Tecniche di Specifica e di Verifica

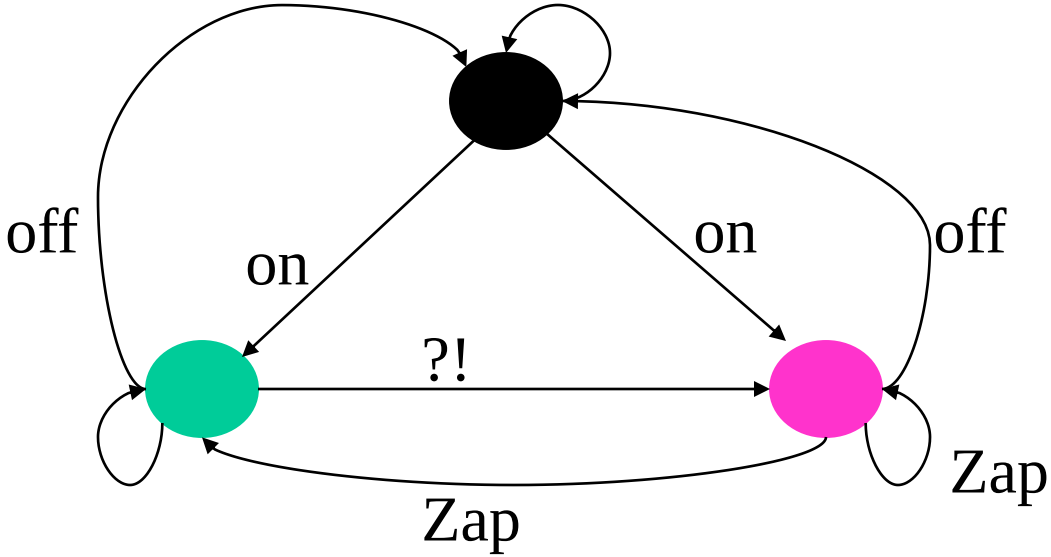
Branching Time Temporal Logics I

Outline

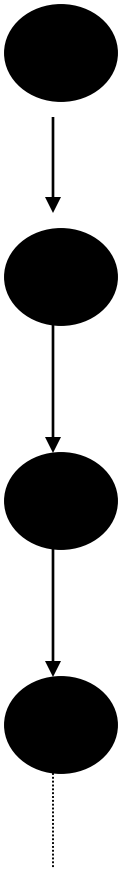
- ***CTL*** (**C**omputation **T**ree **L**ogic)
  - **Branching Time**
  - Unwindings --- computation trees
  - Syntax and semantics of CTL.

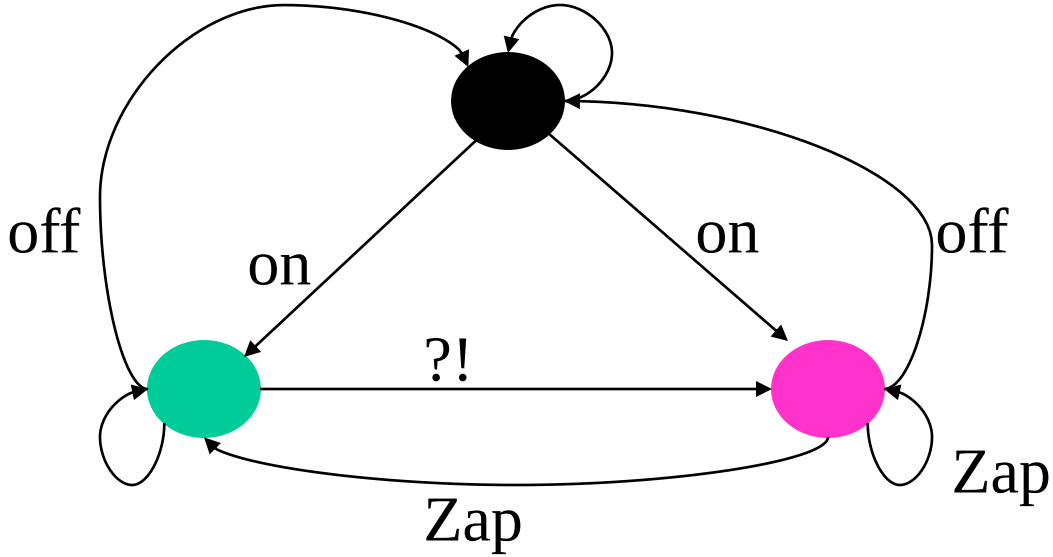
Branching Time Structures

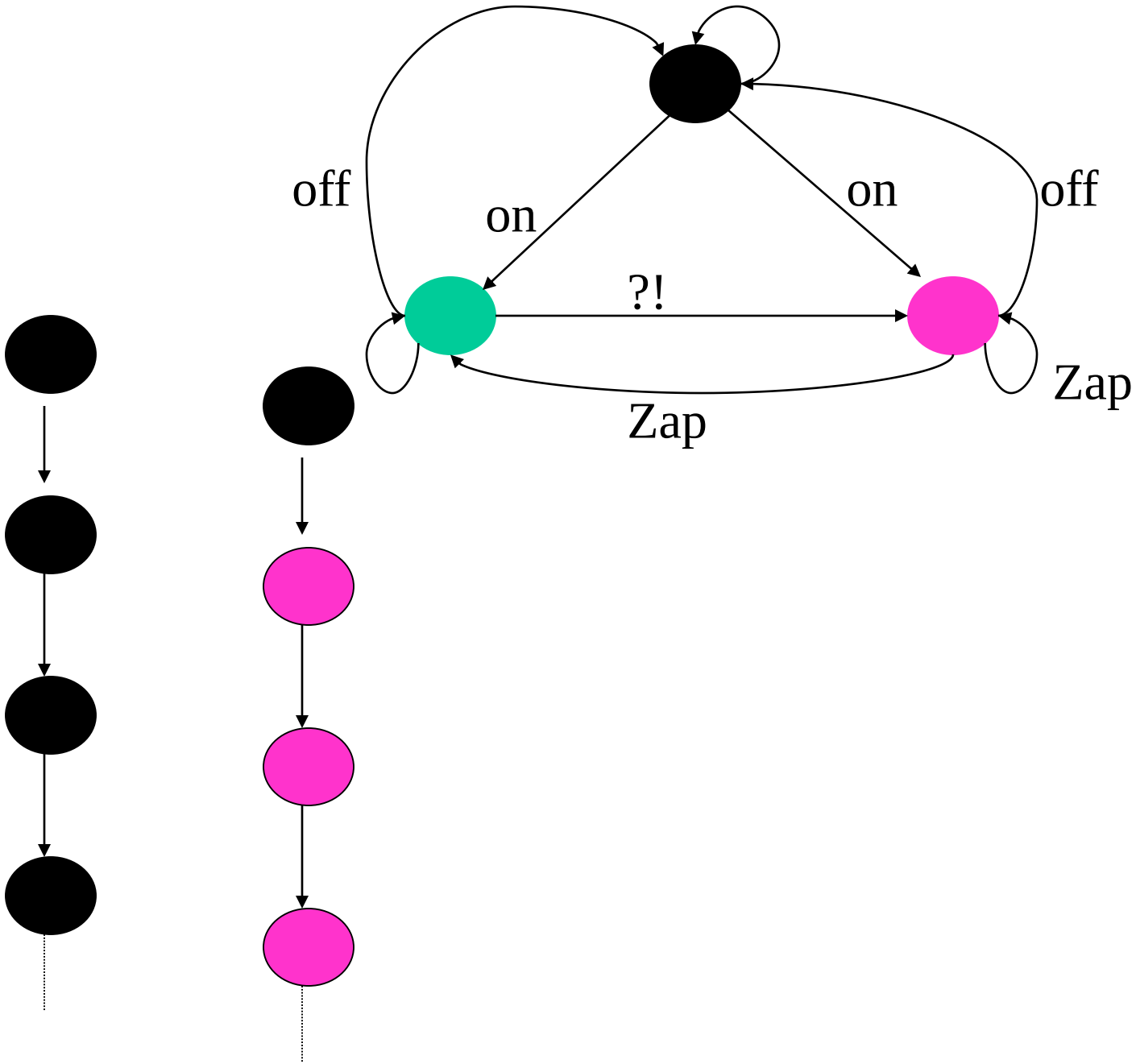
- **Linear Time**:
  - A ***computation*** at its first state satisfies a property.
  - Property ---- **LTL** formula
- **Branching Time**
  - The ***computation tree*** at its root satisfies a property.
  - Property: **CTL** (**CTL***, **μ-calculus**) formula.
  - **Computation Tree**
    - **All *computations*** starting from a state ***glued together*** (to form a tree structure).
- In branching time, ***the decisions*** taken during a run are taken into account.

