

4 conditions for ample sets LTL3.4-A4

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

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

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

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

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

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

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

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

4 conditions for ample sets LTL3.4-34

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

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

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

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

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

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

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

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

Soundness of conditions (A1), (A2), (A3), (A4)

LTL3.4-35

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

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

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

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

Soundness of conditions (A1), (A2), (A3), (A4)

LTL3.4-35

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

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

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

hence: for all $\text{LTL}_{\setminus \circ}$ formulas φ :

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

Soundness of conditions (A1), (A2), (A3), (A4)

LTL3.4-35

Let \mathcal{T} be a finite, action-deterministic transition system. If the ample sets $\text{ample}(s)$ satisfy conditions (A1), (A2), (A3), (A4) then

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

Proof: show that

$$\mathcal{T} \sqsubseteq \mathcal{T}_{\text{red}} \quad \text{and} \quad \mathcal{T}_{\text{red}} \sqsubseteq \mathcal{T}$$

where $\sqsubseteq =$ stutter trace inclusion

Soundness of conditions (A1), (A2), (A3), (A4)

LTL3.4-35

Let \mathcal{T} be a finite, action-deterministic transition system. If the ample sets $\text{ample}(s)$ satisfy conditions (A1), (A2), (A3), (A4) then

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

Proof:

- $\mathcal{T}_{\text{red}} \sqsubseteq \mathcal{T}$: \checkmark

Soundness of conditions (A1), (A2), (A3), (A4)

LTL3.4-35

Let \mathcal{T} be a finite, action-deterministic transition system. If the ample sets $\text{ample}(s)$ satisfy conditions (A1), (A2), (A3), (A4) then

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

Proof:

- $\mathcal{T}_{\text{red}} \sqsubseteq \mathcal{T}$: \checkmark

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

show that each execution ρ of \mathcal{T} can be transformed into a **stutter equivalent** execution ρ' of \mathcal{T}_{red}

Proof of $\mathcal{T} \sqsubseteq \mathcal{T}_{\text{red}}$

LTL3.4-35A

given: infinite execution fragment ρ of \mathcal{T}

goal: construction of a stutter equivalent execution fragment ρ' of \mathcal{T}_{red}

Proof of $\mathcal{T} \sqsubseteq \mathcal{T}_{\text{red}}$

LTL3.4-35A

given: infinite execution fragment ρ of \mathcal{T}

goal: construction of a stutter equivalent execution fragment ρ' of \mathcal{T}_{red}

idea: ρ' results from the “limit” of transformations

$$\rho = \rho_0 \rightsquigarrow \rho_1 \rightsquigarrow \rho_2 \rightsquigarrow \rho_3 \rightsquigarrow$$

Proof of $\mathcal{T} \sqsubseteq \mathcal{T}_{\text{red}}$

LTL3.4-35A

given: infinite execution fragment ρ of \mathcal{T}

goal: construction of a stutter equivalent execution fragment ρ' of \mathcal{T}_{red}

idea: ρ' results from the “limit” of transformations

$$\rho = \rho_0 \rightsquigarrow \rho_1 \rightsquigarrow \rho_2 \rightsquigarrow \rho_3 \rightsquigarrow$$

where, for $i > j \geq 0$, the execution fragments ρ_i and ρ_j have a common prefix

- of length j
- consisting of transitions in \mathcal{T}_{red}

Stepwise transformation $\rho_0 \rightsquigarrow \rho_1$

LTL3.4-35A

Stepwise transformation $\rho_0 \rightsquigarrow \rho_1$

LTL3.4-35A

case 0: $\rho_0 = s_0 \xrightarrow{\alpha} \xrightarrow{\beta_1} \xrightarrow{\beta_2} \xrightarrow{\beta_3} \dots$ with $\alpha \in \text{ample}(s_0)$

$$\begin{aligned} \text{case } 0: \quad \rho_0 &= s_0 \xrightarrow{\alpha} \xrightarrow{\beta_1} \xrightarrow{\beta_2} \xrightarrow{\beta_3} \dots \text{ with } \alpha \in \text{ample}(s_0) \\ \rho_1 &= s_0 \xrightarrow{\alpha} \xrightarrow{\beta_1} \xrightarrow{\beta_2} \xrightarrow{\beta_3} \dots = \rho_0 \end{aligned}$$

case 0: $\rho_0 = s_0 \xrightarrow{\alpha} \xrightarrow{\beta_1} \xrightarrow{\beta_2} \xrightarrow{\beta_3} \dots$ with $\alpha \in \text{ample}(s_0)$

$\rho_1 = s_0 \xrightarrow{\alpha} \xrightarrow{\beta_1} \xrightarrow{\beta_2} \xrightarrow{\beta_3} \dots = \rho_0$

case 1: $\rho_0 = s_0 \xrightarrow{\beta_1} \xrightarrow{\beta_2} \dots \xrightarrow{\beta_{n-1}} \xrightarrow{\alpha} \xrightarrow{\beta_{n+1}} \xrightarrow{\beta_{n+2}} \dots$

where $\beta_1, \dots, \beta_{n-1} \notin \text{ample}(s_0)$, $\alpha \in \text{ample}(s_0)$

case 0: $\rho_0 = s_0 \xrightarrow{\alpha} \xrightarrow{\beta_1} \xrightarrow{\beta_2} \xrightarrow{\beta_3} \dots$ with $\alpha \in \text{ample}(s_0)$

$$\rho_1 = s_0 \xrightarrow{\alpha} \xrightarrow{\beta_1} \xrightarrow{\beta_2} \xrightarrow{\beta_3} \dots = \rho_0$$

case 1: $\rho_0 = s_0 \xrightarrow{\beta_1} \xrightarrow{\beta_2} \dots \xrightarrow{\beta_{n-1}} \xrightarrow{\alpha} \xrightarrow{\beta_{n+1}} \xrightarrow{\beta_{n+2}} \dots$

where $\beta_1, \dots, \beta_{n-1} \notin \text{ample}(s_0)$, $\alpha \in \text{ample}(s_0)$

$$\rho_1 = s_0 \xrightarrow{\alpha} \xrightarrow{\beta_1} \dots \xrightarrow{\beta_{n-2}} \xrightarrow{\beta_{n-1}} \xrightarrow{\beta_{n+1}} \xrightarrow{\beta_{n+2}} \dots$$

case 0: $\rho_0 = s_0 \xrightarrow{\alpha} \xrightarrow{\beta_1} \xrightarrow{\beta_2} \xrightarrow{\beta_3} \dots$ with $\alpha \in \text{ample}(s_0)$

$$\rho_1 = s_0 \xrightarrow{\alpha} \xrightarrow{\beta_1} \xrightarrow{\beta_2} \xrightarrow{\beta_3} \dots = \rho_0$$

case 1: $\rho_0 = s_0 \xrightarrow{\beta_1} \xrightarrow{\beta_2} \dots \xrightarrow{\beta_{n-1}} \xrightarrow{\alpha} \xrightarrow{\beta_{n+1}} \xrightarrow{\beta_{n+2}} \dots$

where $\beta_1, \dots, \beta_{n-1} \notin \text{ample}(s_0)$, $\alpha \in \text{ample}(s_0)$

$$\rho_1 = s_0 \xrightarrow{\alpha} \xrightarrow{\beta_1} \dots \xrightarrow{\beta_{n-2}} \xrightarrow{\beta_{n-1}} \xrightarrow{\beta_{n+1}} \xrightarrow{\beta_{n+2}} \dots$$

case 2: $\rho_0 = s_0 \xrightarrow{\beta_1} \xrightarrow{\beta_2} \xrightarrow{\beta_3} \dots$

where $\beta_i \notin \text{ample}(s_0)$, $i = 1, 2, \dots$

case 0: $\rho_0 = s_0 \xrightarrow{\alpha} \xrightarrow{\beta_1} \xrightarrow{\beta_2} \xrightarrow{\beta_3} \dots$ with $\alpha \in \text{ample}(s_0)$

$$\rho_1 = s_0 \xrightarrow{\alpha} \xrightarrow{\beta_1} \xrightarrow{\beta_2} \xrightarrow{\beta_3} \dots = \rho_0$$

case 1: $\rho_0 = s_0 \xrightarrow{\beta_1} \xrightarrow{\beta_2} \dots \xrightarrow{\beta_{n-1}} \xrightarrow{\alpha} \xrightarrow{\beta_{n+1}} \xrightarrow{\beta_{n+2}} \dots$

where $\beta_1, \dots, \beta_{n-1} \notin \text{ample}(s_0)$, $\alpha \in \text{ample}(s_0)$

$$\rho_1 = s_0 \xrightarrow{\alpha} \xrightarrow{\beta_1} \dots \xrightarrow{\beta_{n-2}} \xrightarrow{\beta_{n-1}} \xrightarrow{\beta_{n+1}} \xrightarrow{\beta_{n+2}} \dots$$

case 2: $\rho_0 = s_0 \xrightarrow{\beta_1} \xrightarrow{\beta_2} \xrightarrow{\beta_3} \dots$

where $\beta_i \notin \text{ample}(s_0)$, $i = 1, 2, \dots$

$$\rho_1 = s_0 \xrightarrow{\alpha} \xrightarrow{\beta_1} \xrightarrow{\beta_2} \xrightarrow{\beta_3} \dots$$

case 0: $\rho_0 = s_0 \xrightarrow{\alpha} \xrightarrow{\beta_1} \xrightarrow{\beta_2} \xrightarrow{\beta_3} \dots$ with $\alpha \in \text{ample}(s_0)$

$\rho_1 = s_0 \xrightarrow{\alpha} \xrightarrow{\beta_1} \xrightarrow{\beta_2} \xrightarrow{\beta_3} \dots = \rho_0$

case 1: $\rho_0 = s_0 \xrightarrow{\beta_1} \xrightarrow{\beta_2} \dots \xrightarrow{\beta_{n-1}} \xrightarrow{\alpha} \xrightarrow{\beta_{n+1}} \xrightarrow{\beta_{n+2}} \dots$

where $\beta_1, \dots, \beta_{n-1} \notin \text{ample}(s_0)$, $\alpha \in \text{ample}(s_0)$

$\rho_1 = s_0 \xrightarrow{\alpha} \xrightarrow{\beta_1} \dots \xrightarrow{\beta_{n-2}} \xrightarrow{\beta_{n-1}} \xrightarrow{\beta_{n+1}} \xrightarrow{\beta_{n+2}} \dots$

case 2: $\rho_0 = s_0 \xrightarrow{\beta_1} \xrightarrow{\beta_2} \xrightarrow{\beta_3} \dots$

where $\beta_i \notin \text{ample}(s_0)$, $i = 1, 2, \dots$

$\rho_1 = s_0 \xrightarrow{\alpha} \xrightarrow{\beta_1} \xrightarrow{\beta_2} \xrightarrow{\beta_3} \dots$

by (A3): α is a stutter action in cases 1 and 2

case 0: $\rho_0 = s_0 \xrightarrow{\alpha} \xrightarrow{\beta_1} \xrightarrow{\beta_2} \xrightarrow{\beta_3} \dots$ with $\alpha \in \text{ample}(s_0)$

$$\rho_1 = s_0 \xrightarrow{\alpha} \xrightarrow{\beta_1} \xrightarrow{\beta_2} \xrightarrow{\beta_3} \dots = \rho_0$$

case 1: $\rho_0 = s_0 \xrightarrow{\beta_1} \xrightarrow{\beta_2} \dots \xrightarrow{\beta_{n-1}} \xrightarrow{\alpha} \xrightarrow{\beta_{n+1}} \xrightarrow{\beta_{n+2}} \dots$

where $\beta_1, \dots, \beta_{n-1} \notin \text{ample}(s_0)$, $\alpha \in \text{ample}(s_0)$

$$\rho_1 = s_0 \xrightarrow{\alpha} \xrightarrow{\beta_1} \dots \xrightarrow{\beta_{n-2}} \xrightarrow{\beta_{n-1}} \xrightarrow{\beta_{n+1}} \xrightarrow{\beta_{n+2}} \dots \stackrel{\Delta}{=} \rho_0$$

case 2: $\rho_0 = s_0 \xrightarrow{\beta_1} \xrightarrow{\beta_2} \xrightarrow{\beta_3} \dots$

where $\beta_i \notin \text{ample}(s_0)$, $i = 1, 2, \dots$

$$\rho_1 = s_0 \xrightarrow{\alpha} \xrightarrow{\beta_1} \xrightarrow{\beta_2} \xrightarrow{\beta_3} \dots \stackrel{\Delta}{=} \rho_0$$

by (A3): α is a stutter action in cases 1 and 2

case 0: $\rho_0 = s_0 \xrightarrow{\alpha} \xrightarrow{\beta_1} \xrightarrow{\beta_2} \xrightarrow{\beta_3} \dots$ with $\alpha \in \text{ample}(s_0)$

$$\rho_1 = s_0 \xrightarrow{\alpha} \xrightarrow{\beta_1} \xrightarrow{\beta_2} \xrightarrow{\beta_3} \dots = \rho_0$$

case 1: $\rho_0 = s_0 \xrightarrow{\beta_1} \xrightarrow{\beta_2} \dots \xrightarrow{\beta_{n-1}} \xrightarrow{\alpha} \xrightarrow{\beta_{n+1}} \xrightarrow{\beta_{n+2}} \dots$

where $\beta_1, \dots, \beta_{n-1} \notin \text{ample}(s_0)$, $\alpha \in \text{ample}(s_0)$

$$\rho_1 = s_0 \xrightarrow{\alpha} \xrightarrow{\beta_1} \dots \xrightarrow{\beta_{n-2}} \xrightarrow{\beta_{n-1}} \xrightarrow{\beta_{n+1}} \xrightarrow{\beta_{n+2}} \dots \stackrel{\Delta}{=} \rho_0$$

case 2: $\rho_0 = s_0 \xrightarrow{\beta_1} \xrightarrow{\beta_2} \xrightarrow{\beta_3} \dots$

where $\beta_i \notin \text{ample}(s_0)$, $i = 1, 2, \dots$

$$\rho_1 = s_0 \xrightarrow{\alpha} \xrightarrow{\beta_1} \xrightarrow{\beta_2} \xrightarrow{\beta_3} \dots \stackrel{\Delta}{=} \rho_0$$

