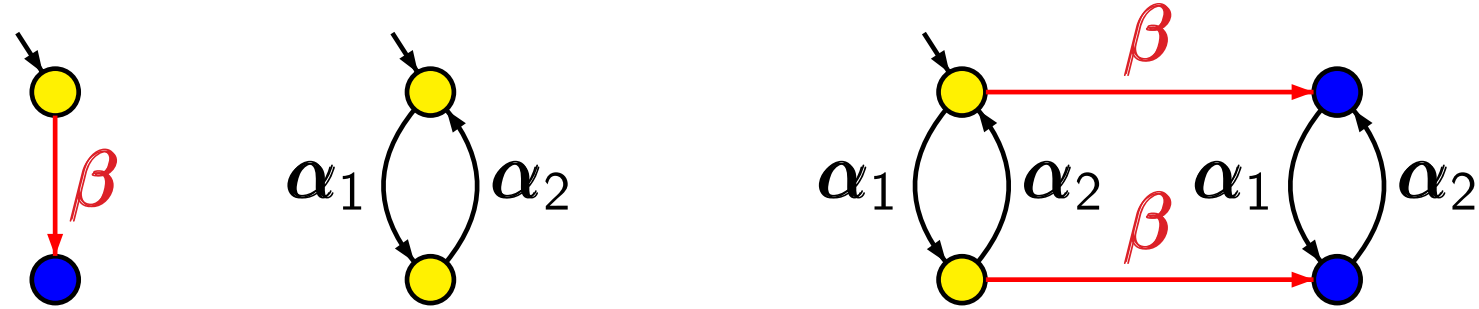
β, α_i independent



β, α_i independent

α_1, α_2 stutter actions

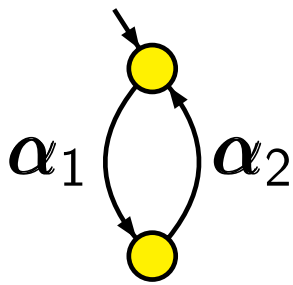
T_1 T_2 $T = T_1 \,|||\, T_2$



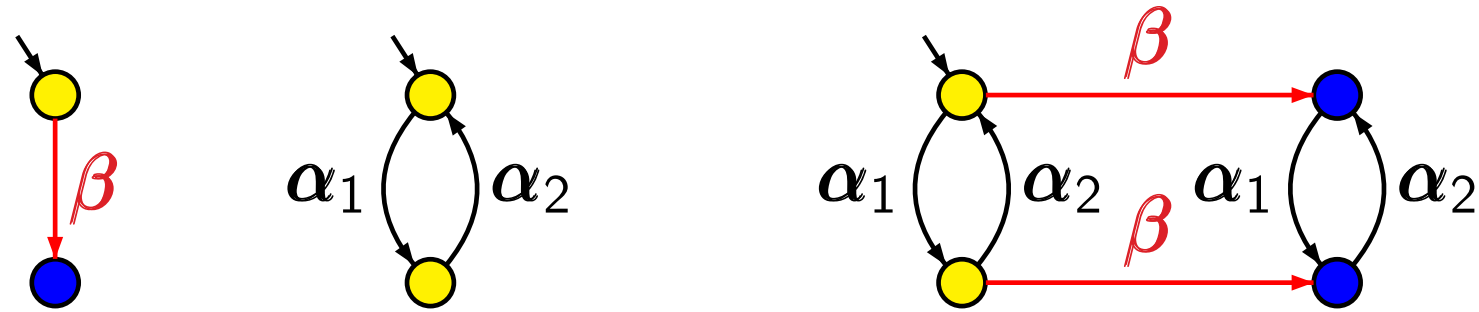
β, α_i independent

α_1, α_2 stutter actions

T_{red}



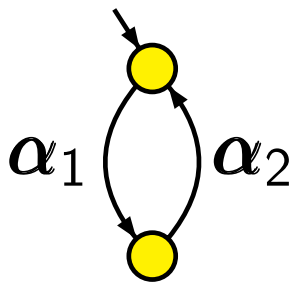
$\mathcal{T}_1$ $\mathcal{T}_2$ $\mathcal{T} = \mathcal{T}_1 \,|||\, \mathcal{T}_2$