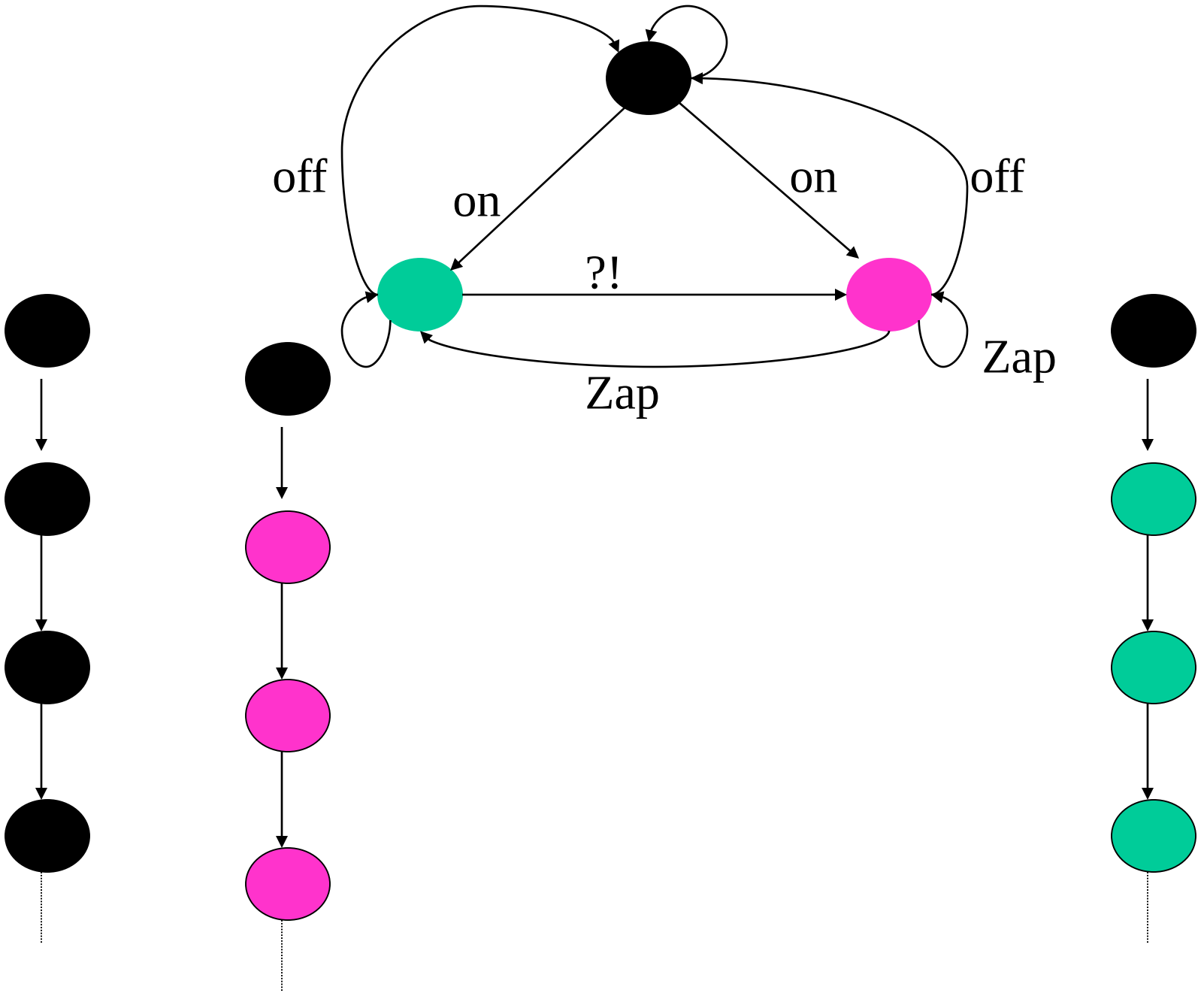


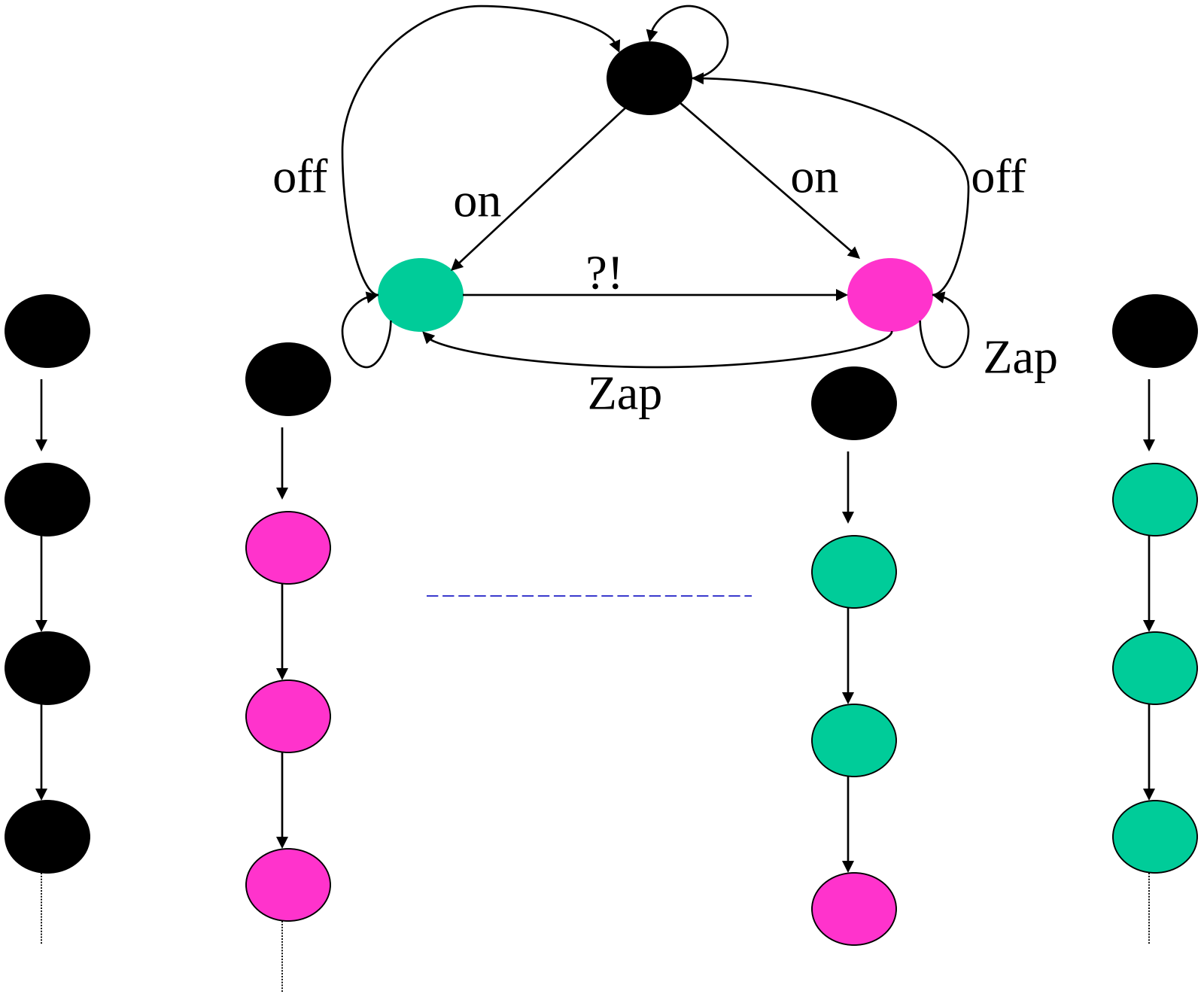
The TV Example



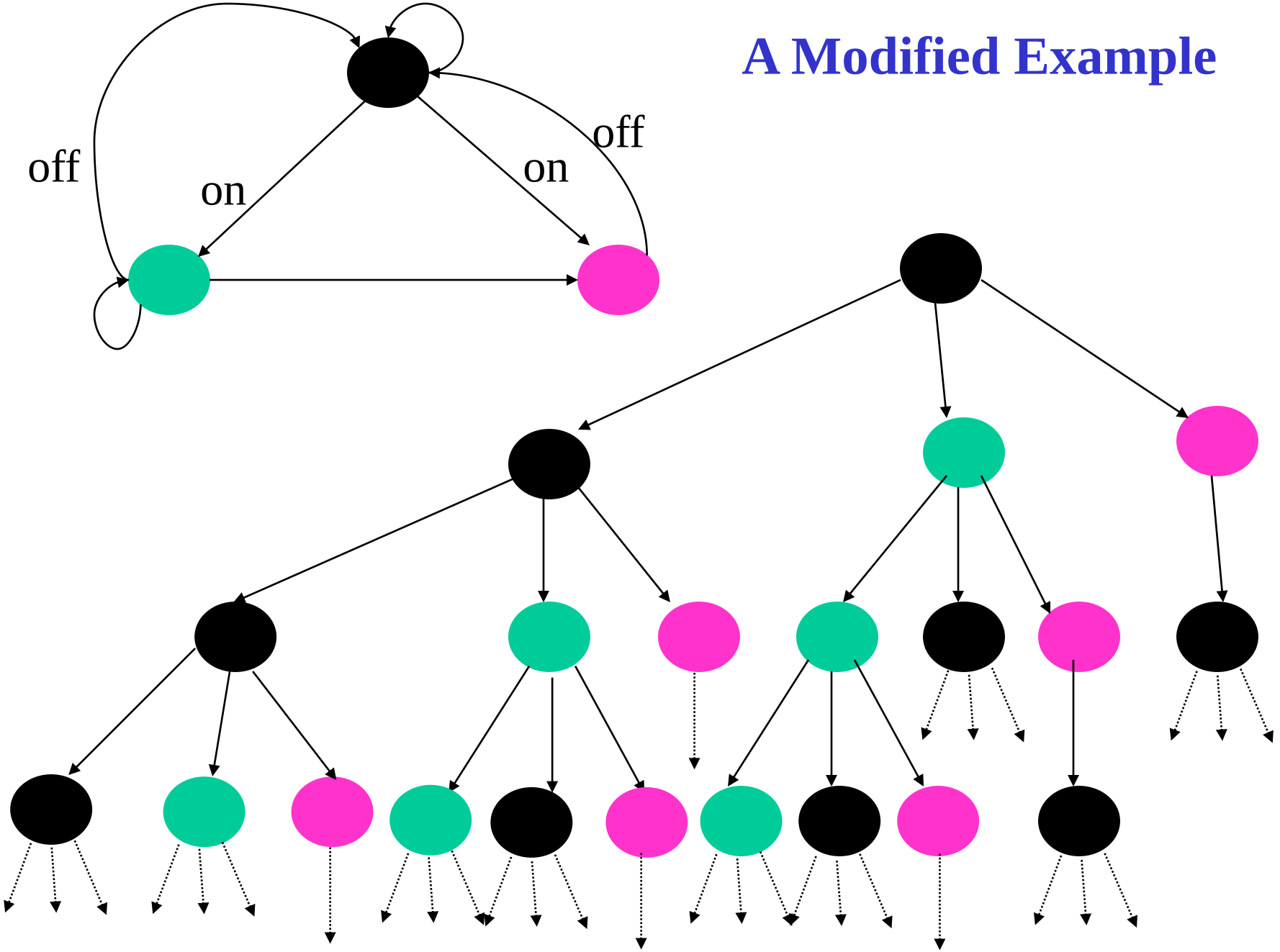

off
on
on
off
?!
Zap
Zap


off
on
on
off
?!
Zap
Zap

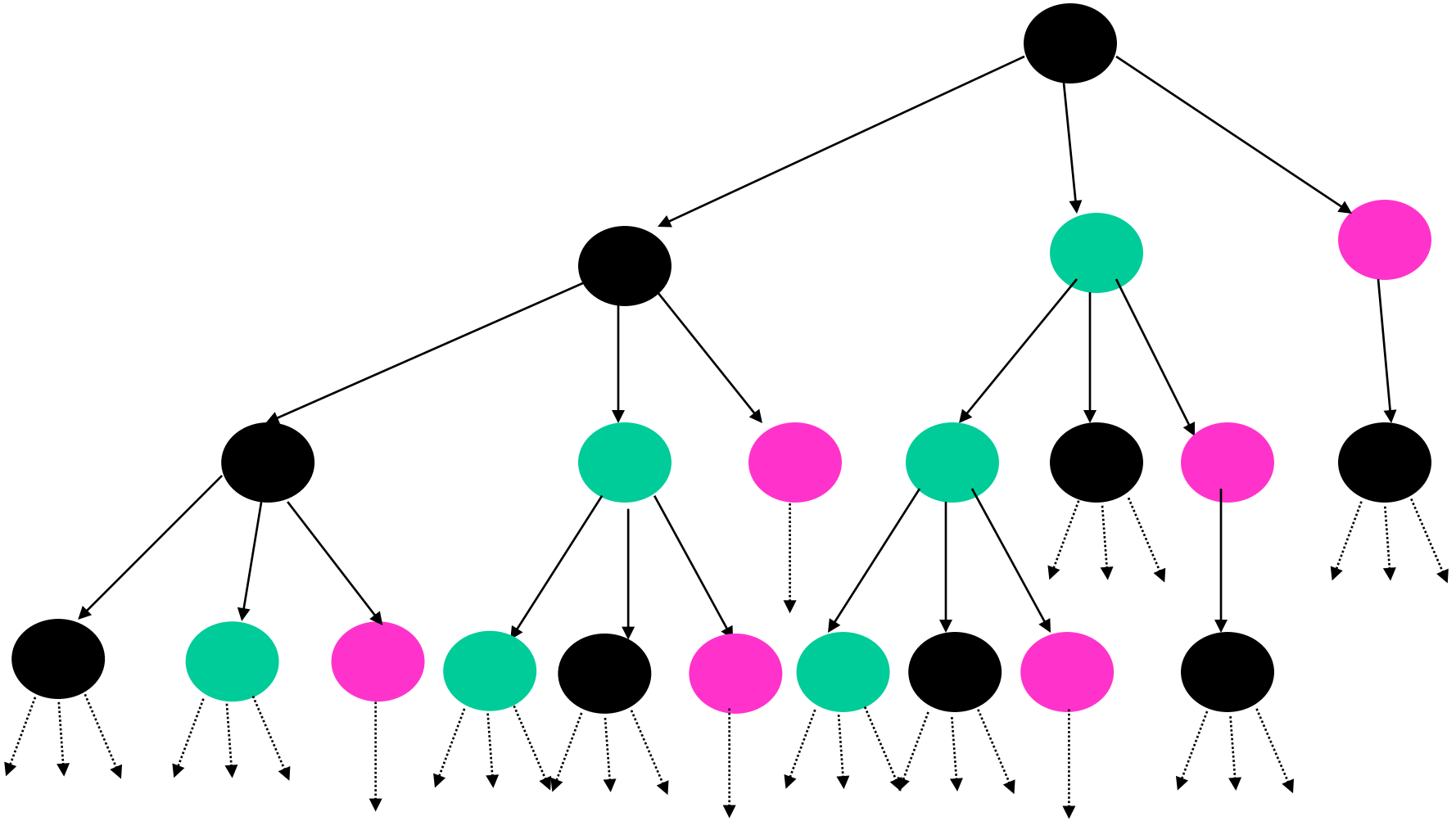

off
on
on
off
?!
Zap
Zap


off
on
on
off
?!
Zap
Zap

A Modified Example



For every path **π** and every state **s** on that path, there is a path **π'** starting from **s** and a state **s'** on **π'** which is **green**.