$\rho_1 \rightsquigarrow \rho_2$:

repeat the same procedure from the 2nd state on

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

LTL3.4-21

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}_{red}

can be constructed by successively

- permutating the order independent actions
- adding independent stutter actions

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

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

by successively applying the following transformations:

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

case 0: $\rho = s_0 \xrightarrow{\alpha} s'_0 \rightarrow \dots$ with $\alpha \in \text{ample}(s_0)$

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

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

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

case 0: $\rho = s_0 \xrightarrow{\alpha} s'_0 \rightarrow \dots$ with $\alpha \in \text{ample}(s_0)$
 $s_0 \xRightarrow{\alpha} s'_0 \rightarrow \dots$

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

case 2: $\rho = s_0 \xrightarrow{\beta_1} \xrightarrow{\beta_2} \xrightarrow{\beta_3} \dots$ with $\beta_i \notin \text{ample}(s_0)$
 $s_0 \xRightarrow{\alpha} \xrightarrow{\beta_1} \xrightarrow{\beta_2} \xrightarrow{\beta_3} \dots$ for some $\alpha \in \text{ample}(s_0)$

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

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

$$\rho \rightsquigarrow \rho_1 \rightsquigarrow \rho_2 \rightsquigarrow \rho_3 \rightsquigarrow \dots$$

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}_{red} , i.e., ρ_i has the form

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

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

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

$$\rho \rightsquigarrow \rho_1 \rightsquigarrow \rho_2 \rightsquigarrow \rho_3 \rightsquigarrow \dots$$

where

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

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

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

case 0: $\rho = s_0 \xrightarrow{\alpha} s'_0 \rightarrow \dots$ with $\alpha \in \text{ample}(s_0)$

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

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

case 0: $\rho = s_0 \xrightarrow{\alpha} s'_0 \rightarrow \dots$ with $\alpha \in \text{ample}(s_0)$

$$\rho_1 = s_0 \xrightarrow{\alpha} s'_0 \rightarrow \dots$$

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

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

case 0: $\rho = s_0 \xrightarrow{\alpha} s'_0 \rightarrow \dots$ with $\alpha \in \text{ample}(s_0)$

$\rho_1 = s_0 \xrightarrow{\alpha} s'_0 \rightarrow \dots$

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

$\rho_1 = s_0 \xrightarrow{\alpha} s'_0 \xrightarrow{\beta_1} \dots \xrightarrow{\beta_n} \rightarrow \dots$

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

case 0: $\rho = s_0 \xrightarrow{\alpha} s'_0 \rightarrow \dots$ with $\alpha \in \text{ample}(s_0)$

$\rho_1 = s_0 \xrightarrow{\alpha} s'_0 \rightarrow \dots$

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

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

$\rho_1 = s_0 \xrightarrow{\alpha} s'_0 \xrightarrow{\beta_1} \dots \xrightarrow{\beta_n} \rightarrow \dots$

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

$\rho_1 = s_0 \xrightarrow{\alpha} s'_0 \xrightarrow{\beta_1} \xrightarrow{\beta_2} \xrightarrow{\beta_3} \dots$ for some $\alpha \in \text{ample}(s_0)$

case 0: $\rho = s_0 \xrightarrow{\alpha} s'_0 \rightarrow \dots$ with $\alpha \in \text{ample}(s_0)$

$$\rho_1 = s_0 \xrightarrow{\alpha} s'_0 \rightarrow \dots$$

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

$$\rho_1 = s_0 \xrightarrow{\alpha} s'_0 \xrightarrow{\beta_1} \dots \xrightarrow{\beta_n} \rightarrow \dots$$

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

$$\rho_1 = s_0 \xrightarrow{\alpha} s'_0 \xrightarrow{\beta_1} \xrightarrow{\beta_2} \xrightarrow{\beta_3} \dots \text{ for some } \alpha \in \text{ample}(s_0)$$

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

apply case 0,1 or 2 to the suffix starting in state s'_0

Transformation according to cases 1 and 2

LTL3.4-36

$$\begin{array}{l} \rho_0 = \quad s_0 \xrightarrow{\beta_1} \dots \quad \xrightarrow{\beta_{k-1}} \xrightarrow{\beta_k} \xrightarrow{\beta_{k+1}} \dots \\ \rho_1 = \quad s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\beta_1} \dots \quad \xrightarrow{\beta_{k-1}} \xrightarrow{\beta_k} \xrightarrow{\beta_{k+1}} \dots \\ \rho_2 = \quad s_0 \xrightarrow{\alpha_1} \xrightarrow{\alpha_2} s_2 \xrightarrow{\beta_1} \dots \quad \xrightarrow{\beta_{k-1}} \xrightarrow{\beta_k} \xrightarrow{\beta_{k+1}} \dots \\ \vdots \\ \rho_m = \quad s_0 \xrightarrow{\alpha_1} \xrightarrow{\alpha_2} \dots \xrightarrow{\alpha_m} s_m \xrightarrow{\beta_1} \dots \quad \xrightarrow{\beta_k} \xrightarrow{\beta_{k+1}} \dots \end{array}$$

Transformation according to cases 1 and 2

LTL3.4-36

$$\begin{array}{l} \rho_0 = \quad s_0 \xrightarrow{\beta_1} \dots \quad \quad \quad \xrightarrow{\beta_{k-1}} \xrightarrow{\beta_k} \xrightarrow{\beta_{k+1}} \dots \\ \rho_1 = \quad s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\beta_1} \dots \quad \quad \quad \xrightarrow{\beta_{k-1}} \xrightarrow{\beta_k} \xrightarrow{\beta_{k+1}} \dots \\ \rho_2 = \quad s_0 \xrightarrow{\alpha_1} \xrightarrow{\alpha_2} s_2 \xrightarrow{\beta_1} \dots \quad \quad \quad \xrightarrow{\beta_{k-1}} \xrightarrow{\beta_k} \xrightarrow{\beta_{k+1}} \dots \\ \vdots \\ \rho_m = \quad s_0 \xrightarrow{\alpha_1} \xrightarrow{\alpha_2} \dots \xrightarrow{\alpha_m} s_m \xrightarrow{\beta_1} \dots \quad \quad \quad \xrightarrow{\beta_k} \xrightarrow{\beta_{k+1}} \dots \end{array}$$

α_j stutter action

Transformation according to cases 1 and 2

LTL3.4-36

$$\begin{array}{l} \rho_0 = \quad s_0 \xrightarrow{\beta_1} \dots \quad \xrightarrow{\beta_{k-1}} \xrightarrow{\beta_k} \xrightarrow{\beta_{k+1}} \dots \\ \rho_1 = \quad s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\beta_1} \dots \quad \xrightarrow{\beta_{k-1}} \xrightarrow{\beta_k} \xrightarrow{\beta_{k+1}} \dots \\ \rho_2 = \quad s_0 \xrightarrow{\alpha_1} \xrightarrow{\alpha_2} s_2 \xrightarrow{\beta_1} \dots \quad \xrightarrow{\beta_{k-1}} \xrightarrow{\beta_k} \xrightarrow{\beta_{k+1}} \dots \\ \vdots \\ \rho_m = \quad s_0 \xrightarrow{\alpha_1} \xrightarrow{\alpha_2} \dots \xrightarrow{\alpha_m} s_m \xrightarrow{\beta_1} \dots \quad \xrightarrow{\beta_k} \xrightarrow{\beta_{k+1}} \dots \end{array}$$

α_j stutter action $\rightsquigarrow \rho_0 \stackrel{\Delta}{=} \rho_1 \stackrel{\Delta}{=} \rho_2 \stackrel{\Delta}{=} \dots$

Transformation according to cases 1 and 2

LTL3.4-36

$$\begin{array}{l} \rho_0 = \quad s_0 \xrightarrow{\beta_1} \dots \quad \xrightarrow{\beta_{k-1}} \xrightarrow{\beta_k} \xrightarrow{\beta_{k+1}} \dots \\ \rho_1 = \quad s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\beta_1} \dots \quad \xrightarrow{\beta_{k-1}} \xrightarrow{\beta_k} \xrightarrow{\beta_{k+1}} \dots \\ \rho_2 = \quad s_0 \xrightarrow{\alpha_1} \xrightarrow{\alpha_2} s_2 \xrightarrow{\beta_1} \dots \quad \xrightarrow{\beta_{k-1}} \xrightarrow{\beta_k} \xrightarrow{\beta_{k+1}} \dots \\ \vdots \\ \rho_m = \quad s_0 \xrightarrow{\alpha_1} \xrightarrow{\alpha_2} \dots \xrightarrow{\alpha_m} s_m \xrightarrow{\beta_1} \dots \quad \xrightarrow{\beta_k} \xrightarrow{\beta_{k+1}} \dots \end{array}$$

by the cycle condition (A4):

“action β_1 will *not* be postponed forever”

Transformation according to cases 1 and 2

$$\begin{array}{l}
 \rho_0 = \quad s_0 \xrightarrow{\beta_1} \dots \quad \xrightarrow{\beta_{k-1}} \xrightarrow{\beta_k} \xrightarrow{\beta_{k+1}} \dots \\
 \rho_1 = \quad s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\beta_1} \dots \quad \xrightarrow{\beta_{k-1}} \xrightarrow{\beta_k} \xrightarrow{\beta_{k+1}} \dots \\
 \rho_2 = \quad s_0 \xrightarrow{\alpha_1} \xrightarrow{\alpha_2} s_2 \xrightarrow{\beta_1} \dots \quad \xrightarrow{\beta_{k-1}} \xrightarrow{\beta_k} \xrightarrow{\beta_{k+1}} \dots \\
 \vdots \\
 \rho_m = \quad s_0 \xrightarrow{\alpha_1} \xrightarrow{\alpha_2} \dots \xrightarrow{\alpha_m} s_m \xrightarrow{\beta_1} \dots \quad \xrightarrow{\beta_k} \xrightarrow{\beta_{k+1}} \dots
 \end{array}$$

by the cycle condition (A4):

“action β_1 will *not* be postponed forever”

i.e., there exists some m such that case 0 applies
and $\rho_m = \rho_{m+1}$

Transformation according to cases 1 and 2

$$\begin{array}{l}
 \rho_0 = \quad s_0 \xrightarrow{\beta_1} \dots \quad \xrightarrow{\beta_{k-1}} \xrightarrow{\beta_k} \xrightarrow{\beta_{k+1}} \dots \\
 \rho_1 = \quad s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\beta_1} \dots \quad \xrightarrow{\beta_{k-1}} \xrightarrow{\beta_k} \xrightarrow{\beta_{k+1}} \dots \\
 \rho_2 = \quad s_0 \xrightarrow{\alpha_1} \xrightarrow{\alpha_2} s_2 \xrightarrow{\beta_1} \dots \quad \xrightarrow{\beta_{k-1}} \xrightarrow{\beta_k} \xrightarrow{\beta_{k+1}} \dots \\
 \vdots \\
 \rho_m = \quad s_0 \xrightarrow{\alpha_1} \xrightarrow{\alpha_2} \dots \xrightarrow{\alpha_m} s_m \xrightarrow{\beta_1} \dots \xrightarrow{\beta_k} \xrightarrow{\beta_{k+1}} \dots
 \end{array}$$

by the cycle condition (A4):

“action β_1 will *not* be postponed forever”

i.e., there exists some m such that case 0 applies
and $\rho_m = \rho_{m+1}$

Transformation according to cases 1 and 2

$$\begin{array}{l}
 \rho_0 = \quad S_0 \xrightarrow{\beta_1} \dots \quad \xrightarrow{\beta_{k-1}} \xrightarrow{\beta_k} \xrightarrow{\beta_{k+1}} \dots \\
 \rho_1 = \quad S_0 \xrightarrow{\alpha_1} S_1 \xrightarrow{\beta_1} \dots \quad \xrightarrow{\beta_{k-1}} \xrightarrow{\beta_k} \xrightarrow{\beta_{k+1}} \dots \\
 \rho_2 = \quad S_0 \xrightarrow{\alpha_1} \xrightarrow{\alpha_2} S_2 \xrightarrow{\beta_1} \dots \quad \xrightarrow{\beta_{k-1}} \xrightarrow{\beta_k} \xrightarrow{\beta_{k+1}} \dots \\
 \vdots \\
 \rho_m = \quad S_0 \xrightarrow{\alpha_1} \xrightarrow{\alpha_2} \dots \xrightarrow{\alpha_m} S_m \xrightarrow{\beta_1} \dots \quad \xrightarrow{\beta_k} \xrightarrow{\beta_{k+1}} \dots \\
 \rho_{m+1} = \quad S_0 \xrightarrow{\alpha_1} \xrightarrow{\alpha_2} \dots \xrightarrow{\alpha_m} S_m \xrightarrow{\beta_1} \dots \quad \xrightarrow{\beta_k} \xrightarrow{\beta_{k+1}} \dots
 \end{array}$$

by the cycle condition (A4):

“action β_1 will *not* be postponed forever”

i.e., there exists some m such that case 0 applies
and $\rho_m = \rho_{m+1}$

4 conditions for ample sets LTL3.4-FOUR-COND

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

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

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

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

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

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

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

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

The ample set method for $LTL_{\setminus O}$ model checking

LTL3.4-37

- *on-the-fly* DFS-based generation of \mathcal{T}_{red}
- *exploration* of state s :
create the states $\alpha(s)$ for $\alpha \in \text{ample}(s)$, but ignore the β -successors of s for $\beta \notin \text{ample}(s)$
- *interleave* the generation of \mathcal{T}_{red} with the product construction $\mathcal{T}_{red} \otimes \mathcal{A}$ and *nested DFS*

where \mathcal{A} is an NBA for the negation of the formula to be checked

here: only explanations for *reachability analysis*

The ample set method for reachability

LTL3.4-37

given: finite transition system \mathcal{T}
atomic proposition \mathbf{a}

goal: on-the-fly construction of \mathcal{T}_{red}
abort as soon as a state s with
 $s \not\models \mathbf{a}$ has been generated