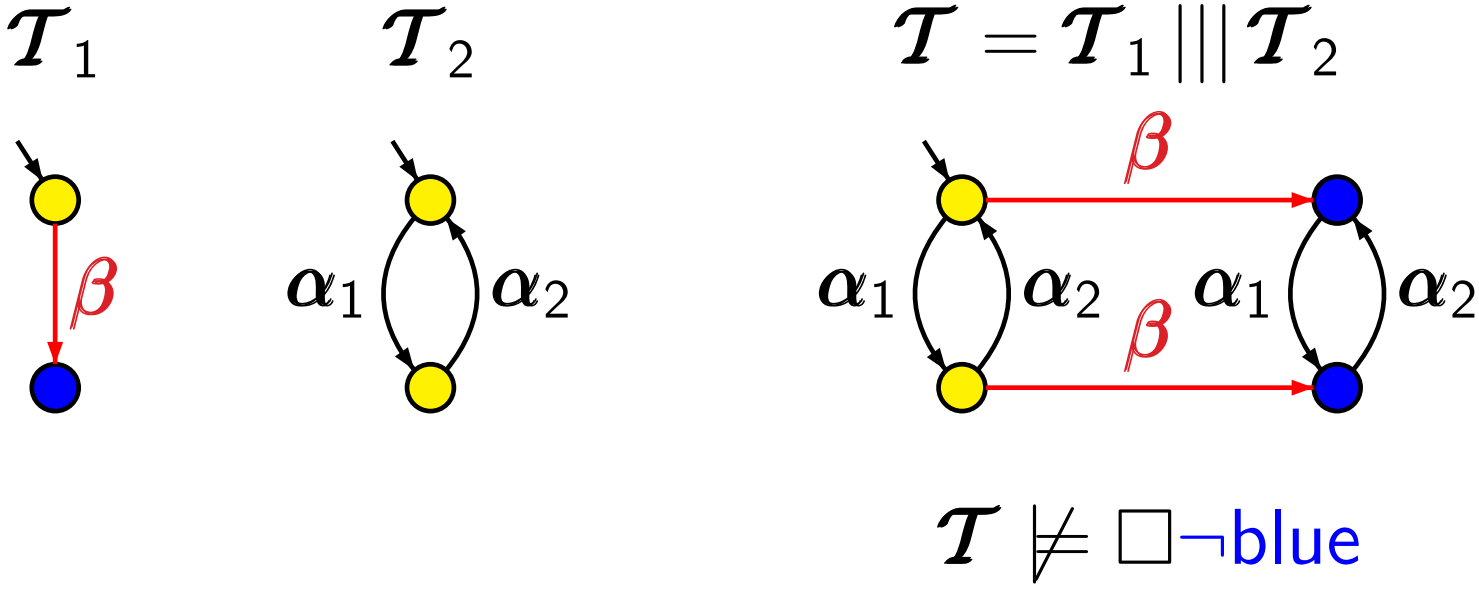


β, α_i independent

α_1, α_2 stutter actions

$\mathcal{T}_{\text{red}}$ satisfies (A1), (A2), (A3)