Branching Time Temporal Logic

- **$K = (S, S_0, R, AP, L)$**
- **$K, s \models \psi$** -- the computation tree rooted at **s** satisfies **ψ**.
- **$K \models \psi$** iff **$K, s_0 \models \psi$** for every **$s_0 \in S_0$**.
- **Branching Time Temporal Logics**:
  - **CTL**
  - **CTL***
  - (The modal) **μ-calculus**

Unwinding

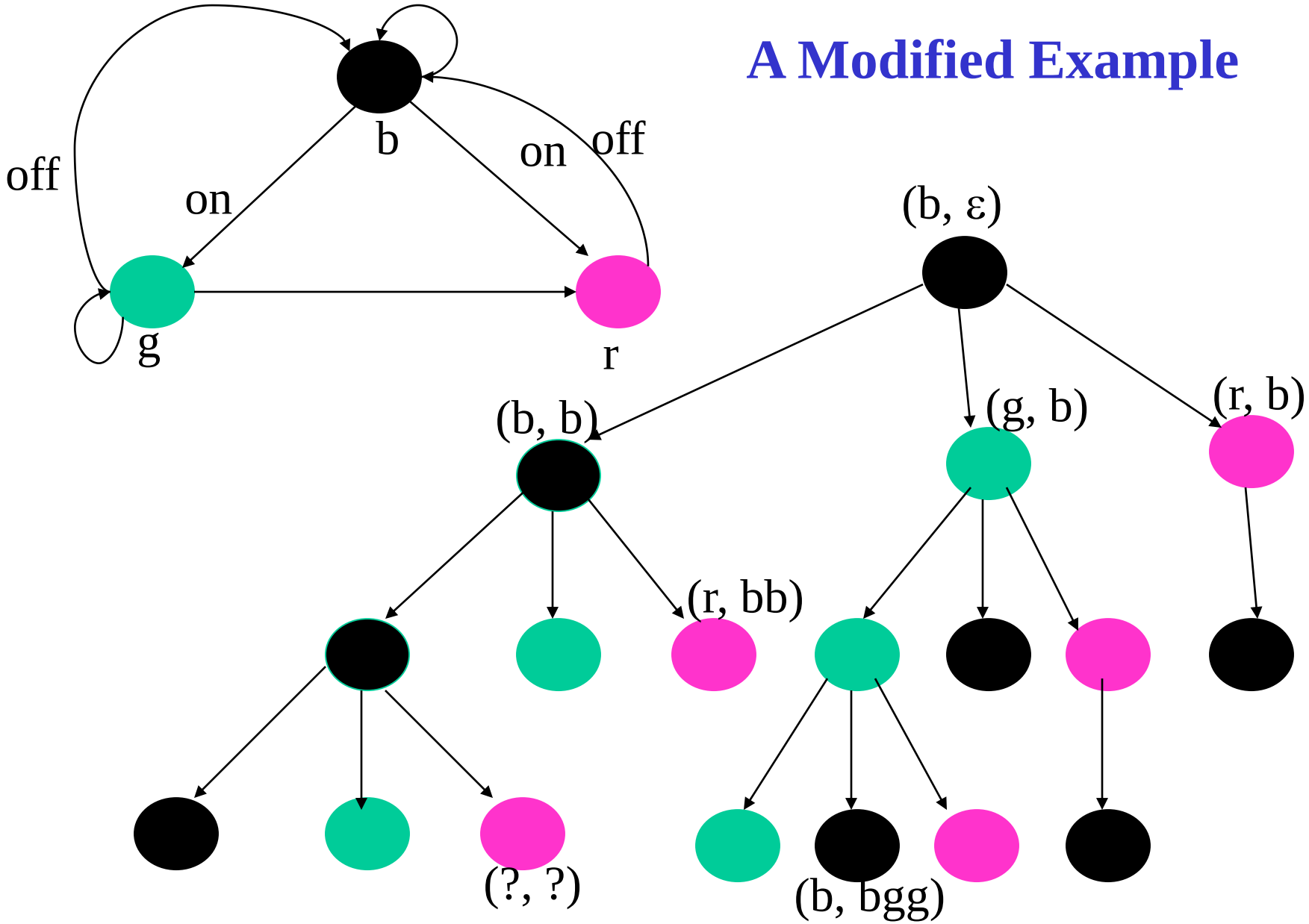
- **$K = (S, S_0, R, AP, L)$** $s_0 \in S$
- **$TR(K, s_0)$** --- The computation tree rooted at s_0.
- **$TR(K, s_0) = (\mathcal{S}_{s0}, (s_0,\varepsilon), \mathcal{R}_{s0}, AP, \mathcal{L}_{s0})$**
  - $(s_0, \varepsilon) \in \mathcal{S}_{s0}$;
  - If $(s_1, \sigma) \in \mathcal{S}_{s0}$ and $R(s_1, s_2)$ then
    - $(s_2, \sigma.s_1) \in \mathcal{S}_{s0}$ and
    - $\mathcal{R}_{s0}((s_1, \sigma), (s_2, \sigma.s_1))$;
    - $\mathcal{L}((s_1, \sigma)) = L(s_1)$.

Therefore, for all $(s, \sigma) \in \mathcal{S}_{s0}$, $s \in S$ and $\sigma = s_0\, s_1 \ldots s_n$ is a path in **K** from s_0 to s_n and $R(s_n, s)$ (hence, $\sigma.s$ is a path in **K** from s_0 to s)

Unwinding

- **TR(K, s)** is almost a Kripke structure.
  - $\mathcal{S}_s$ will tipically be infinite.
  - But $\mathcal{R}_s$ is ***tree-like***.
  - The "***graph***" of **TR(K, s)** is a tree rooted at **(s, ε)**.
- **TR(K, s)** is the ***computation tree rooted*** of **K** at **s**.

A Modified Example



Linear time Vs Branching time

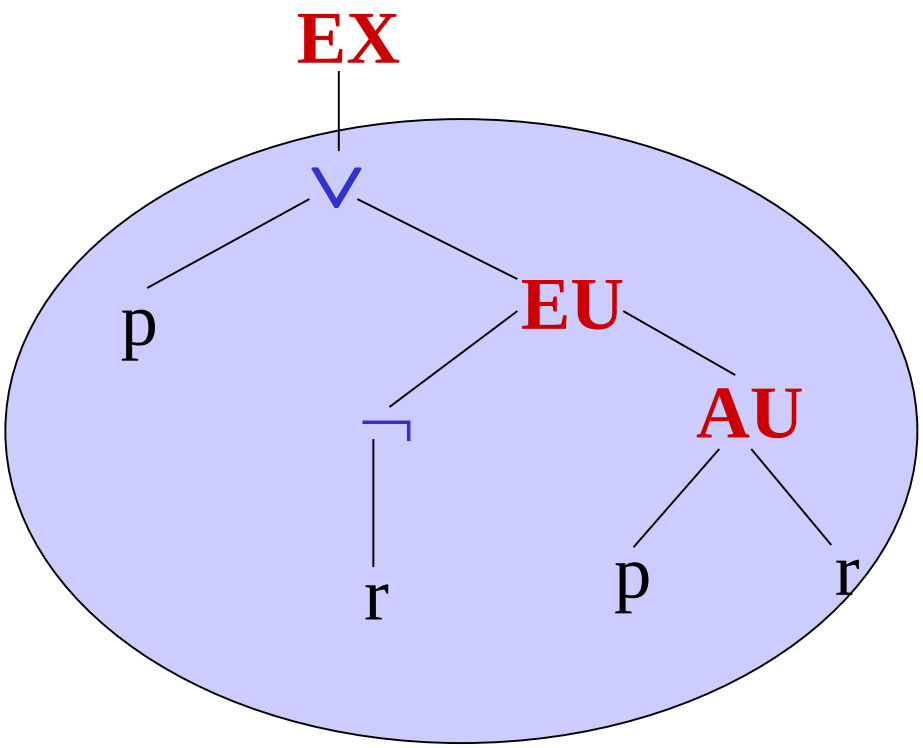
- There are ***properties*** that can be ***expressed in* LTL** but which ***can not be expressed in* CTL**.
- There are ***properties*** that can be ***expressed in* CTL** but ***not in* LTL**.
- The **LTL *model checking problem*** can be ***converted*** into a ***restricted*** kind of a **CTL*** ***model checking problem***.

CTL

- **Syntax**
  - **AP** – a finite set of ***atomic propositions***.
  - **$p \in AP$** is a formula.
  - If ψ and ψ' are formulas then so are $\neg\psi$ and $\psi \vee \psi'$.
  - If ψ is a formula then so is **EX**ψ
  - If ψ_1 and ψ_2 are formulas then so are **EU**(ψ_1, ψ_2) and **AU**(ψ_1, ψ_2).