The ample set method for reachability

LTL3.4-37

given: finite transition system \mathcal{T}
atomic proposition \mathbf{a}

goal: on-the-fly construction of \mathcal{T}_{red}
abort as soon as a state s with
 $s \not\models \mathbf{a}$ has been generated

uses

- \mathbf{V} = set of states that have been generated so far (organized as a hash table)

The ample set method for reachability

LTL3.4-37

given: finite transition system \mathcal{T}
atomic proposition \mathbf{a}

goal: on-the-fly construction of \mathcal{T}_{red}
abort as soon as a state s with
 $s \not\models \mathbf{a}$ has been generated

uses

- \mathbf{V} = set of states that have been generated so far (organized as a hash table)
- DFS-stack π

The ample set method for reachability

LTL3.4-37

given: finite transition system \mathcal{T} for $P_1 \parallel \dots \parallel P_n$
atomic proposition \mathbf{a}

goal: on-the-fly construction of \mathcal{T}_{red}
abort as soon as a state \mathbf{s} with
 $\mathbf{s} \not\models \mathbf{a}$ has been generated

uses

- \mathbf{V} = set of states that have been generated so far (organized as a hash table)
- DFS-stack π
- “local” criteria to compute $\text{ample}(\mathbf{s})$ from a syntactic representation of the processes P_i

Ample set method (full generation of \mathcal{T}_{red})

LTL3.4-37

$\pi := \emptyset; \mathbf{V} := \emptyset$

WHILE $\mathbf{S}_0 \not\subseteq \mathbf{V}$ DO

select an initial state $\mathbf{s} \in \mathbf{S}_0 \setminus \mathbf{V}$; add \mathbf{s} to \mathbf{V} ;

Push(π, \mathbf{s});

OD

Ample set method (full generation of \mathcal{T}_{red})

LTL3.4-37

$\pi := \emptyset; \mathbf{V} := \emptyset$

WHILE $\mathbf{S}_0 \not\subseteq \mathbf{V}$ DO

select an initial state $\mathbf{s} \in \mathbf{S}_0 \setminus \mathbf{V}$; add \mathbf{s} to \mathbf{V} ;

Push(π, \mathbf{s}); compute ample(\mathbf{s});

OD

Ample set method (full generation of \mathcal{T}_{red})

LTL3.4-37

$\pi := \emptyset; \mathbf{V} := \emptyset$

WHILE $\mathbf{S}_0 \not\subseteq \mathbf{V}$ DO

select an initial state $\mathbf{s} \in \mathbf{S}_0 \setminus \mathbf{V}$; add \mathbf{s} to \mathbf{V} ;

Push(π, \mathbf{s}); compute ample(\mathbf{s});

WHILE $\pi \neq \emptyset$ DO

$\mathbf{s} := \text{Top}(\pi)$;

OD

OD

Ample set method (full generation of \mathcal{T}_{red})

LTL3.4-37

$\pi := \emptyset; \mathbf{V} := \emptyset$

WHILE $\mathbf{S}_0 \not\subseteq \mathbf{V}$ **DO**

select an initial state $\mathbf{s} \in \mathbf{S}_0 \setminus \mathbf{V}$; add \mathbf{s} to \mathbf{V} ;

Push(π, \mathbf{s}); compute ample(\mathbf{s});

WHILE $\pi \neq \emptyset$ **DO**

$\mathbf{s} := \text{Top}(\pi)$;

IF $\exists \alpha \in \text{ample}(\mathbf{s})$ with $\alpha(\mathbf{s}) \notin \mathbf{V}$

FI

OD

OD

Ample set method (full generation of \mathcal{T}_{red})

LTL3.4-37

$\pi := \emptyset; \mathbf{V} := \emptyset$

WHILE $\mathbf{S}_0 \not\subseteq \mathbf{V}$ **DO**

select an initial state $\mathbf{s} \in \mathbf{S}_0 \setminus \mathbf{V}$; add \mathbf{s} to \mathbf{V} ;

Push(π, \mathbf{s}); compute ample(\mathbf{s});

WHILE $\pi \neq \emptyset$ **DO**

$\mathbf{s} := \text{Top}(\pi)$;

IF $\exists \alpha \in \text{ample}(\mathbf{s})$ with $\alpha(\mathbf{s}) \notin \mathbf{V}$

THEN select such α ; add $\mathbf{s}' := \alpha(\mathbf{s})$ to \mathbf{V} ;

Push(π, \mathbf{s}');

FI

OD

OD

Ample set method (full generation of \mathcal{T}_{red})

LTL3.4-37

$\pi := \emptyset; \mathbf{V} := \emptyset$

WHILE $\mathbf{S}_0 \not\subseteq \mathbf{V}$ **DO**

select an initial state $\mathbf{s} \in \mathbf{S}_0 \setminus \mathbf{V}$; add \mathbf{s} to \mathbf{V} ;

Push(π, \mathbf{s}); compute ample(\mathbf{s});

WHILE $\pi \neq \emptyset$ **DO**

$\mathbf{s} := \text{Top}(\pi)$;

IF $\exists \alpha \in \text{ample}(\mathbf{s})$ with $\alpha(\mathbf{s}) \notin \mathbf{V}$

THEN select such α ; add $\mathbf{s}' := \alpha(\mathbf{s})$ to \mathbf{V} ;

Push(π, \mathbf{s}'); compute ample(\mathbf{s}');

FI

OD

OD

Ample set method (full generation of \mathcal{T}_{red})

LTL3.4-37

$\pi := \emptyset; \mathbf{V} := \emptyset$

WHILE $\mathbf{S}_0 \not\subseteq \mathbf{V}$ **DO**

select an initial state $\mathbf{s} \in \mathbf{S}_0 \setminus \mathbf{V}$; add \mathbf{s} to \mathbf{V} ;

Push(π, \mathbf{s}); compute ample(\mathbf{s});

WHILE $\pi \neq \emptyset$ **DO**

$\mathbf{s} := \text{Top}(\pi)$;

IF $\exists \alpha \in \text{ample}(\mathbf{s})$ with $\alpha(\mathbf{s}) \notin \mathbf{V}$

THEN select such α ; add $\mathbf{s}' := \alpha(\mathbf{s})$ to \mathbf{V} ;

Push(π, \mathbf{s}'); compute ample(\mathbf{s}');

ELSE Pop(π)

FI

OD

OD

$\pi := \emptyset; \mathbf{V} := \emptyset$

WHILE $\mathbf{S}_0 \not\subseteq \mathbf{V}$ **DO**

select an initial state $\mathbf{s} \in \mathbf{S}_0 \setminus \mathbf{V}$; add \mathbf{s} to \mathbf{V} ;

Push(π, \mathbf{s}); compute ample(\mathbf{s});

WHILE $\pi \neq \emptyset$ **DO**

$\mathbf{s} := \text{Top}(\pi)$;

IF $\exists \alpha \in \text{ample}(\mathbf{s})$ with $\alpha(\mathbf{s}) \notin \mathbf{V}$

THEN select such α ; add $\mathbf{s}' := \alpha(\mathbf{s})$ to \mathbf{V} ;

Push(π, \mathbf{s}'); compute ample(\mathbf{s}');

ELSE Pop(π)

FI

OD

OD

Does $\mathcal{T} \models \Box a$ hold?

LTL3.4-37

$\pi := \emptyset; \mathbf{V} := \emptyset$

WHILE $\mathbf{S}_0 \not\subseteq \mathbf{V}$ **DO**

select an initial state $\mathbf{s} \in \mathbf{S}_0 \setminus \mathbf{V}$; add \mathbf{s} to \mathbf{V} ;

Push(π, \mathbf{s}); compute ample(\mathbf{s});

WHILE $\pi \neq \emptyset$ **DO**

$\mathbf{s} := \text{Top}(\pi)$;

IF $\exists \alpha \in \text{ample}(\mathbf{s})$ with $\alpha(\mathbf{s}) \notin \mathbf{V}$

THEN select such α ; add $\mathbf{s}' := \alpha(\mathbf{s})$ to \mathbf{V} ;

Push(π, \mathbf{s}'); compute ample(\mathbf{s}');

ELSE Pop(π)

FI

OD

OD

Does $\mathcal{T} \models \Box a$ hold?

LTL3.4-37

$\pi := \emptyset; \mathbf{V} := \emptyset$

WHILE $\mathbf{S}_0 \not\subseteq \mathbf{V}$ **DO**

select an initial state $\mathbf{s} \in \mathbf{S}_0 \setminus \mathbf{V}$; add \mathbf{s} to \mathbf{V} ;

Push(π, \mathbf{s}); compute ample(\mathbf{s});

WHILE $\pi \neq \emptyset$ **DO**

$\mathbf{s} := \text{Top}(\pi)$;

IF $\mathbf{s} \not\models a$ **THEN** return “NO” + counterexample **FI**;

IF $\exists \alpha \in \text{ample}(\mathbf{s})$ with $\alpha(\mathbf{s}) \notin \mathbf{V}$

THEN

ELSE Pop(π)

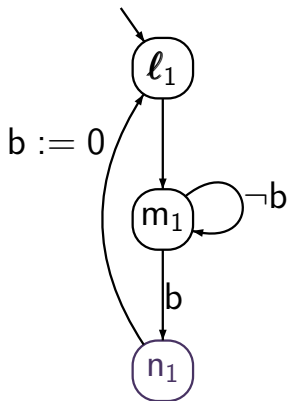
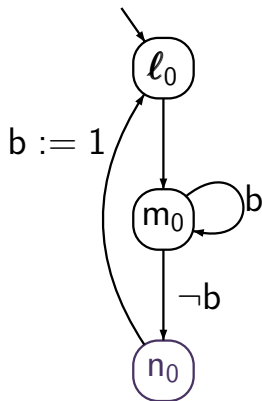
FI

OD

OD

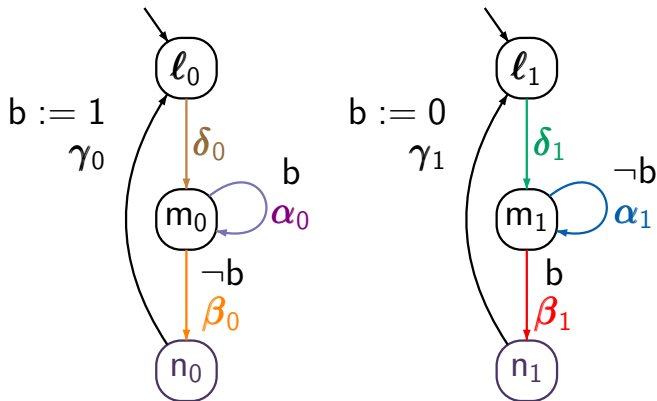
full generation of \mathcal{T}_{red} for $\mathcal{T} = \mathcal{T}_{P_1} \parallel \mathcal{T}_{P_2}$ where

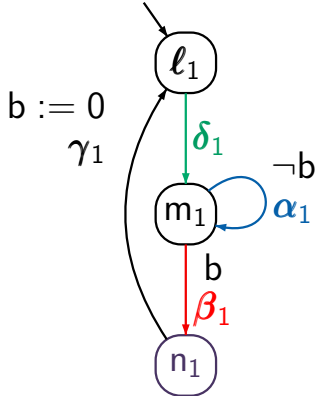
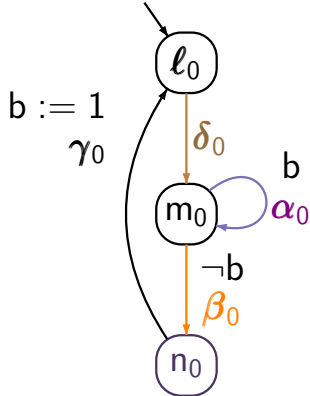
- P_1, P_2 are program graphs with shared variable $b \in \{0, 1\}$
- $AP = \{n_0, n_1\}$



full generation of \mathcal{T}_{red} for $\mathcal{T} = \mathcal{T}_{P_1} \parallel P_2$ where

- P_1, P_2 are program graphs with shared variable $b \in \{0, 1\}$
- $AP = \{n_0, n_1\}$





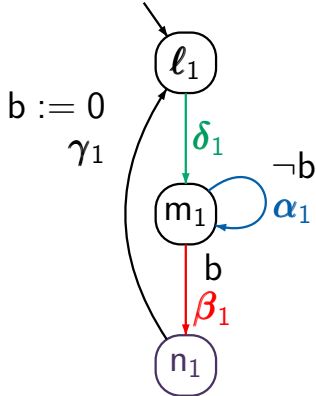
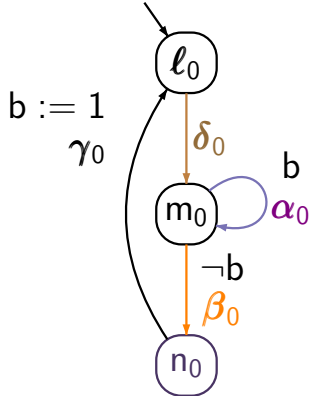
independent actions:

$\delta_0 \delta_1$

$\delta_0 \alpha_1$

$\delta_0 \beta_1$

$\delta_0 \gamma_1$



independent actions:

$\delta_0 \delta_1$

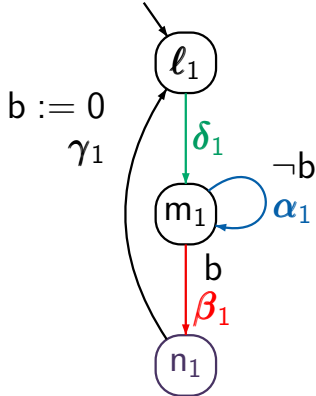
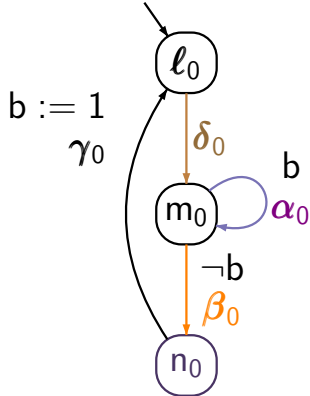
$\delta_0 \alpha_1$