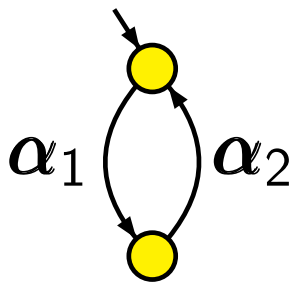


$\mathcal{T} \not\models \Box\neg$blue

β, α_i independent

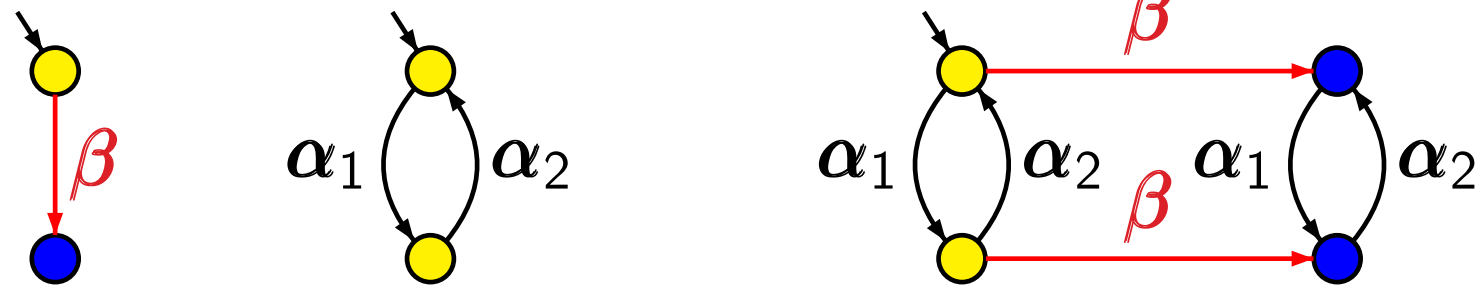
α_1, α_2 stutter actions

$\mathcal{T}_{\text{red}}$ satisfies (A1), (A2), (A3)



$\mathcal{T}_{\text{red}} \models \Box\neg$blue

$\mathcal{T}_1$ $\mathcal{T}_2$ $\mathcal{T} = \mathcal{T}_1 ||| \mathcal{T}_2$

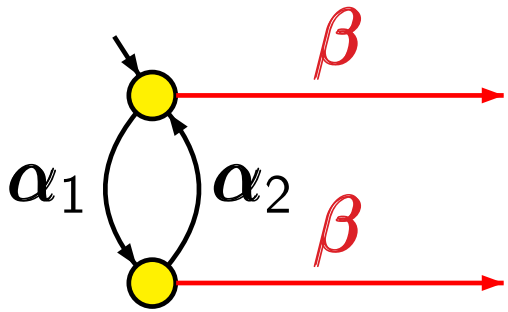


$\mathcal{T} \not\models \Box\neg$blue

β, α_i independent

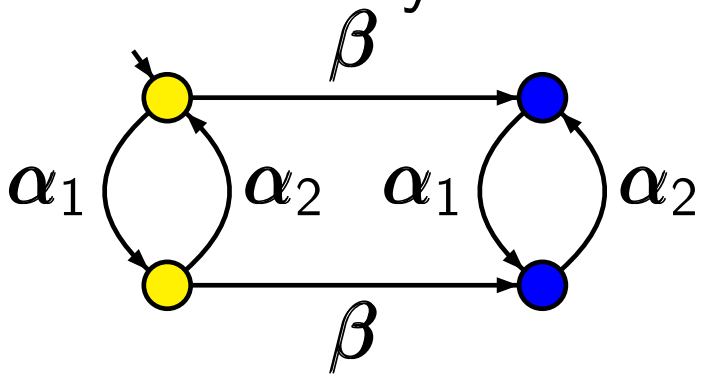
α_1, α_2 stutter actions

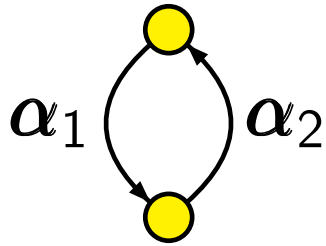
$\mathcal{T}_{\text{red}}$ satisfies (A1), (A2), (A3)

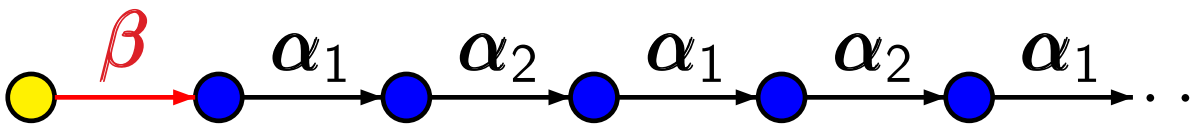


$\mathcal{T}_{\text{red}} \models \Box\neg$blue

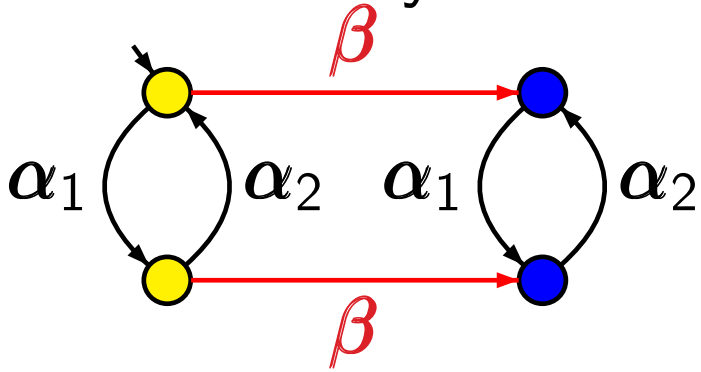
transition system $\mathcal{T}$ reduced TS $\mathcal{T}_{\text{red}}$