Formulas

- **EX(p ∨ EU(¬r, AU(p, r)))**



Semantics

- **$K = (S, S_0, R, AP, L)$**
  - **$L : S \longrightarrow 2^{AP}$**
- **ψ** a **CTL** formula and **$s \in S$**
- **$K, s \models \psi$**
- **ψ** (holds) ***is satisfied*** at **s**.
- **FACT**:

$K, s \models \psi$ iff **$TR(K, s), (s, \varepsilon) \models \psi$.**

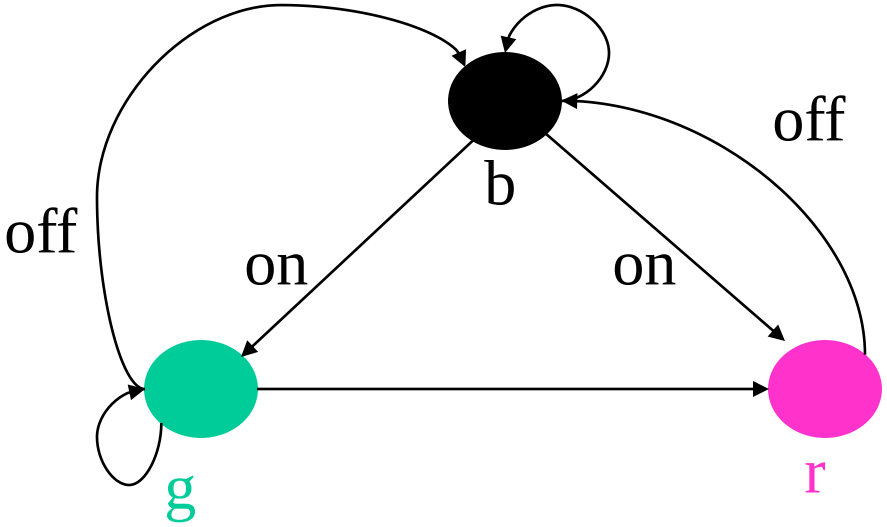
Semantics

- **CTL ::= p | $\neg\psi$ | $\psi_1 \vee \psi_2$ | EX(ψ) |**
  | EU(ψ_1, ψ_2) | AU(ψ_1, ψ_2)
- **K = (S, S_0, R, AP, L) ; L: S $\longrightarrow$ 2^{AP} ; s $\in$ S**
- **K, s $\models$ p** iff **p $\in$ L(s).**
- **K, s $\models$ $\neg\psi$** iff **not K, s $\models$ ψ**
- **K, s $\models$ $\psi_1 \vee \psi_2$** iff
  K, s $\models$ ψ_1 or **K, s $\models$ ψ_2.**

Semantics

- **CTL ::= p | $\neg\psi$ | $\psi_1 \vee \psi_2$ | EX(ψ) |**
  | EU(ψ_1, ψ_2) | AU(ψ_1, ψ_2)
- **K = (S, S_0, R, AP, L) ; L: S $\longrightarrow$ 2^{AP} ; s $\in$ S**
- **K, s $\models$ EX(ψ)** iff there exists **s'** such that:
  - **s $\longrightarrow$ s'** (i.e. **R(s, s')**) and **K, s' $\models$ ψ**

  s has a successor state **s'** at which ψ holds.



AP = {n, h, uh}

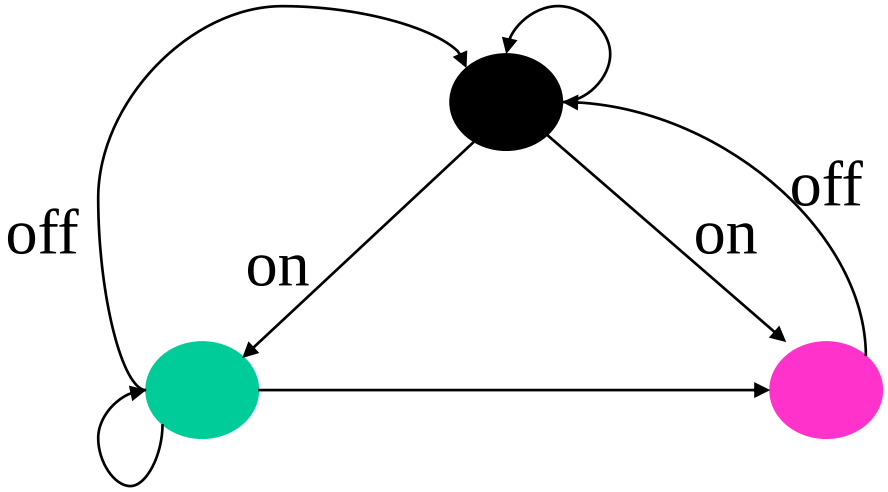
K, b ⊨ EX(uh) ? K, b ⊨ EX(¬uh) ?

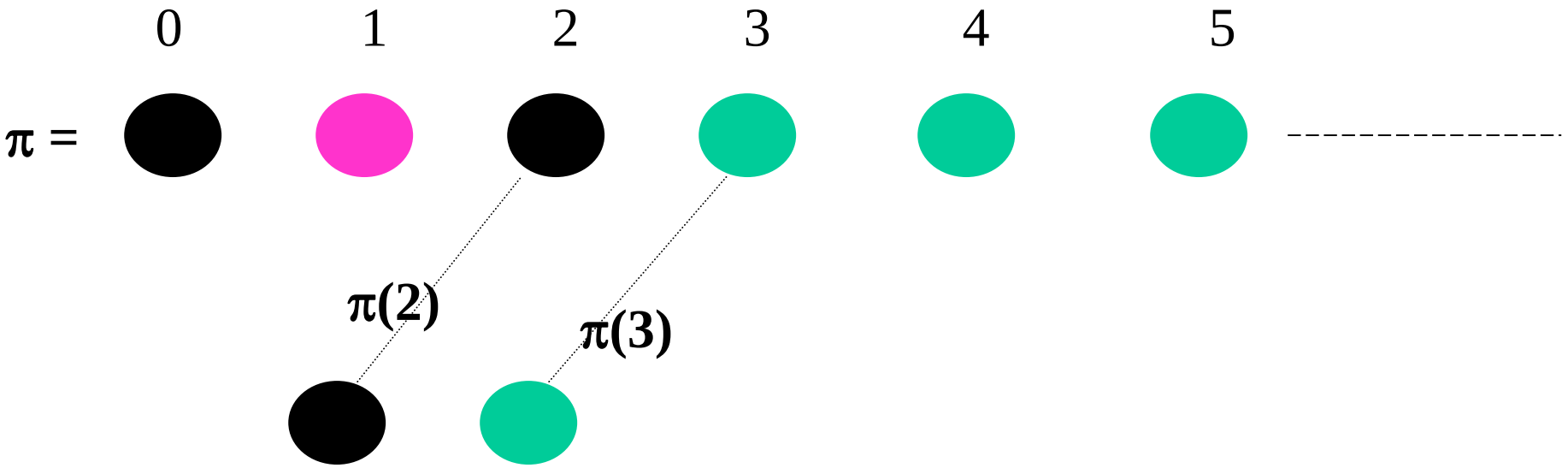
K, g ⊨ EX(uh) ?

K, r ⊨ EX(h) ?