$\delta_0 \beta_1$

$\delta_0 \gamma_1$

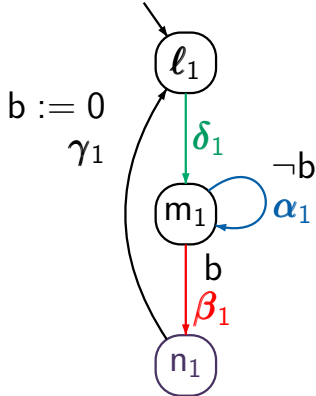
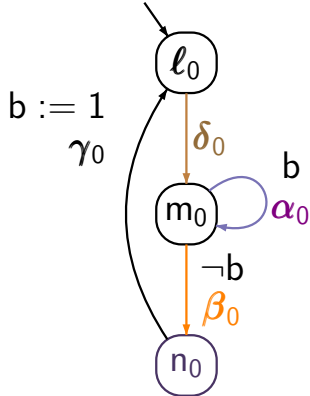
$\alpha_0 \delta_1$

$\alpha_0 \beta_1$



independent actions:

$\delta_0 \delta_1$	$\delta_0 \alpha_1$	$\delta_0 \beta_1$	$\delta_0 \gamma_1$
$\alpha_0 \delta_1$		$\alpha_0 \beta_1$	
$\beta_0 \delta_1$	$\beta_0 \alpha_1$	$\beta_0 \beta_1$	$\beta_0 \gamma_1$



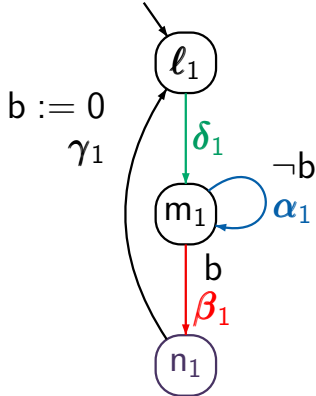
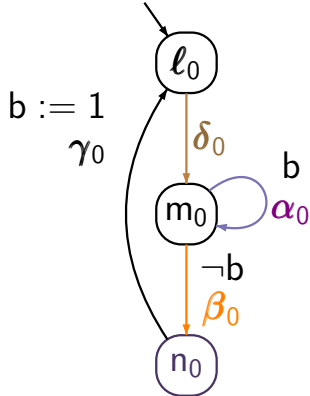
independent actions:

$\delta_0 \delta_1$ $\delta_0 \alpha_1$ $\delta_0 \beta_1$ $\delta_0 \gamma_1$

$\alpha_0 \delta_1$ $\alpha_0 \beta_1$

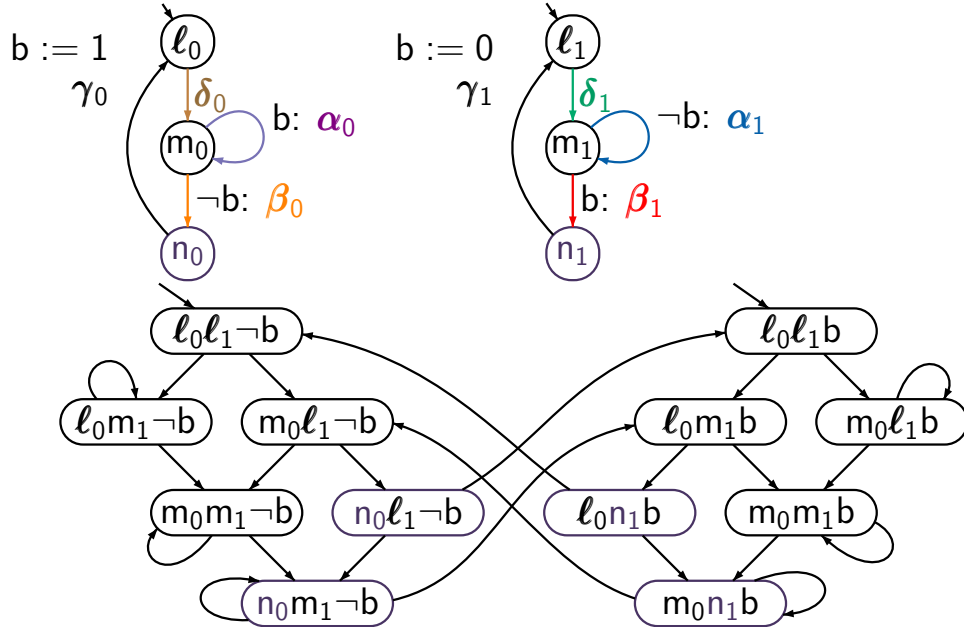
$\beta_0 \delta_1$ $\beta_0 \alpha_1$ $\beta_0 \beta_1$ $\beta_0 \gamma_1$

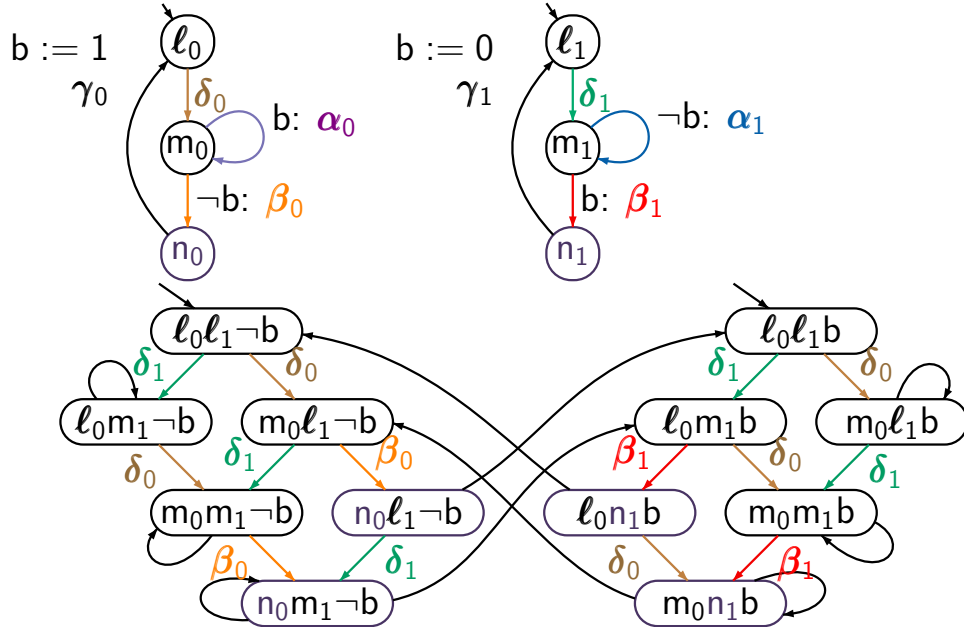
β_0 and β_1 are never enabled simultaneously

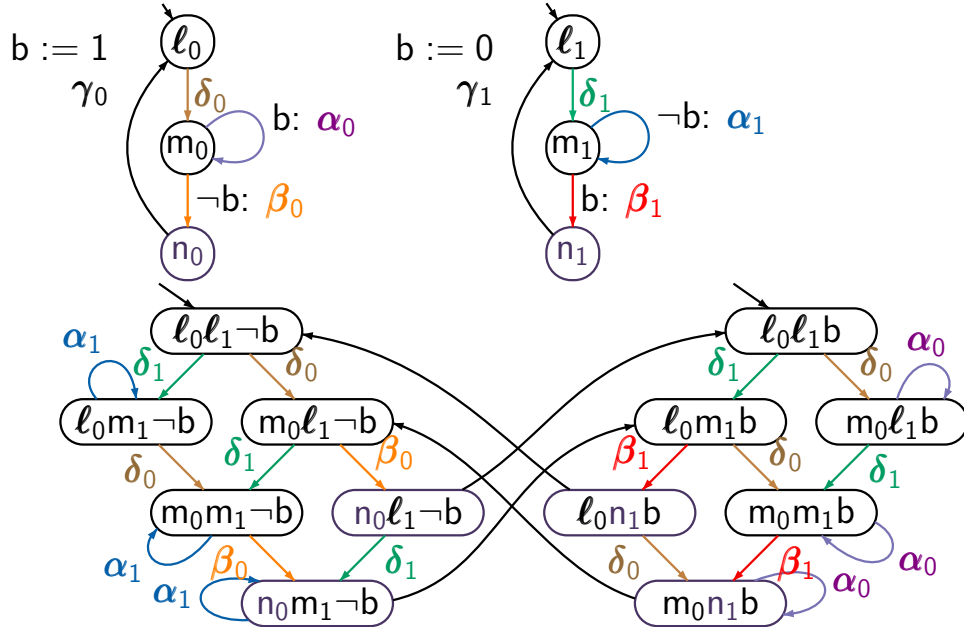


independent actions:

$\delta_0 \delta_1$	$\delta_0 \alpha_1$	$\delta_0 \beta_1$	$\delta_0 \gamma_1$
$\alpha_0 \delta_1$		$\alpha_0 \beta_1$	
$\beta_0 \delta_1$	$\beta_0 \alpha_1$	$\beta_0 \beta_1$	$\beta_0 \gamma_1$
$\gamma_0 \delta_1$		$\gamma_0 \beta_1$	

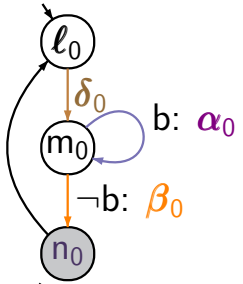




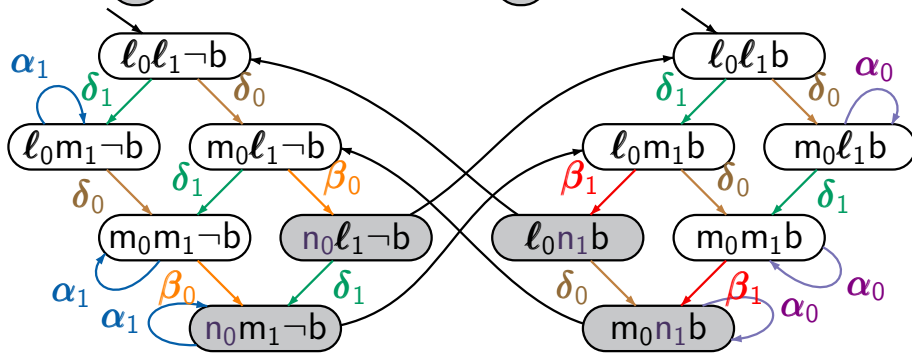
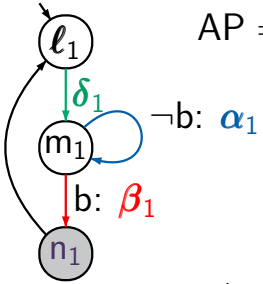


$AP = \{n_0, n_1\}$

$b := 1$
 γ_0

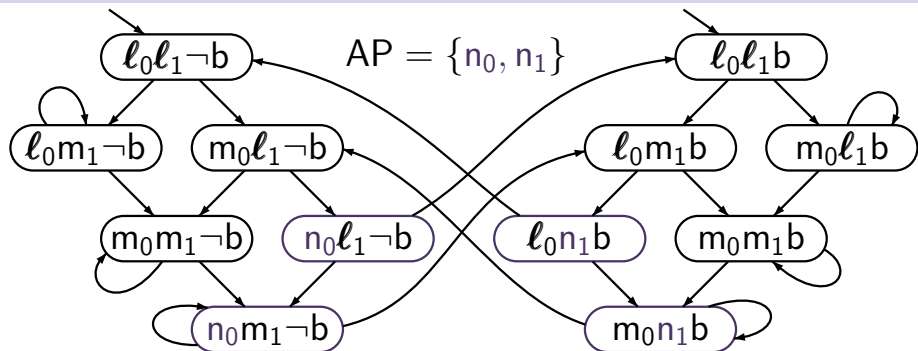


$b := 0$
 γ_1



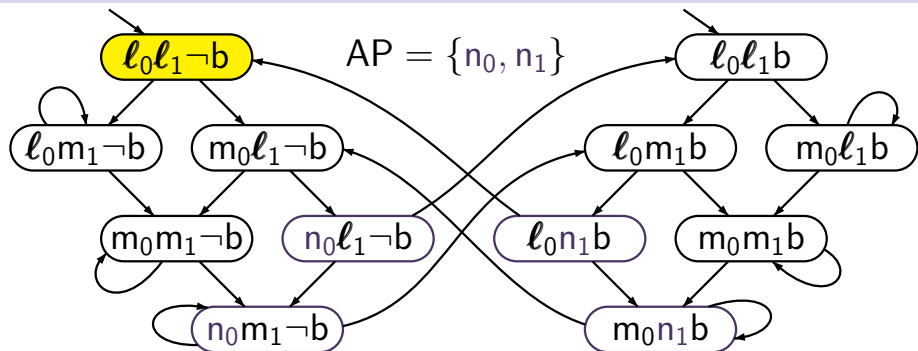
Example: on-the-fly generation of \mathcal{T}_{red}