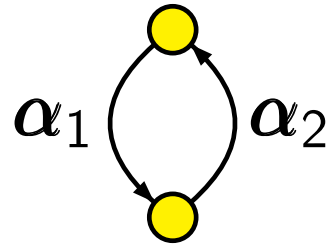


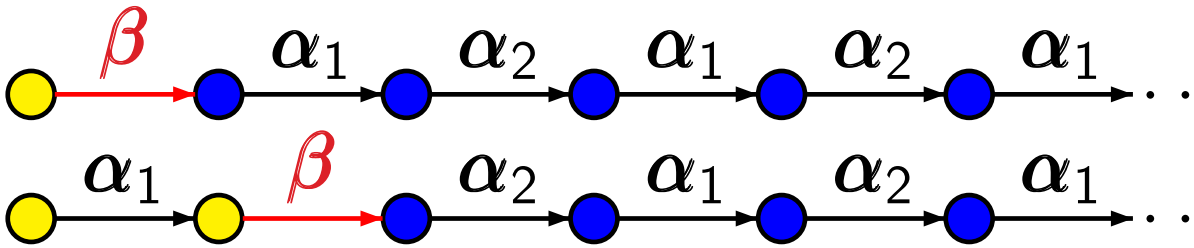




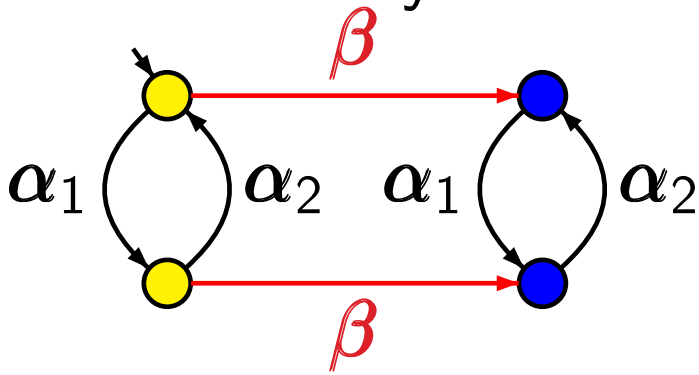
transition system $\mathcal{T}$ reduced TS $\mathcal{T}_{\text{red}}$