Semantics

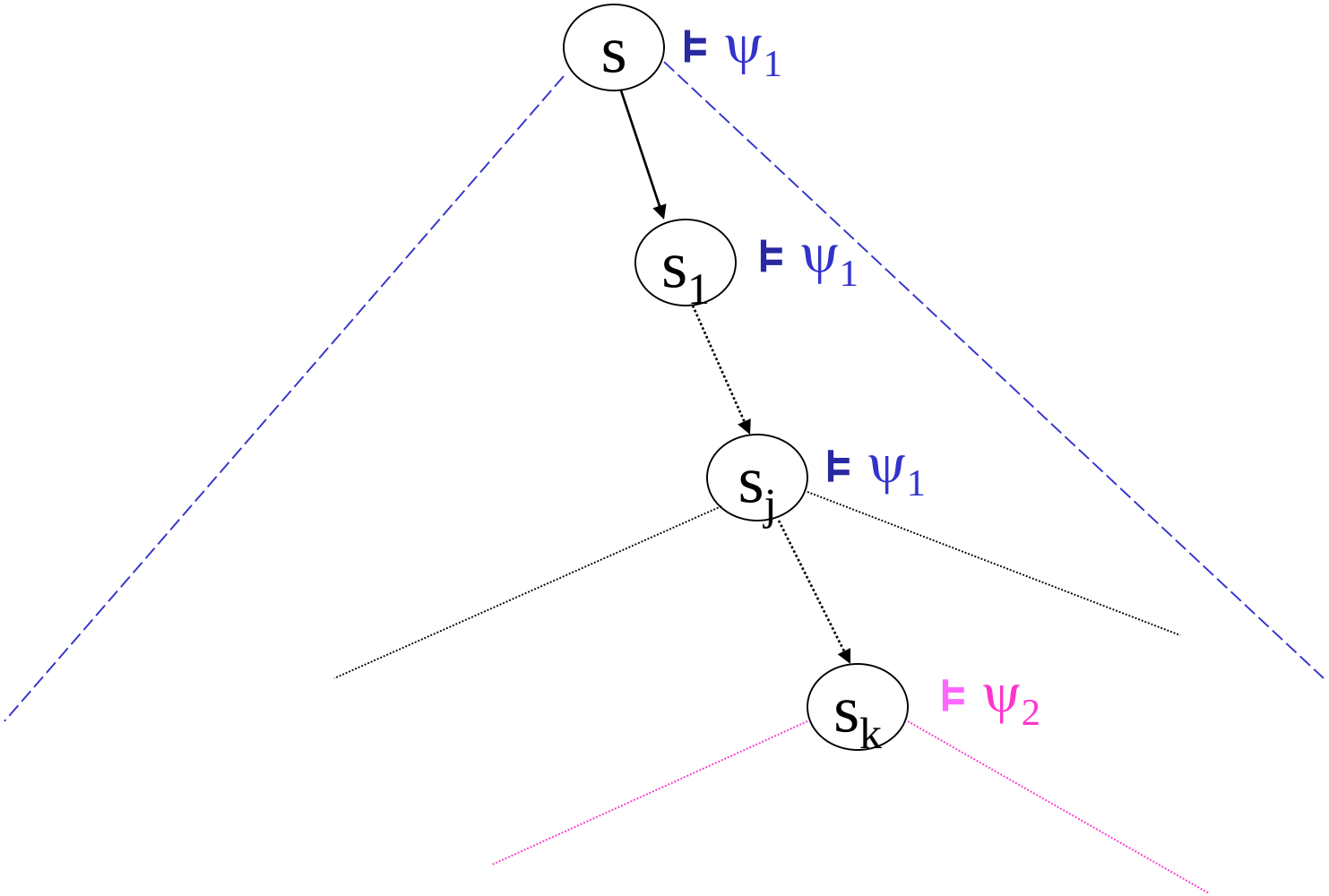
- $K = (S, S_0, R, AP, L)$; $L: S \longrightarrow 2^{AP}$; $s \in S$
- ***A path from s*** is a (infinite) sequence of states
  $\pi = s_0, s_1, s_2, \ldots, s_i, s_{i+1}, \ldots$ s.t:
  - $s = s_0$
  - $s_i \longrightarrow s_{i+1}$ (i.e. $R(s_i, s_{i+1})$) for every i.
- $\pi(i) = s_i$ the i-th element of π.


off
on
on
off


0
1
2
3
4
5
π =
π(2)
π(3)