LTL3.4-40



Example: on-the-fly generation of \mathcal{T}_{red}

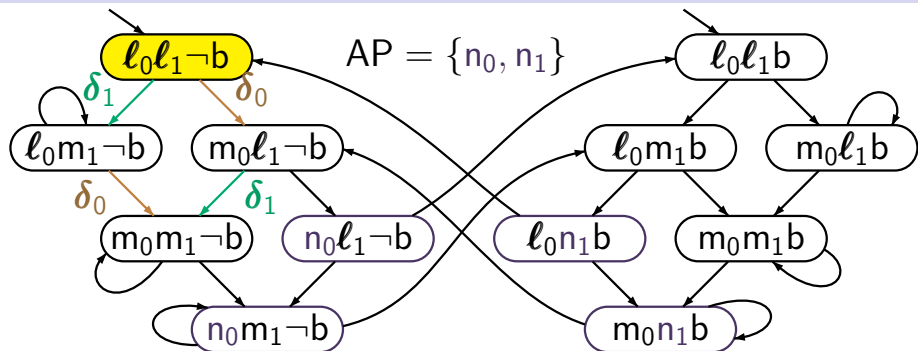
LTL3.4-40



$\text{ample}(l_0l_1\neg b) =$

Example: on-the-fly generation of \mathcal{T}_{red}

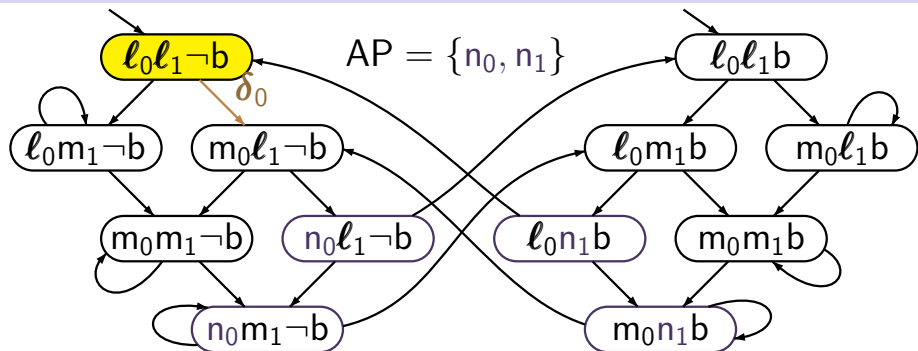
LTL3.4-40



$\text{ample}(l_0l_1\neg b) =$

Example: on-the-fly generation of \mathcal{T}_{red}

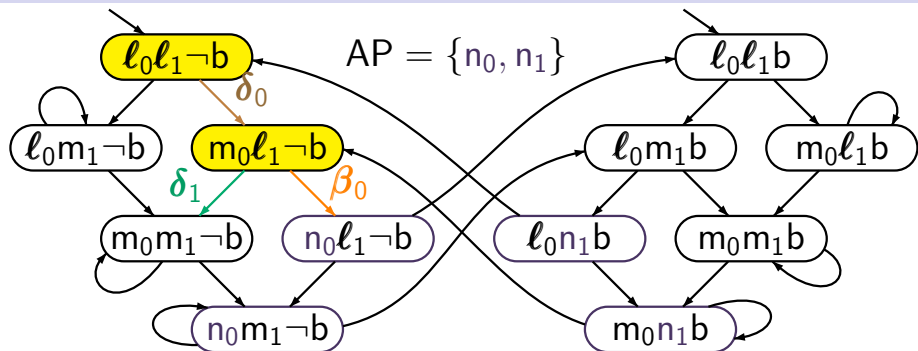
LTL3.4-40



$$\text{ample}(l_0 l_1 \neg b) = \{\delta_0\},$$

Example: on-the-fly generation of \mathcal{T}_{red}

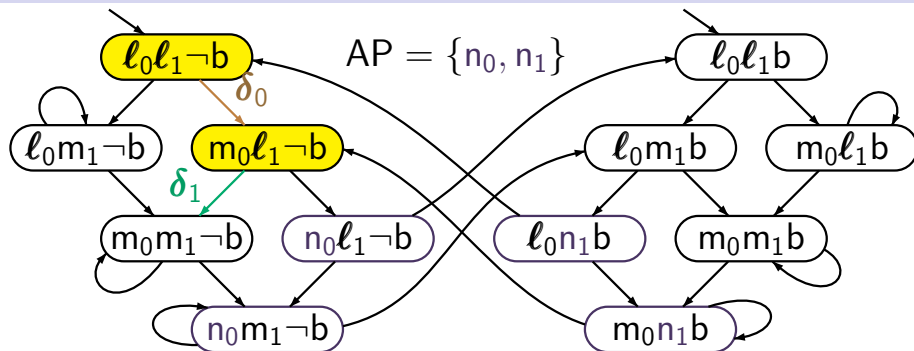
LTL3.4-40



$$\text{ample}(l_0 l_1 \neg b) = \{\delta_0\}, \quad \text{ample}(m_0 l_1 \neg b) =$$

Example: on-the-fly generation of \mathcal{T}_{red}

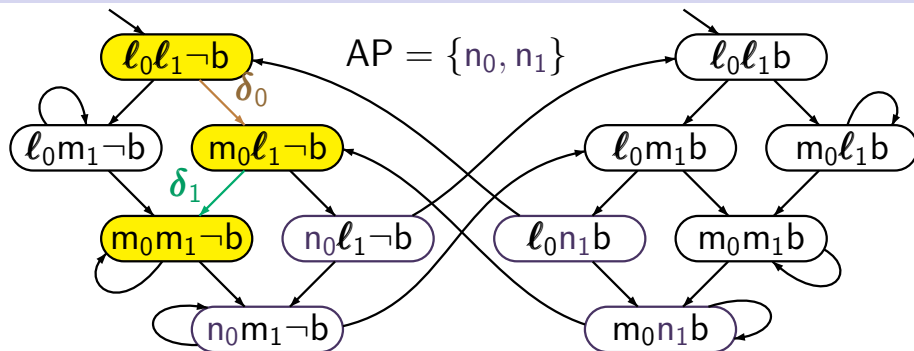
LTL3.4-40



$$\text{ample}(l_0 l_1 \neg b) = \{\delta_0\}, \quad \text{ample}(m_0 l_1 \neg b) = \{\delta_1\}$$

Example: on-the-fly generation of \mathcal{T}_{red}

LTL3.4-40

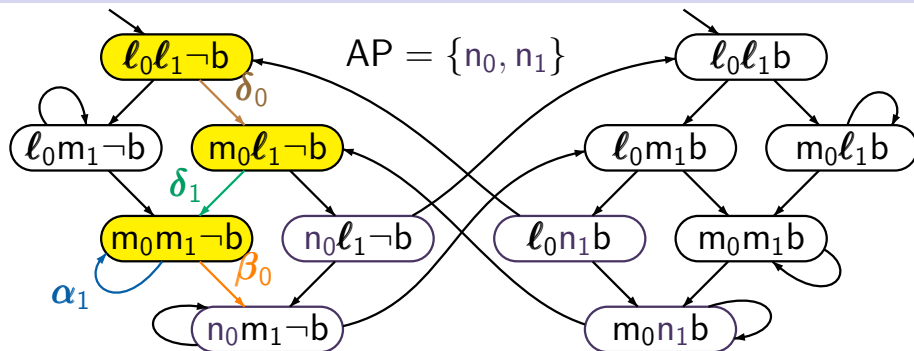


$$\text{ample}(l_0l_1\neg b) = \{\delta_0\}, \quad \text{ample}(m_0l_1\neg b) = \{\delta_1\}$$

$$\text{ample}(m_0m_1\neg b) =$$

Example: on-the-fly generation of \mathcal{T}_{red}

LTL3.4-40

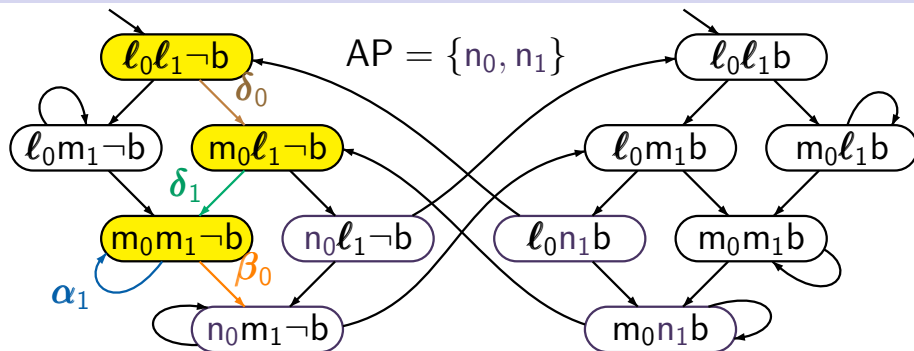


$$\text{ample}(l_0 l_1 \neg b) = \{\delta_0\}, \quad \text{ample}(m_0 l_1 \neg b) = \{\delta_1\}$$

$$\text{ample}(m_0 m_1 \neg b) =$$

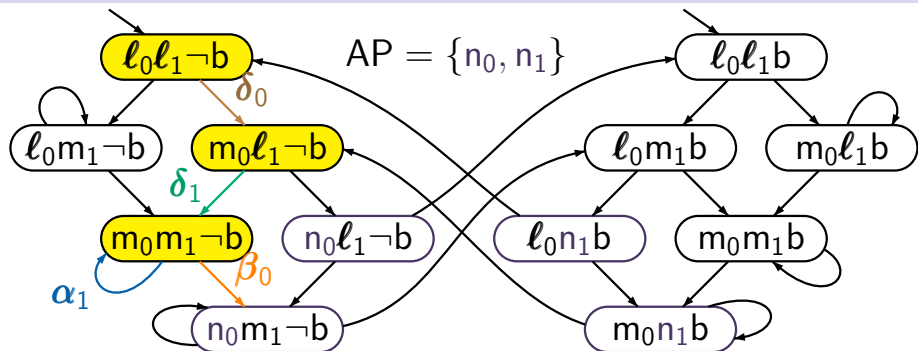
Example: on-the-fly generation of \mathcal{T}_{red}

LTL3.4-40



$$\text{ample}(l_0l_1\neg b) = \{\delta_0\}, \quad \text{ample}(m_0l_1\neg b) = \{\delta_1\}$$

$$\text{ample}(m_0m_1\neg b) = \{\alpha_1, \beta_0\}$$

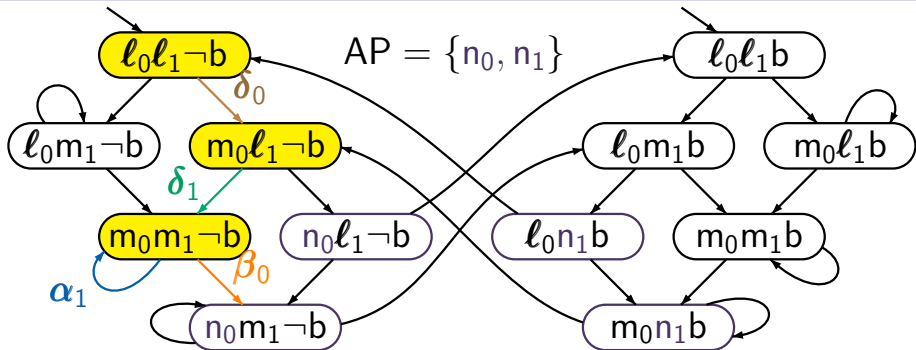


$\text{ample}(l_0 l_1 \neg b) = \{\delta_0\}, \quad \text{ample}(m_0 l_1 \neg b) = \{\delta_1\}$

$\text{ample}(m_0 m_1 \neg b) = \{\alpha_1, \beta_0\}$

note:

α_1 closes cycle (A4),



$\text{ample}(l_0l_1\neg b) = \{\delta_0\}$, $\text{ample}(m_0l_1\neg b) = \{\delta_1\}$

$\text{ample}(m_0m_1\neg b) = \{\alpha_1, \beta_0\}$

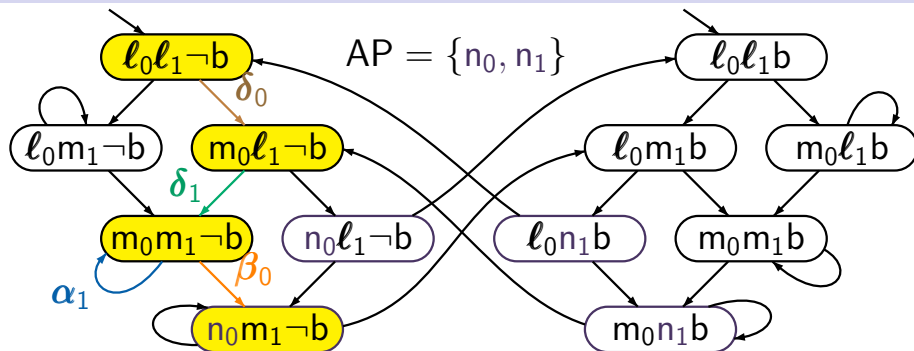
note:

α_1 closes cycle (A4),

β_0 no stutter action (A3)