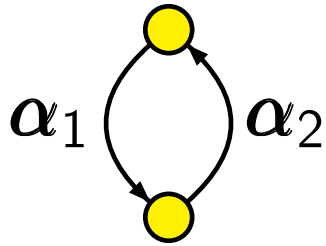


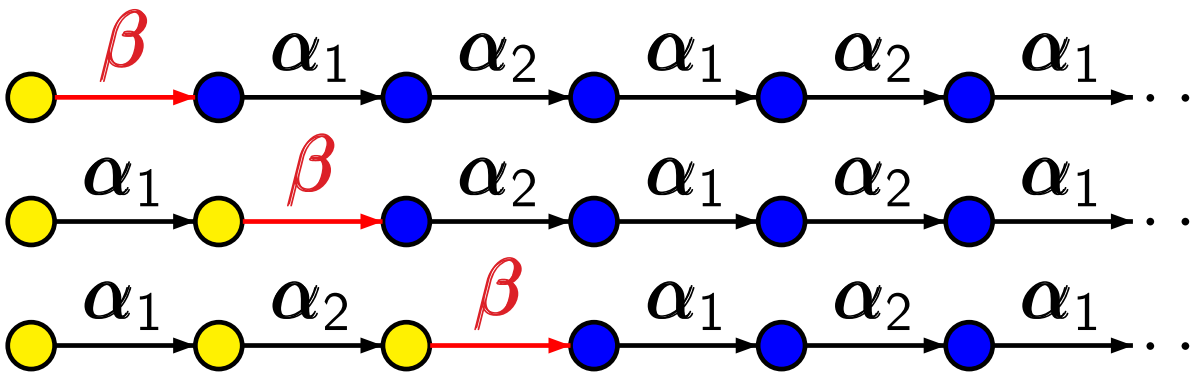




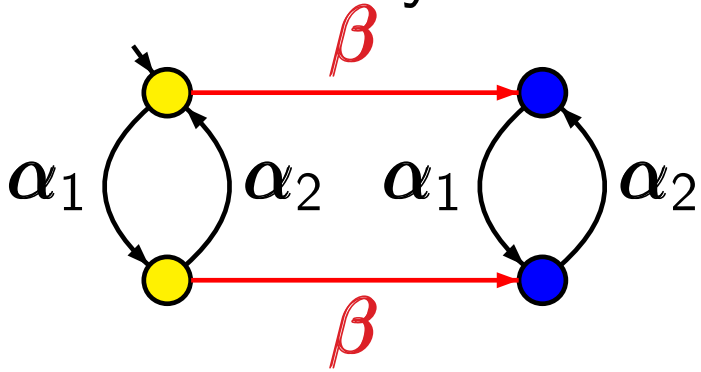
transition system $\mathcal{T}$ reduced TS $\mathcal{T}_{\text{red}}$

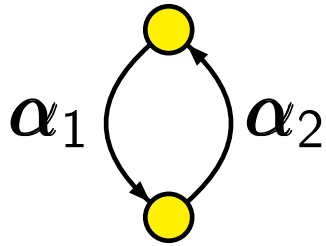


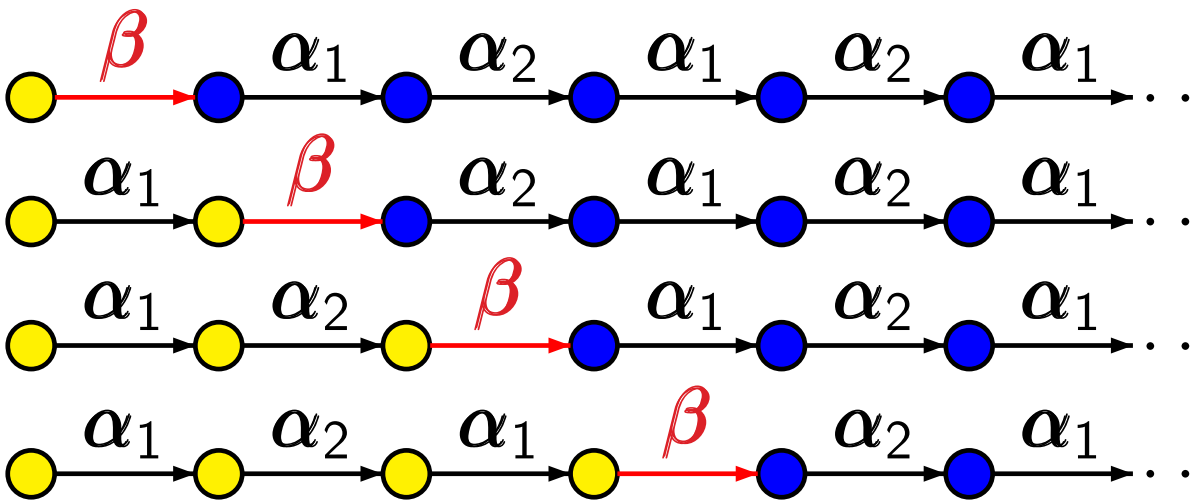




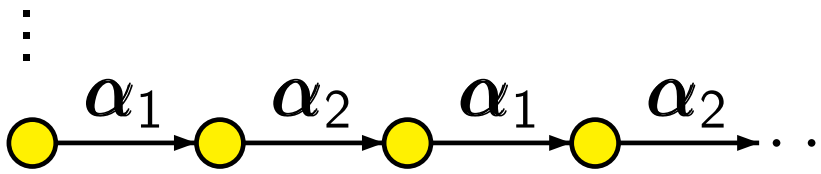
transition system $\mathcal{T}$ reduced TS $\mathcal{T}_{\text{red}}$