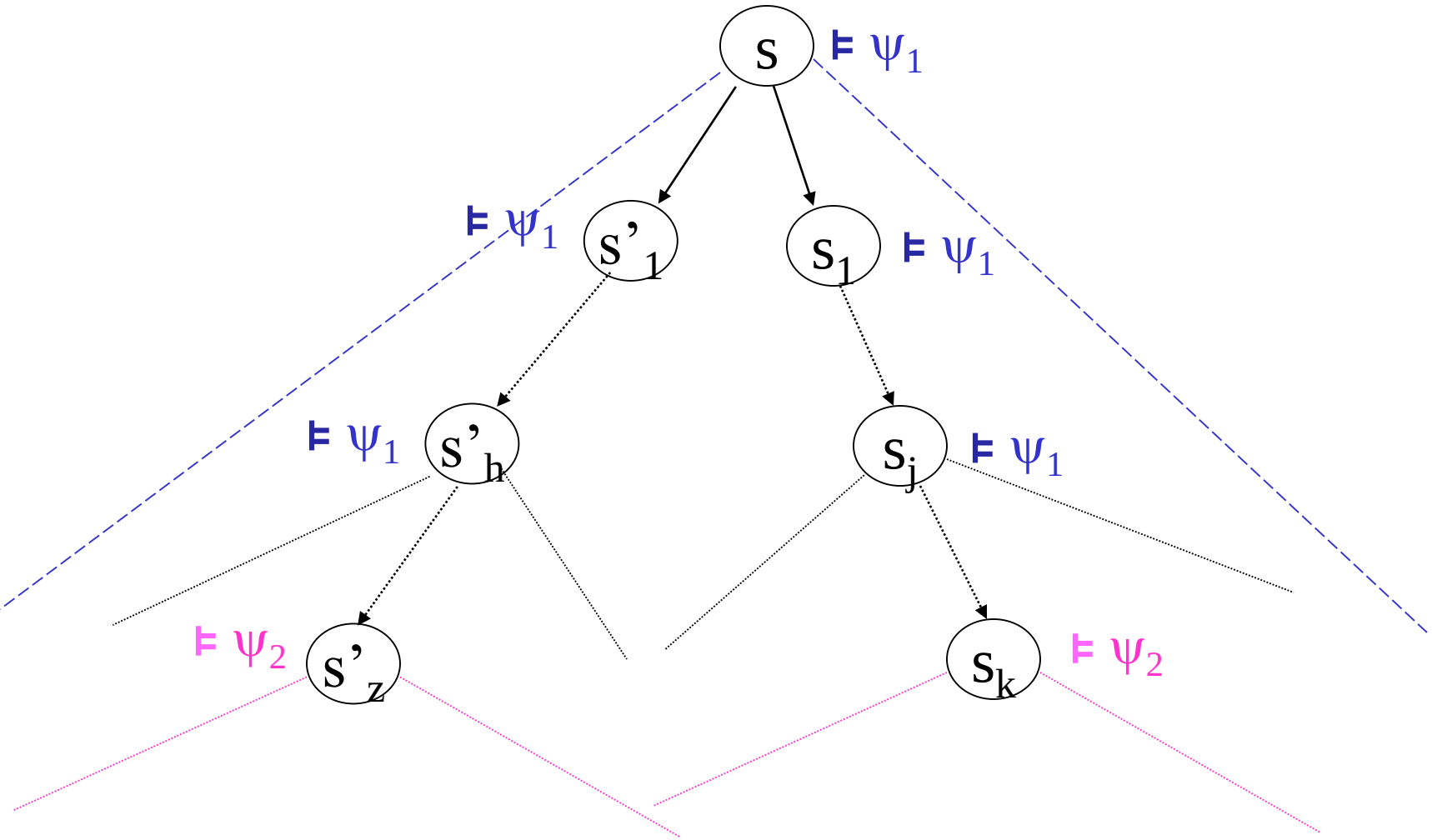
Semantics

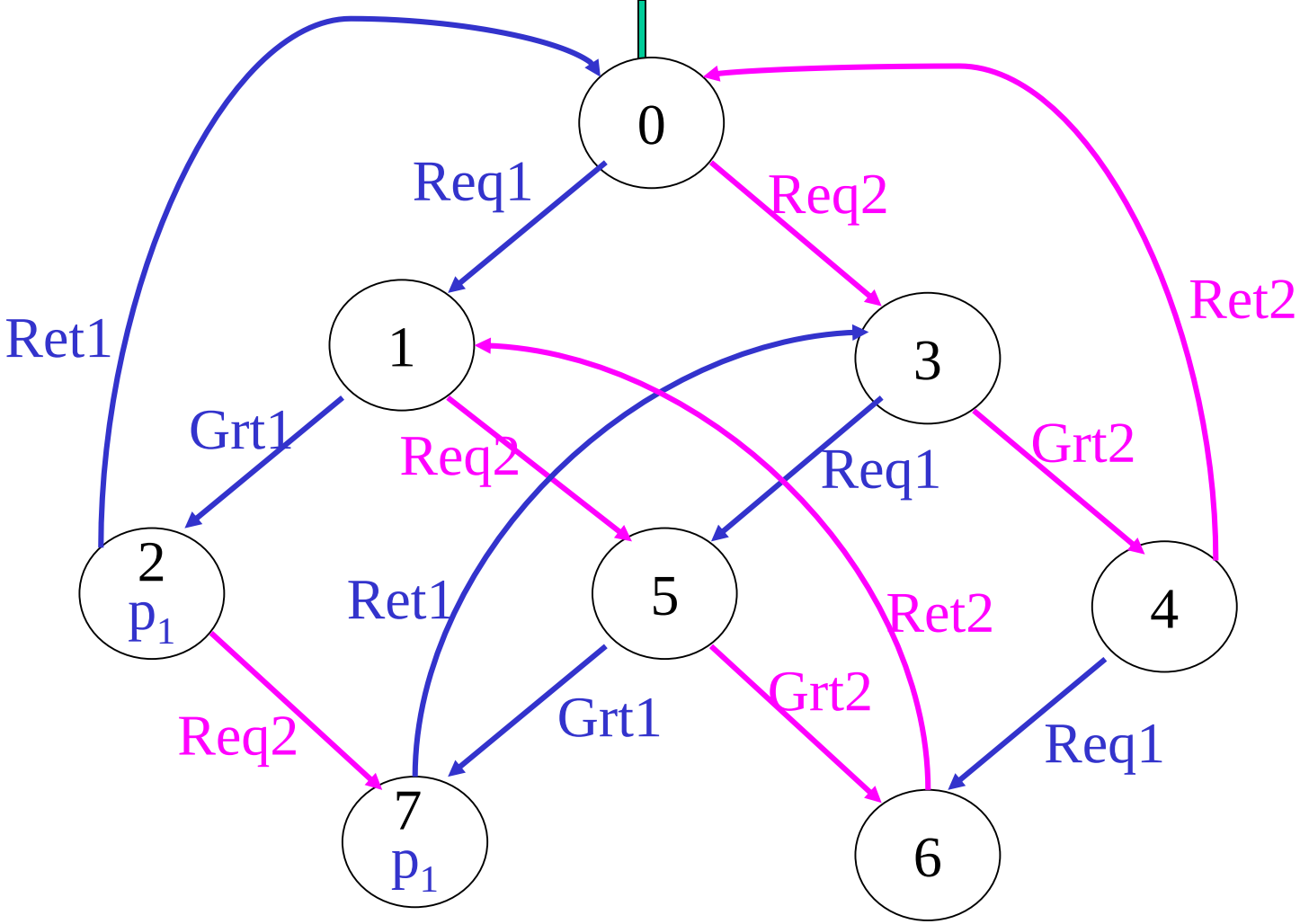
- **CTL ::= $p \mid \neg\psi \mid \psi_1 \vee \psi_2 \mid EX(\psi) \mid$**
  $\mid EU(\psi_1, \psi_2) \mid AU(\psi_1, \psi_2)$
- **$K = (S, S_0, R, AP, L)$; $L: S \longrightarrow 2^{AP}$; $s \in S$**
- **$K, s \vDash EU(\psi_1, \psi_2)$** iff ***there exists a path*** **$\pi = s_0, s_1, \ldots$** from **s** (i.e. **s_0=s**) and **$k \geq 0$** such that:
  - **$K, \pi(k) \vDash \psi_2$**
  - **$K, \pi(j) \vDash \psi_1$** , for all **$0 \leq j < k$**.


s
⊨ ψ1
s1
⊨ ψ1
sj
⊨ ψ1
sk
⊨ ψ2

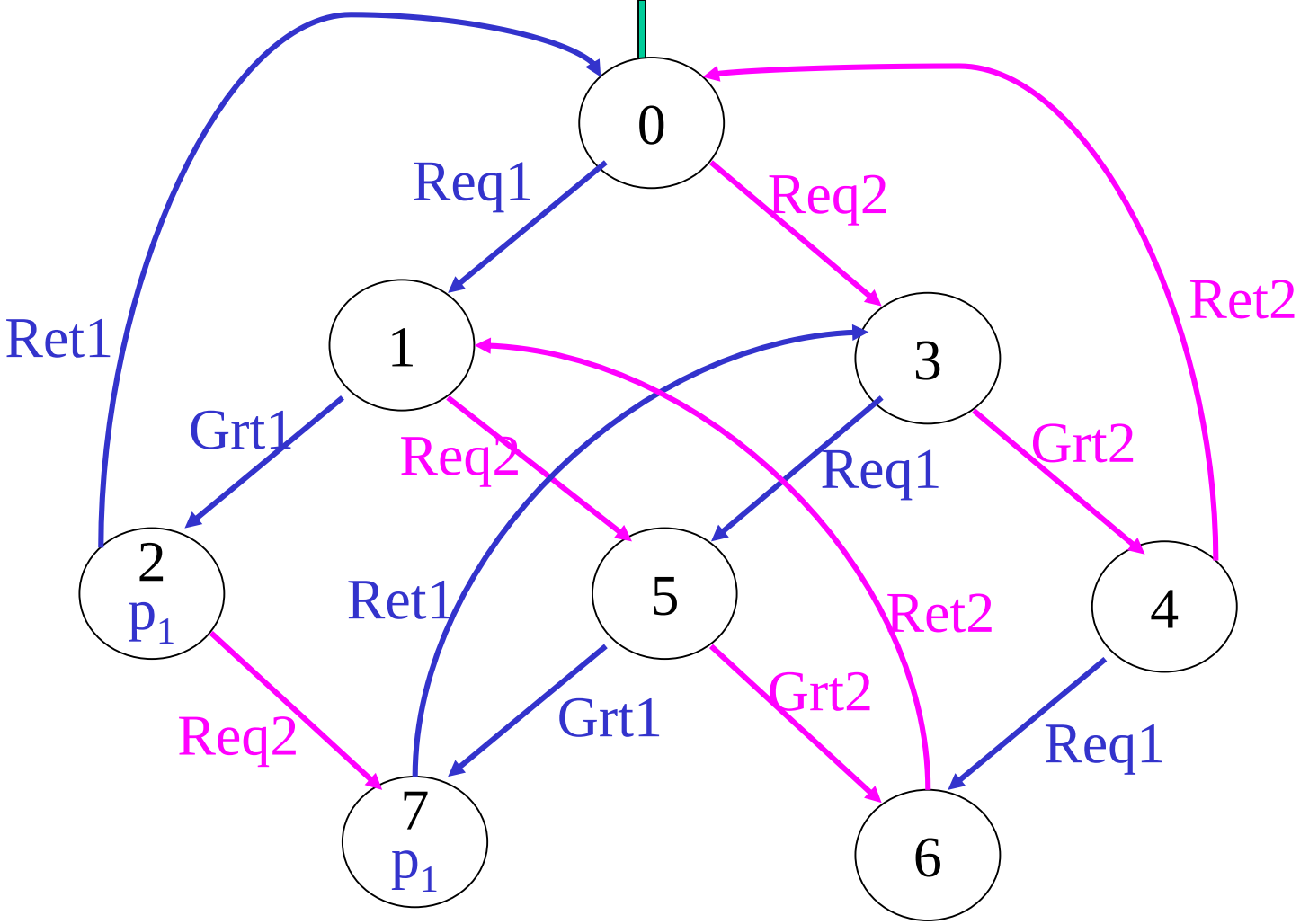
Semantics

- **CTL ::= $p \mid \neg\psi \mid \psi_1 \vee \psi_2 \mid EX(\psi) \mid$**
  $\mid EU(\psi_1, \psi_2) \mid AU(\psi_1, \psi_2)$
- **$K = (S, S_0, R, AP, L)$; $L: S \longrightarrow 2^{AP}$; $s \in S$**
- **$K, s \vDash AU(\psi_1, \psi_2)$** iff ***for every path***
  $\pi = s_0, s_1, \ldots$ from **s** there exists **$k \geq 0$** such that:
  - **$K, \pi(k) \vDash \psi_2$**
  - **$K, \pi(j) \vDash \psi_1$**, for all **$0 \leq j < k$**.