Example: on-the-fly generation of \mathcal{T}_{red}

LTL3.4-40



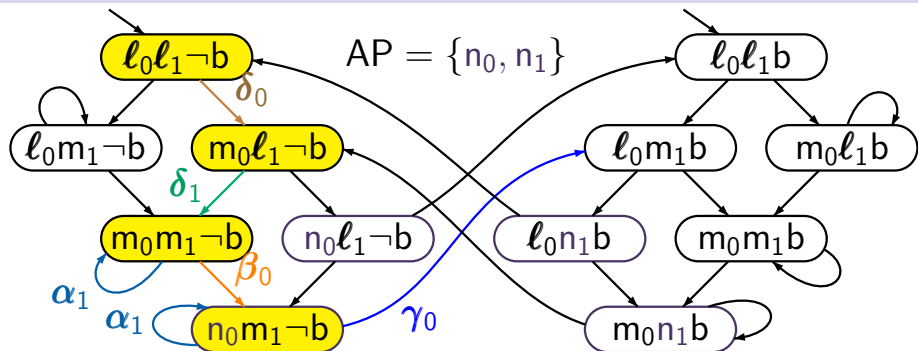
$$\text{ample}(lol_1\neg b) = \{\delta_0\}, \quad \text{ample}(mol_1\neg b) = \{\delta_1\}$$

$$\text{ample}(mom_1\neg b) = \{\alpha_1, \beta_0\}$$

$$\text{ample}(nom_1\neg b) =$$

Example: on-the-fly generation of \mathcal{T}_{red}

LTL3.4-40



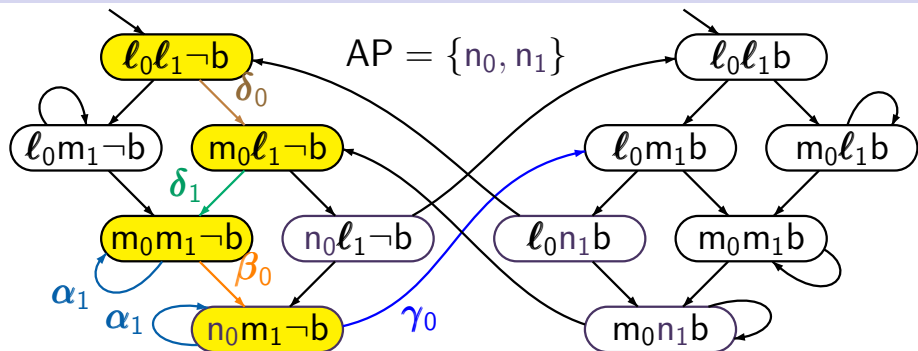
$$\text{ample}(l_0l_1\neg b) = \{\delta_0\}, \quad \text{ample}(m_0l_1\neg b) = \{\delta_1\}$$

$$\text{ample}(m_0m_1\neg b) = \{\alpha_1, \beta_0\}$$

$$\text{ample}(n_0m_1\neg b) =$$

Example: on-the-fly generation of \mathcal{T}_{red}

LTL3.4-40



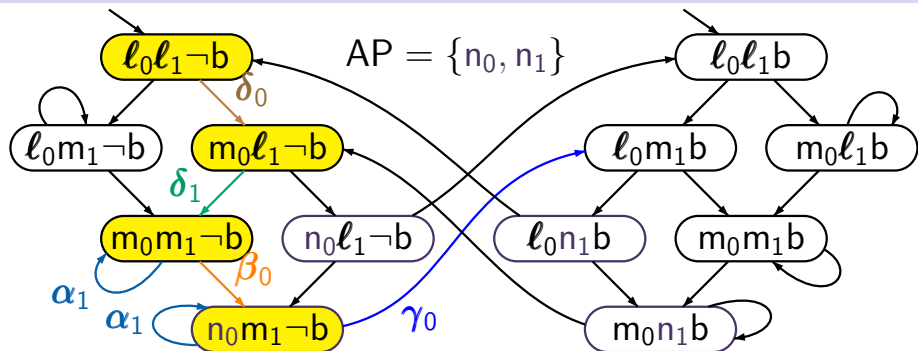
$$\text{ample}(l_0l_1\neg b) = \{\delta_0\}, \quad \text{ample}(m_0l_1\neg b) = \{\delta_1\}$$

$$\text{ample}(m_0m_1\neg b) = \{\alpha_1, \beta_0\}$$

$$\text{ample}(n_0m_1\neg b) = \{\alpha_1, \gamma_0\}$$

Example: on-the-fly generation of \mathcal{T}_{red}

LTL3.4-40



$\text{ample}(l_0l_1\neg b) = \{\delta_0\}$, $\text{ample}(m_0l_1\neg b) = \{\delta_1\}$

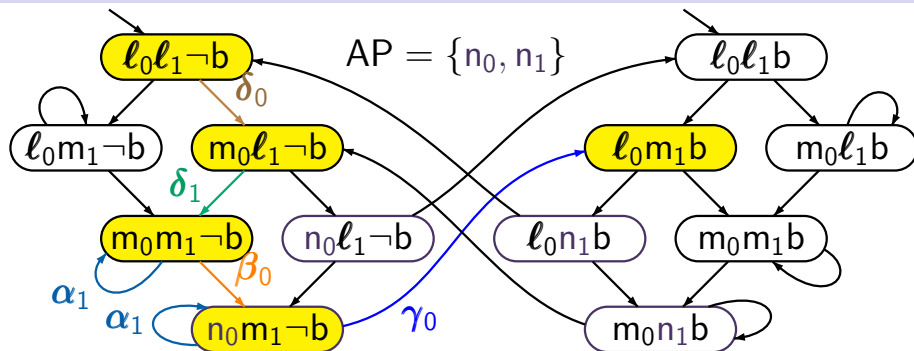
$\text{ample}(m_0m_1\neg b) = \{\alpha_1, \beta_0\}$

$\text{ample}(n_0m_1\neg b) = \{\alpha_1, \gamma_0\}$

note: α_1, γ_0 are dependent (A2)

Example: on-the-fly generation of \mathcal{T}_{red}

LTL3.4-40



$$\text{ample}(l_0l_1\neg b) = \{\delta_0\}, \quad \text{ample}(m_0l_1\neg b) = \{\delta_1\}$$

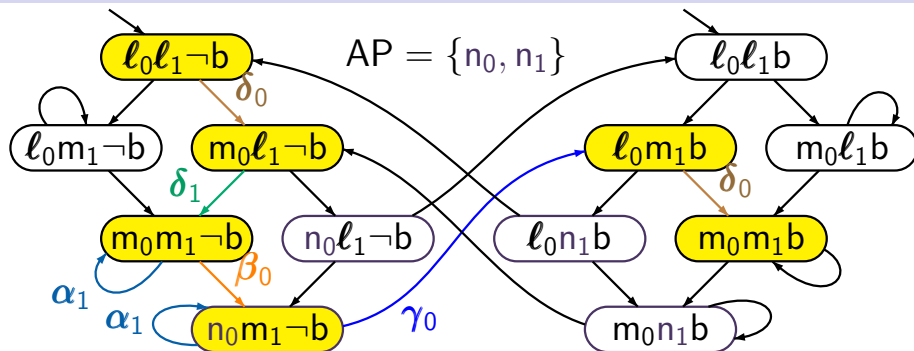
$$\text{ample}(m_0m_1\neg b) = \{\alpha_1, \beta_0\}$$

$$\text{ample}(n_0m_1\neg b) = \{\alpha_1, \gamma_0\}$$

note: α_1, γ_0 are dependent (A2)

Example: on-the-fly generation of \mathcal{T}_{red}

LTL3.4-40



$$\text{ample}(l_0l_1\neg b) = \{\delta_0\}, \quad \text{ample}(m_0l_1\neg b) = \{\delta_1\}$$

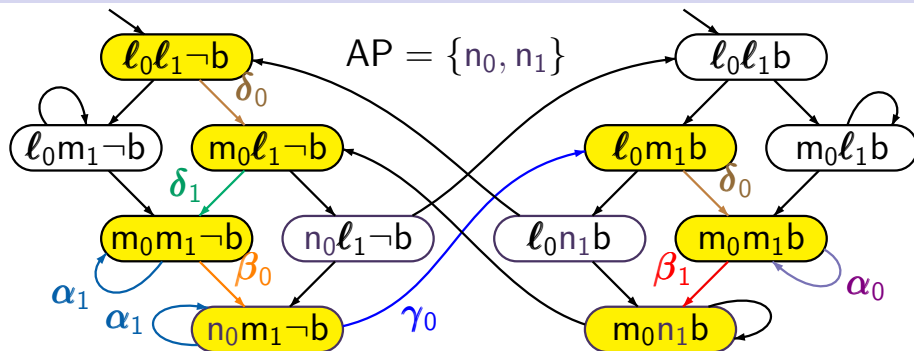
$$\text{ample}(m_0m_1\neg b) = \{\alpha_1, \beta_0\}$$

$$\text{ample}(n_0m_1\neg b) = \{\alpha_1, \gamma_0\}$$

note: α_1, γ_0 are dependent (A2)

Example: on-the-fly generation of \mathcal{T}_{red}

LTL3.4-40



$\text{ample}(l_0l_1\neg b) = \{\delta_0\}$, $\text{ample}(m_0l_1\neg b) = \{\delta_1\}$

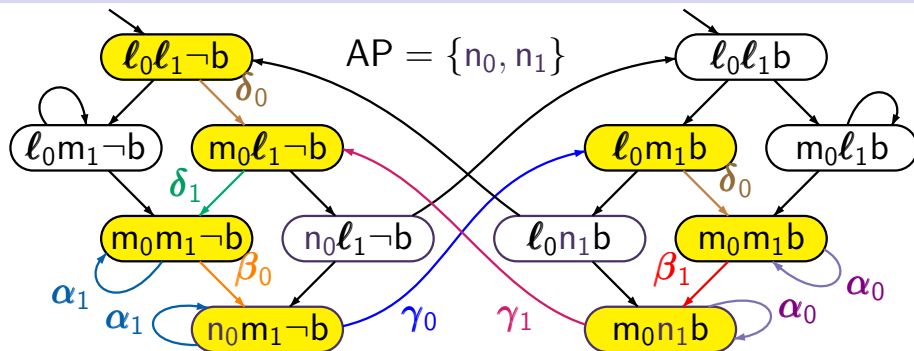
$\text{ample}(m_0m_1\neg b) = \{\alpha_1, \beta_0\}$

$\text{ample}(n_0m_1\neg b) = \{\alpha_1, \gamma_0\}$

note: α_1, γ_0 are dependent (A2)

Example: on-the-fly generation of \mathcal{T}_{red}

LTL3.4-40



$$\text{ample}(l_0l_1\neg b) = \{\delta_0\}, \quad \text{ample}(m_0l_1\neg b) = \{\delta_1\}$$

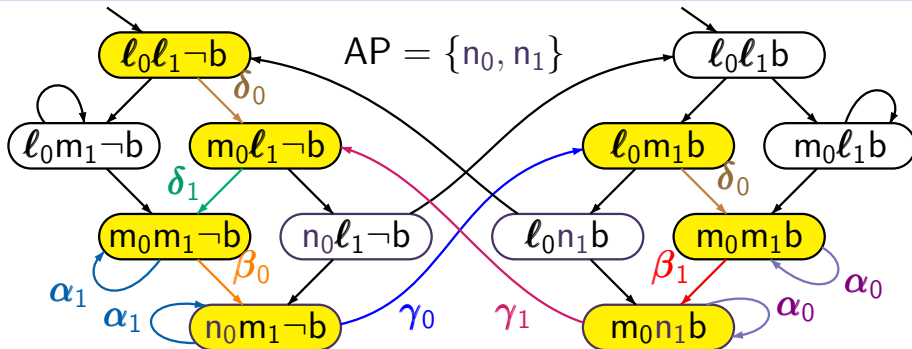
$$\text{ample}(m_0m_1\neg b) = \{\alpha_1, \beta_0\}$$

$$\text{ample}(n_0m_1\neg b) = \{\alpha_1, \gamma_0\}$$

$$\text{ample}(m_0n_1b) =$$

Example: on-the-fly generation of \mathcal{T}_{red}

LTL3.4-40



$\text{ample}(l_0l_1\neg b) = \{\delta_0\}$, $\text{ample}(m_0l_1\neg b) = \{\delta_1\}$

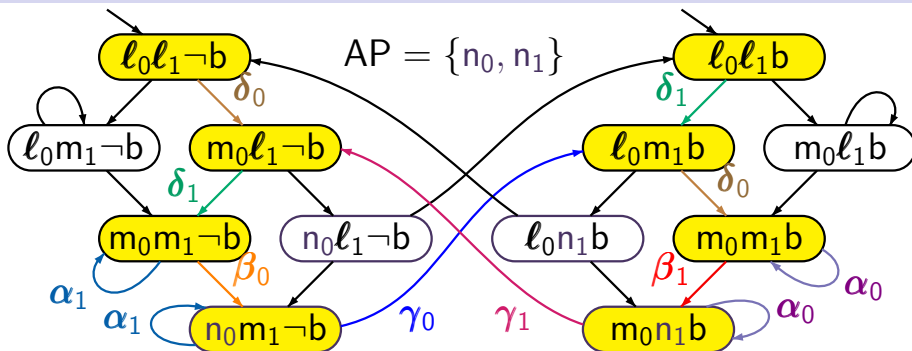
$\text{ample}(m_0m_1\neg b) = \{\alpha_1, \beta_0\}$

$\text{ample}(n_0m_1\neg b) = \{\alpha_1, \gamma_0\}$

$\text{ample}(m_0n_1b) = \{\alpha_0, \gamma_1\}$: cycle condition (A4)

Example: on-the-fly generation of \mathcal{T}_{red}

LTL3.4-40

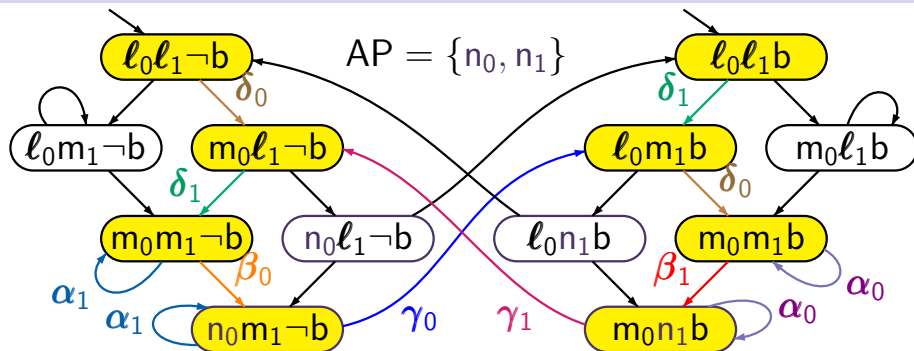


$\text{ample}(l_0l_1\neg b) = \{\delta_0\}$, $\text{ample}(m_0l_1\neg b) = \{\delta_1\}$

$\text{ample}(m_0m_1\neg b) = \{\alpha_1, \beta_0\}$

$\text{ample}(n_0m_1\neg b) = \{\alpha_1, \gamma_0\}$

$\text{ample}(m_0n_1b) = \{\alpha_0, \gamma_1\}$: cycle condition (A4)



$\text{ample}(l_0l_1\neg b) = \{\delta_0\}$, $\text{ample}(m_0l_1\neg b) = \{\delta_1\}$

$\text{ample}(m_0m_1\neg b) = \{\alpha_1, \beta_0\}$

$\text{ample}(n_0m_1\neg b) = \{\alpha_1, \gamma_0\}$

$\text{ample}(m_0n_1b) = \{\alpha_0, \gamma_1\}$: cycle condition (A4)

Nested DFS with POR

LTL3.4-41

Nested DFS (standard approach)