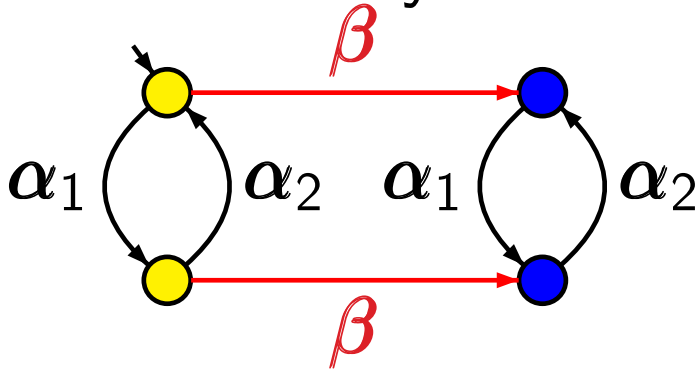


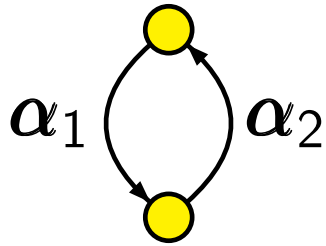


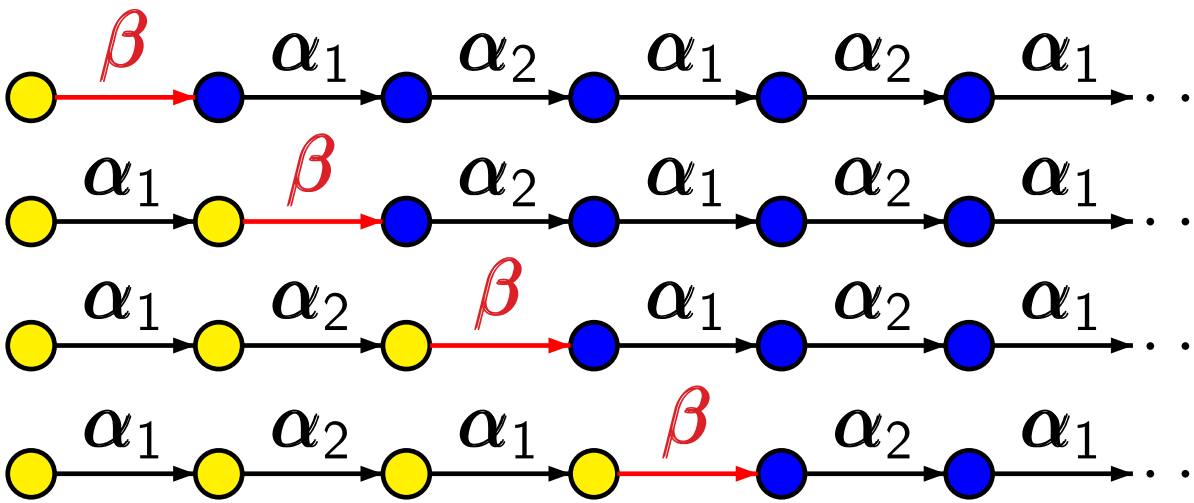
$\not\equiv$



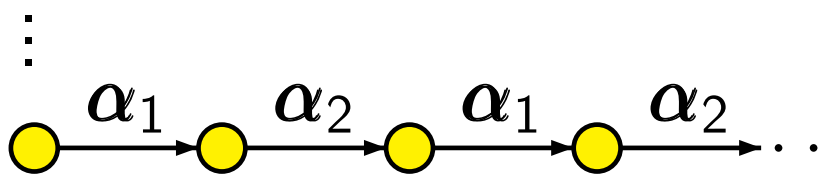
transition system $\mathcal{T}$ reduced TS $\mathcal{T}_{\text{red}}$







$\not\equiv$



$=$ the unique execution of $\mathcal{T}_{\text{red}}$

4 conditions for ample sets LTL3.4-A4

(A1) $\emptyset \neq \text{ample(s)} \subseteq \mathit{Act}(\text{s})$

(A2) for each execution fragment in $\mathcal{T}$

$$\text{s} \xrightarrow{\beta_1}\xrightarrow{\beta_2} \ldots \xrightarrow{\beta_{i-1}} \xrightarrow{\beta_i} \xrightarrow{\beta_{i+1}} \ldots \xrightarrow{\beta_{n-1}} \xrightarrow{\beta_n}$$

such that β_n is *dependent* from ample(s) there is some i $<$ n with $\beta_{\text{i}} \in$ ample(s)

(A3) if ample(s) $\neq \mathit{Act}(\text{s})$ then all actions in ample(s) are *stutter actions*

(A4) cycle condition

4 conditions for ample sets LTL3.4-A4

(A1) $\emptyset \neq \text{ample}(s) \subseteq Act(s)$

(A2) for each execution fragment in $\mathcal{T}$

$$s \xrightarrow{\beta_1}\xrightarrow{\beta_2} \ldots \xrightarrow{\beta_{i-1}} \xrightarrow{\beta_i} \xrightarrow{\beta_{i+1}} \ldots \xrightarrow{\beta_{n-1}} \xrightarrow{\beta_n}$$

such that β_n is *dependent* from $\text{ample}(s)$ there is some $i < n$ with $\beta_i \in \text{ample}(s)$

(A3) if $\text{ample}(s) \neq Act(s)$ then all actions in $\text{ample}(s)$ are *stutter actions*

(A4) for each *cycle* $s_0 \Rightarrow s_1 \Rightarrow \ldots \Rightarrow s_n$ in $\mathcal{T}_{red}$ and each action

$$\beta \in \bigcup_{0 \leq i < n} Act(s_i)$$

there is some $i \in \{1, \ldots, n\}$ with $\beta \in \text{ample}(s_i)$

4 conditions for ample sets LTL3.4-34

(A1) $\emptyset \neq \text{ample}(s) \subseteq Act(s)$

(A2) for each execution fragment in $\mathcal{T}$

$$s \xrightarrow{\beta_1}\xrightarrow{\beta_2} \ldots \xrightarrow{\beta_{i-1}} \xrightarrow{\beta_i} \xrightarrow{\beta_{i+1}} \ldots \xrightarrow{\beta_{n-1}} \xrightarrow{\beta_n}$$

such that β_n is *dependent* from $\text{ample}(s)$ there is some $i < n$ with $\beta_i \in \text{ample}(s)$

(A3) if $\text{ample}(s) \neq Act(s)$ then all actions in $\text{ample}(s)$ are *stutter actions*

(A4) for each *cycle* $s_0 \Rightarrow s_1 \Rightarrow \ldots \Rightarrow s_n$ in $\mathcal{T}_{red}$ and each action

$$\beta \in \bigcup_{0 \leq i < n} Act(s_i)$$

there is some $i \in \{1, \ldots, n\}$ with $\beta \in \text{ample}(s_i)$

Soundness of conditions $(\mathrm{A1})$, $(\mathrm{A2})$, $(\mathrm{A3})$, $(\mathrm{A4})$

Let $\mathcal{T}$ be a finite, action-deterministic transition system.

If the ample sets ample(s) satisfy conditions $(\mathrm{A1})$, $(\mathrm{A2})$, $(\mathrm{A3})$, $(\mathrm{A4})$ then

$$\mathcal{T} \stackrel{\Delta}{=} \mathcal{T}_{\mathsf{red}}$$

remind: $\stackrel{\Delta}{=}$ stutter trace equivalence

Soundness of conditions $(\mathrm{A1})$, $(\mathrm{A2})$, $(\mathrm{A3})$, $(\mathrm{A4})$

Let $\mathcal{T}$ be a finite, action-deterministic transition system.

If the ample sets ample(s) satisfy conditions $(\mathrm{A1})$, $(\mathrm{A2})$, $(\mathrm{A3})$, $(\mathrm{A4})$ then

$$\mathcal{T} \stackrel{\Delta}{=} \mathcal{T}_{\mathsf{red}}$$

hence: for all $\mathbf{LTL}_{\setminus\bigcirc}$ formulas φ:

$$\mathcal{T} \models \varphi \text{ iff } \mathcal{T}_{\mathsf{red}} \models \varphi$$