LTL3.4-41

remind: nested DFS for checking “ $\mathcal{T} \models \diamond \Box a?$ ” uses:

outer DFS: visits all reachable states

inner DFS: $\text{CYCLE_CHECK}(s)$ searches for a backward edge $s' \rightarrow s$

$\text{CYCLE_CHECK}(s)$

- is called for each state s that violates the persistence condition a
- must not be started before the **outer DFS** is finished for s

outer DFS: visits all reachable states

inner DFS: $CYCLE_CHECK(s)$ searches for a backward edge $s' \rightarrow s$

$CYCLE_CHECK(s)$

- is called for each state s that violates the persistence condition a
- must not be started before the outer DFS is finished for s
- early termination, e.g., abort with the answer

$CYCLE_CHECK(s) = true$

as soon as the inner DFS visits a state in the DFS-stack of the outer DFS

Nested DFS with POR

LTL3.4-41

requirement for the nested DFS in the ample set approach:

outer DFS and inner DFS must use
the same ample-sets

Nested DFS with POR

LTL3.4-41

requirement for the nested DFS in the ample set approach:

outer DFS and inner DFS must use the same ample-sets

implementation: uses a hash-table for the set of states that have been visited in the outer DFS

Implementation of the nested DFS with POR

LTL3.4-41

use *hash-table* for the set of states that have been visited in the **outer DFS**

entries in the hash-table have the form

$$\langle s, b, c, a_1, \dots, a_k \rangle$$

where s is a state and b, c, a_1, \dots, a_k are bits

- $b = 1$ iff s has been visited in **inner DFS**
- $c = 1$ iff s is in the **DFS stack**
- for $Act(s) = \{\alpha_1, \dots, \alpha_k\}$:

$$a_i = 1 \text{ iff } \alpha_i \in \text{ample}(s)$$

On-the-fly construction of \mathcal{T}_{red}

LTL3.4-42

starting point: syntactic description of the processes

P_1, \dots, P_n of a parallel system

e.g., PROMELA-specification

On-the-fly construction of \mathcal{T}_{red} in DFS-manner

LTL3.4-42

starting point: syntactic description of the processes

P_1, \dots, P_n of a parallel system

e.g., PROMELA-specification

method: generate the reachable fragment of \mathcal{T}_{red} in DFS-manner by generating **ample sets** by means of **local conditions** that ensure (A1)-(A4)

On-the-fly construction of \mathcal{T}_{red} in DFS-manner

LTL3.4-42

starting point: syntactic description of the processes

P_1, \dots, P_n of a parallel system

e.g., PROMELA-specification

method: generate the reachable fragment of \mathcal{T}_{red} in DFS-manner by generating **ample sets** by means of **local conditions** that ensure (A1)-(A4)

idea: check whether

ample(s) = set of enabled actions of **process P_i**

fulfills (A1), (A2), (A3)

On-the-fly construction of \mathcal{T}_{red} in DFS-manner

LTL3.4-42

starting point: syntactic description of the processes

P_1, \dots, P_n of a parallel system

e.g., PROMELA-specification

method: generate the reachable fragment of \mathcal{T}_{red} in DFS-manner by generating **ample sets** by means of **local conditions** that ensure (A1)-(A4)

idea: check whether

ample(s) = set of enabled actions of **process P_i**

fulfills (A1), (A2), (A3) and ensure (A4) by searching for **backward edges** in \mathcal{T}_{red}

REPEAT

select a process P_i not considered before

REPEAT

select a process P_i not considered before

$A :=$ action set of $P_i \cap Act(s)$

REPEAT

select a process P_i not considered before

$A :=$ action set of $P_i \cap Act(s)$

IF $A \neq \emptyset$ and (A2) is not violated

REPEAT

select a process P_i not considered before

$A :=$ action set of $P_i \cap Act(s)$

IF $A \neq \emptyset$ and (A2) is not violated

and all actions of A are stutter actions

REPEAT

select a process P_i not considered before

$A :=$ action set of $P_i \cap Act(s)$

IF $A \neq \emptyset$ and (A2) is not violated

and all actions of A are stutter actions

THEN $ample(s) := A$ FI

REPEAT

select a process P_i not considered before

$A :=$ action set of $P_i \cap Act(s)$

IF $A \neq \emptyset$ and (A2) is not violated

and all actions of A are stutter actions

THEN $ample(s) := A$ FI

UNTIL all processes have been considered
or $ample(s)$ is defined;

REPEAT

select a process P_i not considered before

$A :=$ action set of $P_i \cap Act(s)$

IF $A \neq \emptyset$ and (A2) is not violated

and all actions of A are stutter actions

THEN $ample(s) := A$ FI

UNTIL all processes have been considered
or $ample(s)$ is defined;

IF $ample(s)$ is not yet defined

THEN $ample(s) := Act(s)$ FI

REPEAT

select a process P_i not considered before

$A :=$ action set of $P_i \cap Act(s)$

IF (A1), (A2), (A3) hold THEN $ample(s) := A$ FI

UNTIL all processes have been considered

or $ample(s)$ is defined;

IF $ample(s)$ is not yet defined

THEN $ample(s) := Act(s)$ FI

... consider state $\alpha(s)$ for some $\alpha \in ample(s)$...

REPEAT

select a process P_i not considered before

$A :=$ action set of $P_i \cap Act(s)$

IF (A1), (A2), (A3) hold THEN $ample(s) := A$ FI

UNTIL all processes have been considered

or $ample(s)$ is defined;

IF $ample(s)$ is not yet defined

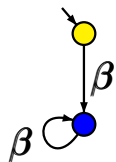
THEN $ample(s) := Act(s)$ FI

... consider state $\alpha(s)$ for some $\alpha \in ample(s)$...

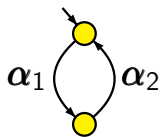
IF the expansion of s finds a backwards edge $s' \implies s$

THEN $ample(s) := Act(s)$ FI

process 1



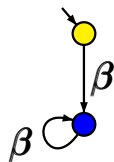
process 2



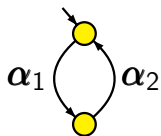
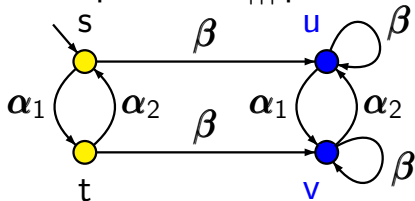
Example: construction of \mathcal{T}_{red}

LTL3.4-43

process 1



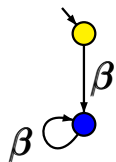
process 2

 $\mathcal{T} = \text{process 1} \parallel \text{process 2}$ 

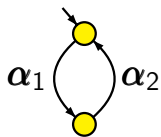
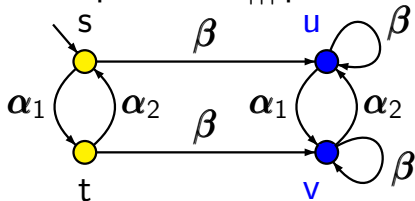
Example: construction of \mathcal{T}_{red}

LTL3.4-43

process 1



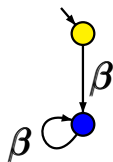
process 2

 $\mathcal{T} = \text{process 1} \parallel \text{process 2}$ DFS(s)

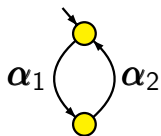
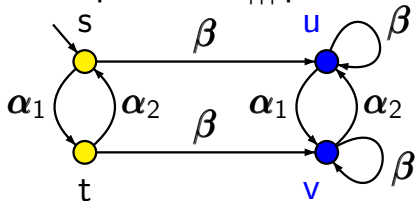
Example: construction of \mathcal{T}_{red}

LTL3.4-43

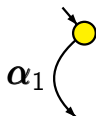
process 1



process 2

 $\mathcal{T} = \text{process 1} \parallel \text{process 2}$ DFS(s)

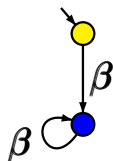
$$\text{ample}(s) = \{\alpha_1\}$$



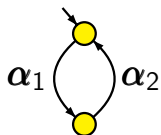
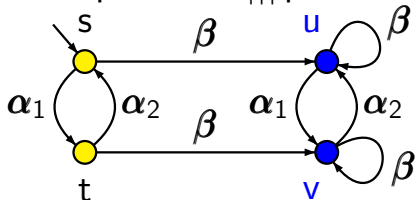
Example: construction of \mathcal{T}_{red}

LTL3.4-43

process 1



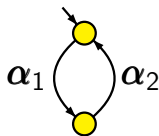
process 2

 $\mathcal{T} = \text{process 1} \parallel \text{process 2}$ 

DFS(s)

ample(s) = $\{\alpha_1\}$

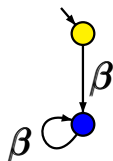
DFS(t)

ample(t) = $\{\alpha_2\}$ 

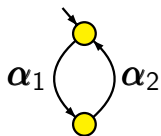
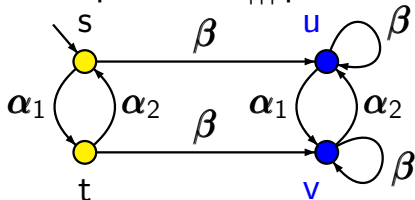
Example: construction of \mathcal{T}_{red}

LTL3.4-43

process 1



process 2

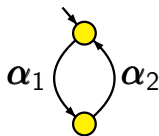
 $\mathcal{T} = \text{process 1} \parallel \text{process 2}$ 

DFS(s)

$$\text{ample}(s) = \{\alpha_1\}$$

DFS(t)

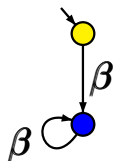
$$\text{ample}(t) = \{\alpha_2\}$$

backward edge $t \rightarrow s$ 

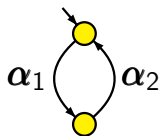
Example: construction of \mathcal{T}_{red}

LTL3.4-43

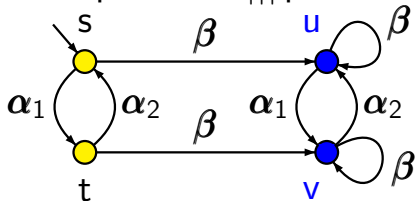
process 1



process 2



$\mathcal{T} = \text{process 1} \parallel \text{process 2}$



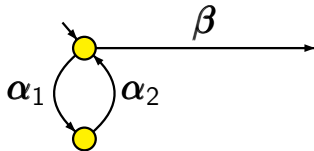
DFS(s)

$$\text{ample}(s) = \{\alpha_1\} \cup \{\beta\}$$

DFS(t)

$$\text{ample}(t) = \{\alpha_2\}$$

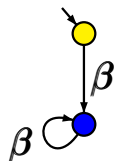
backward edge $t \rightarrow s$



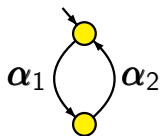
Example: construction of \mathcal{T}_{red}

LTL3.4-43

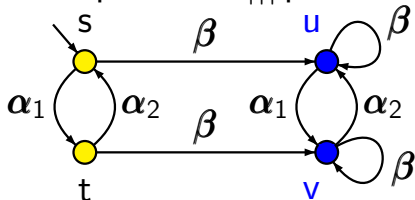
process 1



process 2



$\mathcal{T} = \text{process 1} \parallel \parallel \text{process 2}$



DFS(s)

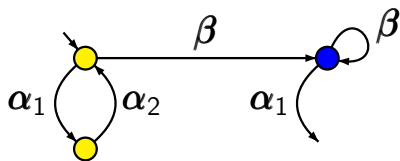
$$\text{ample}(s) = \{\alpha_1\} \cup \{\beta\}$$

DFS(t)

$$\text{ample}(t) = \{\alpha_2\}$$

backward edge $t \rightarrow s$

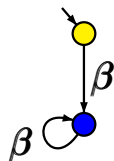
DFS(u) ...



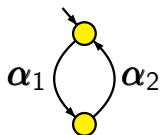
Example: construction of \mathcal{T}_{red}

LTL3.4-43

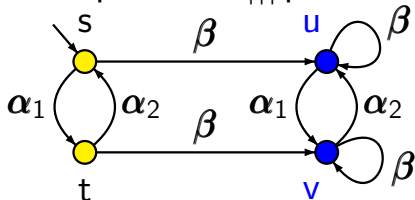
process 1



process 2



$\mathcal{T} = \text{process 1} \parallel \text{process 2}$



DFS(s)

$$\text{ample}(s) = \{\alpha_1\} \cup \{\beta\}$$

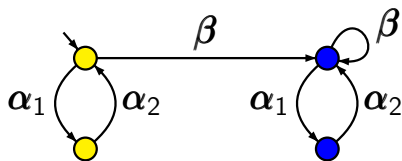
DFS(t)

$$\text{ample}(t) = \{\alpha_2\}$$

backward edge $t \rightarrow s$

DFS(u) ...

DFS(v) ...



REPEAT

select a process P_i not considered before

$A :=$ action set of $P_i \cap Act(s)$

IF $A \neq \emptyset$ and (A2) holds

and all actions of A are stutter actions

THEN $ample(s) := A$ FI

UNTIL all processes have been considered
or $ample(s)$ is defined;

IF $ample(s)$ is not yet defined

THEN $ample(s) := Act(s)$ FI

(A1) nonemptiness condition

(A2) dependence condition:

for each execution fragment in \mathcal{T}

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

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

(A3) stutter condition

(A4) cycle condition

(A1) nonemptiness condition

(A2) dependence condition:

for each execution fragment in \mathcal{T}

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

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

(A3) stutter condition

(A4) cycle condition

checking (A2) is as hard as the **unreachability problem**

given: finite transition system \mathcal{T} , $\mathbf{a} \in AP$

question: does $\mathcal{T} \not\models \exists \diamond \mathbf{a}$ hold?

(A1) nonemptiness condition

(A2) dependence condition: \longleftarrow global condition

for each execution fragment in \mathcal{T}

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

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

(A3) stutter condition

(A4) cycle condition

checking (A2) is as hard as the **unreachability problem**

given: finite transition system \mathcal{T} , $\mathbf{a} \in AP$

question: does $\mathcal{T} \not\models \exists \diamond \mathbf{a}$ hold?

show that the **unreachability problem**

given: finite transition system \mathcal{T}

$\mathbf{a} \in AP$

question: does $\mathcal{T} \not\models \exists \diamond \mathbf{a}$ hold?

is **polynomially reducible** to the problem of checking (A2)

show that the **unreachability problem**

given: finite transition system \mathcal{T}

$\mathbf{a} \in AP$

question: does $\mathcal{T} \not\models \exists \diamond \mathbf{a}$ hold?

is **polynomially reducible** to the **problem of checking (A2)**

given: finite transition system \mathcal{T}' , ample sets for \mathcal{T}'

question: does (A2) hold?

i.e., does for each execution fragment in \mathcal{T}'

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

such that β_n is *dependent* from $\text{ample}(s)$

there is some $i < n$ with $\beta_i \in \text{ample}(s)$?

show that the **unreachability problem**

given: finite transition system \mathcal{T} and initial state s_0
 $a \in AP$

question: does $s_0 \not\models \exists \diamond a$ hold?

is **polynomially reducible** to the **problem of checking (A2)**

given: finite transition system \mathcal{T}' , ample sets for \mathcal{T}'

question: does (A2) hold?

i.e., does for each execution fragment in \mathcal{T}'

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

such that β_n is *dependent* from $\text{ample}(s)$

there is some $i < n$ with $\beta_i \in \text{ample}(s)$?

unreachability
problem

\leq_{poly}

problem of
checking (A2)

unreachability problem	\leq_{poly}	problem of checking (A2)
---------------------------	----------------------	-----------------------------

finite TS \mathcal{T} + state s_0
+ atomic prop. \mathbf{a}

\rightsquigarrow

finite TS \mathcal{T}'
+ ample sets

s.t. $s_0 \not\models \exists \diamond \mathbf{a}$ iff (A2) holds

\mathcal{T}' results from \mathcal{T} by adding two fresh actions α, β s.t.

- α and β are dependent
- α is independent from all actions in \mathcal{T}
- β is enabled exactly in the states \mathbf{t} with $\mathbf{t} \models \mathbf{a}$

finite TS \mathcal{T} + state s_0
+ atomic prop. \mathbf{a}

\rightsquigarrow

finite TS \mathcal{T}'
+ ample sets

s.t.

$s_0 \not\models \exists \diamond \mathbf{a}$

iff

(A2) holds

finite TS \mathcal{T} + state s_0
+ atomic prop. \mathbf{a}

\rightsquigarrow

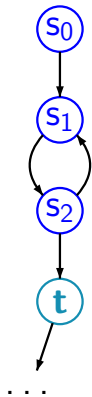
finite TS \mathcal{T}'
+ ample sets

s.t.

$s_0 \not\models \exists \diamond \mathbf{a}$

iff

(A2) holds



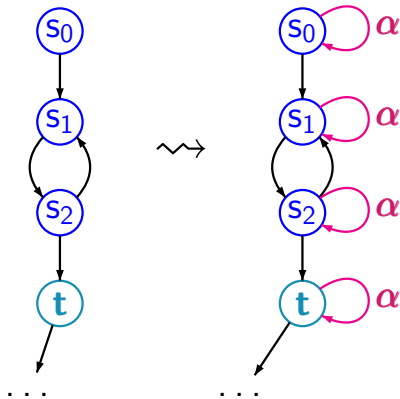
finite TS \mathcal{T} + state s_0
+ atomic prop. \mathbf{a}

finite TS \mathcal{T}'
+ ample sets

s.t.

$s_0 \not\models \exists \diamond \mathbf{a}$

iff (A2) holds



finite TS \mathcal{T} + state s_0
+ atomic prop. \mathbf{a}

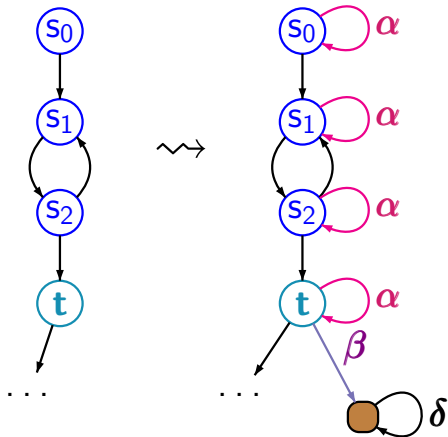
\rightsquigarrow

finite TS \mathcal{T}'
+ ample sets

s.t.

$s_0 \not\models \exists \diamond \mathbf{a}$

iff (A2) holds



finite TS \mathcal{T} + state s_0
+ atomic prop. \mathbf{a}

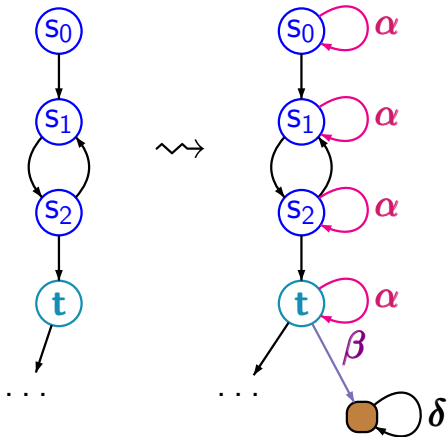
s.t.

$s_0 \not\models \exists \diamond \mathbf{a}$

\rightsquigarrow

finite TS \mathcal{T}'
+ ample sets

iff (A2) holds



α, β dependent
 α independent from
all other actions

finite TS \mathcal{T} + state s_0
 + atomic prop. \mathbf{a}

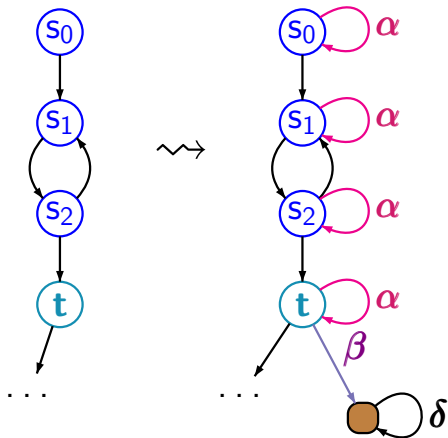
s.t.

$s_0 \not\models \exists \diamond \mathbf{a}$

\rightsquigarrow

finite TS \mathcal{T}'
 + ample sets

iff (A2) holds



α, β dependent
 α independent from
 all other actions

$\text{ample}(s_0) = \{\alpha\}$
 $\text{ample}(u) = \text{Act}(u)$
 for all other states u

Local criterion for condition (A2)

LTL3.4-45

idea: replace the **global** dependency condition (A2) by a **stronger local** condition

Local criterion for condition (A2)

LTL3.4-45

idea: replace the **global** dependency condition (A2) by a **stronger local** condition that can be derived from the **syntactic description** for the **processes** P_i of the given parallel system

$$P_1 \parallel \dots \parallel P_n$$

Local criterion for condition (A2)

LTL3.4-45

idea: replace the **global** dependency condition (A2) by a **stronger local** condition that can be derived from the **syntactic description** for the **processes** P_i of the given parallel system

$$P_1 \parallel \dots \parallel P_n$$

e.g., the P_i 's are given as **program graphs** of a channel system.

Local criterion for condition (A2)

LTL3.4-45

idea: replace the **global** dependency condition (A2) by a **stronger local** condition that can be derived from the **syntactic description** for the **processes** P_i of the given parallel system

$$P_1 \parallel \dots \parallel P_n$$

e.g., the P_i 's are given as **program graphs** of a channel system. Then: each state s has the form

$$s = \langle l_1, \dots, l_n, \eta, \xi \rangle$$

where l_i is a location of process P_i , η a variable evaluation, ξ a channel evaluation

Local criterion for condition (A2)

LTL3.4-45

Let Act_i denote the set of actions of process P_i .

For state s :

$$\begin{aligned} Act_i(s) &= Act_i \cap Act(s) \\ &= \text{set of actions of process } P_i \\ &\quad \text{that are enabled in } s \end{aligned}$$

Local criterion for condition (A2)

LTL3.4-45

Let Act_i denote the set of actions of process P_i .

For state s :

$$\begin{aligned} Act_i(s) &= Act_i \cap Act(s) \\ &= \text{set of actions of process } P_i \\ &\quad \text{that are enabled in } s \end{aligned}$$

Provide **local criteria** such that $ample(s) = Act_i(s)$ fulfills the dependency condition (A2)

Let $s = \langle l_1, \dots, l_{i-1}, l_i, l_{i+1}, \dots, l_n, \dots \rangle$.

Suppose that

Let $s = \langle l_1, \dots, l_{i-1}, l_i, l_{i+1}, \dots, l_n, \dots \rangle$.

Suppose that

(A2.1) all actions of P_j , $j \neq i$, are independent from $Act_i(s)$

Let $\mathbf{s} = \langle \ell_1, \dots, \ell_{i-1}, \ell_i, \ell_{i+1}, \dots, \ell_n, \dots \rangle$.

Suppose that

- (A2.1) all actions of \mathbf{P}_j , $j \neq i$, are independent from $Act_i(\mathbf{s})$, i.e., if $\gamma \in Act_j$ for some $j \neq i$, and $\alpha \in Act_i(\mathbf{s})$ then α and γ are independent

Let $s = \langle l_1, \dots, l_{i-1}, l_i, l_{i+1}, \dots, l_n, \dots \rangle$.

Suppose that

- (A2.1) all actions of P_j , $j \neq i$, are independent from $Act_i(s)$, i.e., if $\gamma \in Act_j$ for some $j \neq i$, and $\alpha \in Act_i(s)$ then α and γ are independent
- (A2.2) there is no action γ of a process P_j where $j \neq i$ s.t.

Let $s = \langle l_1, \dots, l_{i-1}, l_i, l_{i+1}, \dots, l_n, \dots \rangle$.

Suppose that

(A2.1) all actions of P_j , $j \neq i$, are independent from $Act_i(s)$, i.e., if $\gamma \in Act_j$ for some $j \neq i$, and $\alpha \in Act_i(s)$ then α and γ are independent

(A2.2) there is no action γ of a process P_j where $j \neq i$ s.t. γ can enable an action $\beta \in Act_i \setminus Act(s)$ from some state s' with location l_i for process P_i

Let $s = \langle \dots, l_j, \dots, l_i, \dots, l_n, \dots \rangle$.

Suppose that

(A2.1) all actions of P_j , $j \neq i$, are independent from $Act_i(s)$, i.e., if $\gamma \in Act_j$ for some $j \neq i$, and $\alpha \in Act_i(s)$ then α and γ are independent

(A2.2) there is no action γ of a process P_j where $j \neq i$ s.t.

$$\langle \dots h_j \dots l_i \dots \rangle \xrightarrow{\gamma} \langle \dots k_j \dots l_i \dots \rangle \xrightarrow{\beta}$$

$\beta \not\in$

for some $\beta \in Act_i \setminus Act(s)$

Let $s = \langle \dots, l_j, \dots, l_i, \dots, l_n, \dots \rangle$.

Suppose that

(A2.1) all actions of P_j , $j \neq i$, are independent from $Act_i(s)$, i.e., if $\gamma \in Act_j$ for some $j \neq i$, and $\alpha \in Act_i(s)$ then α and γ are independent

(A2.2) there is no action γ of a process P_j where $j \neq i$ s.t.

$$\langle \dots h_j \dots l_i \dots \rangle \xrightarrow{\gamma} \langle \dots k_j \dots l_i \dots \rangle \xrightarrow{\beta}$$

$\beta \not\in$

for some $\beta \in Act_i \setminus Act(s)$

Then (A2) holds for $ample(s) = Act_i(s)$.

Heuristics for condition (A2)

LTL3.4-45

∴ expansion of state $s = \langle \dots l_j \dots l_i \dots \rangle$

Heuristics for condition (A2)

LTL3.4-45

∴ expansion of state $s = \langle \dots l_j \dots l_i \dots \rangle$

$A := Act_i(s)$

Heuristics for condition (A2)

LTL3.4-45

⋮ expansion of state $s = \langle \dots \ell_j \dots \ell_i \dots \rangle$

$A := Act_i(s)$

⋮

check if for all other processes P_j the following holds:

Heuristics for condition (A2)

LTL3.4-45

⋮ expansion of state $s = \langle \dots \ell_j \dots \ell_i \dots \rangle$

$A := Act_i(s)$

⋮

check if for all other processes P_j the following holds:

(A2.1) all actions of P_j are independent from A

Heuristics for condition (A2)

LTL3.4-45

⋮ expansion of state $s = \langle \dots \ell_j \dots \ell_i \dots \rangle$

$A := Act_i(s)$

⋮

check if for all other processes P_j the following holds:

(A2.1) all actions of P_j are independent from A

(A2.2) there is no action γ of P_j such that

Heuristics for condition (A2)

LTL3.4-45

⋮ expansion of state $s = \langle \dots \ell_j \dots \ell_i \dots \rangle$

$A := Act_i(s)$

⋮

check if for all other processes P_j the following holds:

(A2.1) all actions of P_j are independent from A

(A2.2) there is no action γ of P_j such that

$$\langle \dots h_j \dots \ell_i \dots \rangle \xrightarrow{\gamma} \langle \dots k_j \dots \ell_i \dots \rangle \xrightarrow{\beta}$$

$\beta \not\downarrow$

for some $\beta \in Act_i \setminus A$

Heuristics for condition (A2)

LTL3.4-45

∴ expansion of state $s = \langle \dots \ell_j \dots \ell_i \dots \rangle$

$A := Act_i(s)$

∴

check if for all other processes P_j the following holds:

(A2.1) all actions of P_j are independent from A

(A2.2) there is no action γ of P_j such that

$$\langle \dots h_j \dots \ell_i \dots \rangle \xrightarrow{\gamma} \langle \dots k_j \dots \ell_i \dots \rangle \xrightarrow{\beta}$$

$\beta \not\downarrow$

for some $\beta \in Act_i \setminus A$

if yes then set $ample(s) := A$

∴