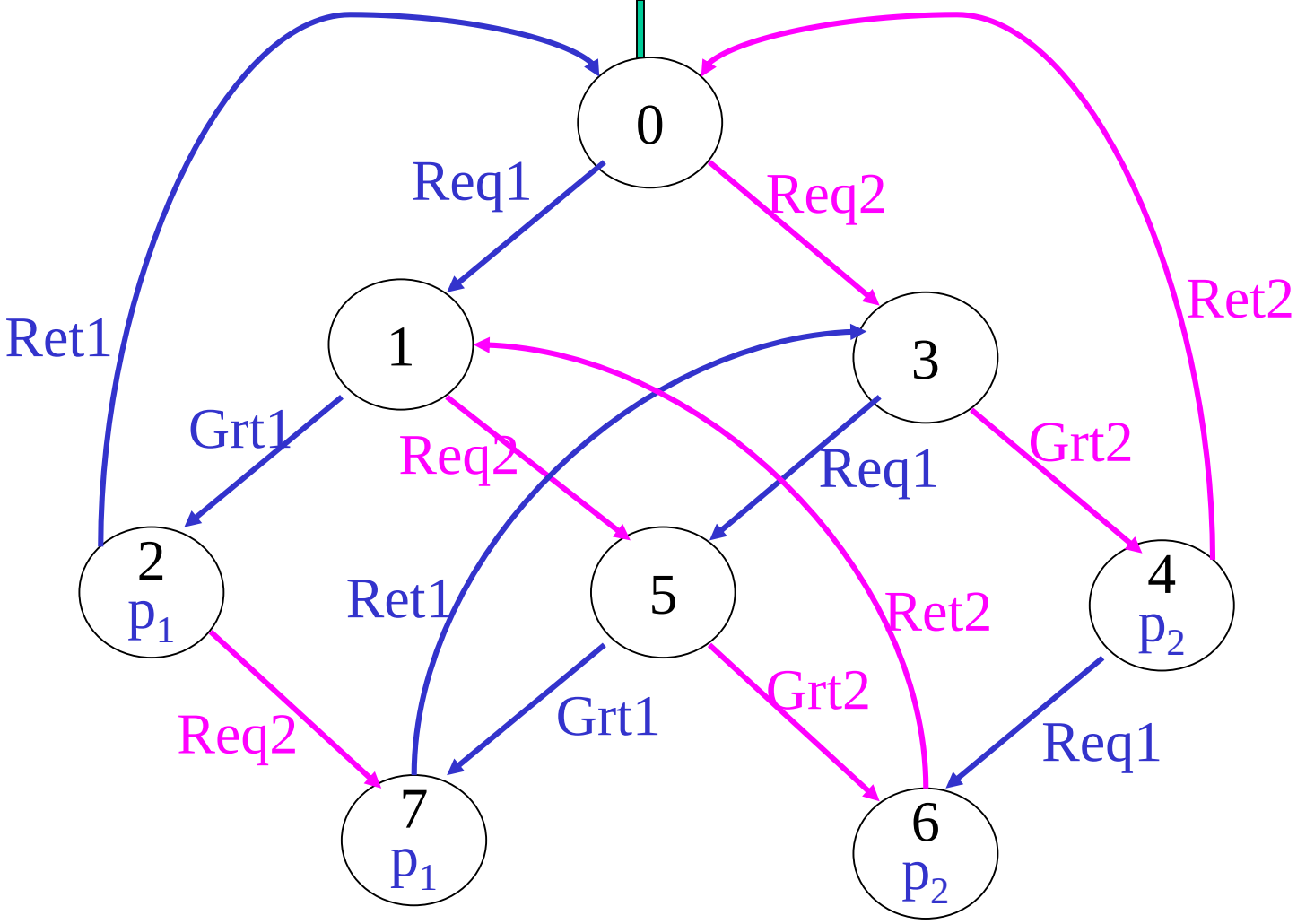

s
⊨ ψ1
⊨ ψ1
s'1
s1
⊨ ψ1
⊨ ψ1
s'h
sj
⊨ ψ1
⊨ ψ2
s'z
sk
⊨ ψ2


0
Req1
Req2
Ret1
Ret2
1
3
Grt1
Req2
Req1
Grt2
2
p1
Ret1
5
Ret2
4
Req2
Grt1
Grt2
Req1
7
p1
6

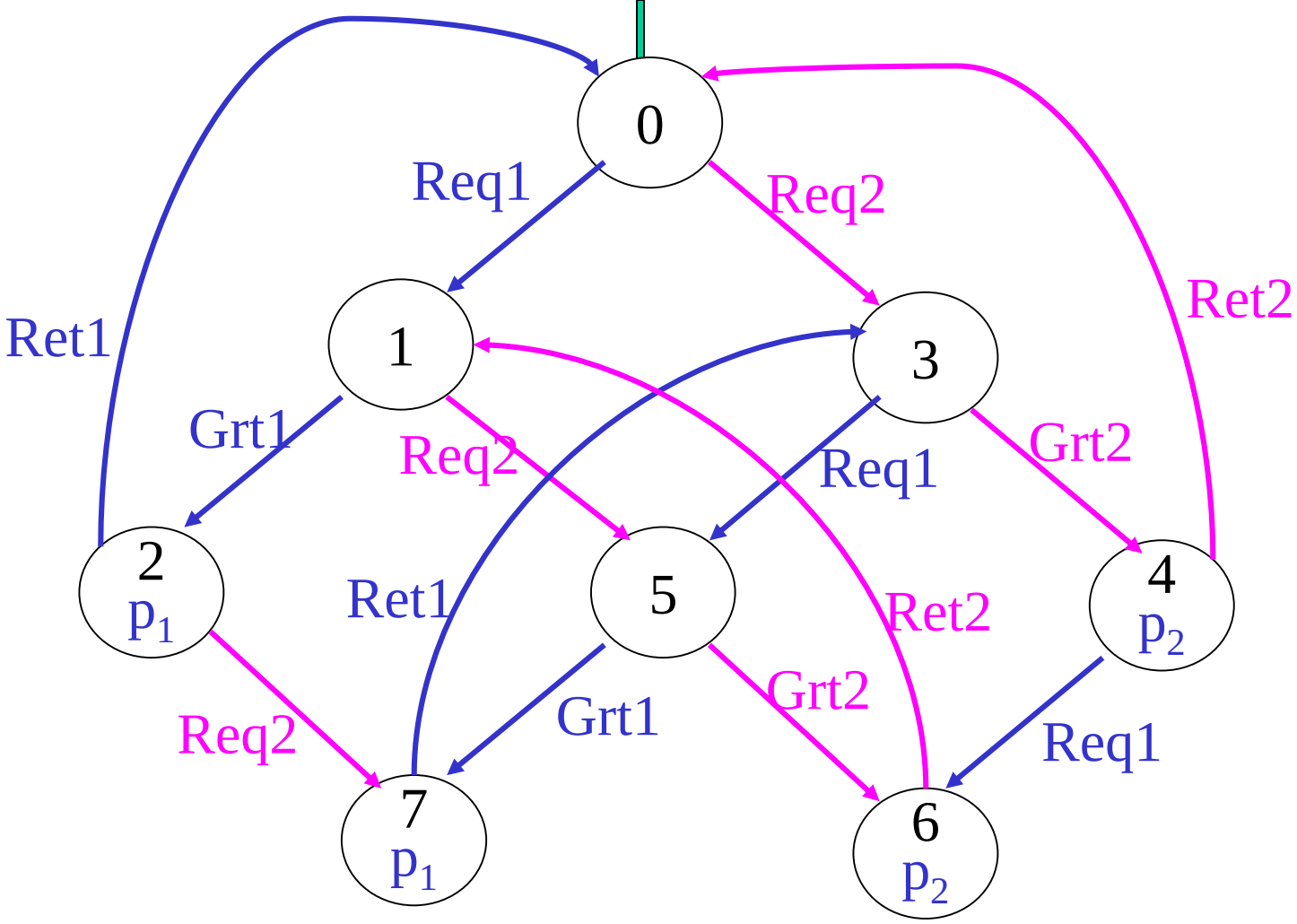
M, 0 ⊨ EU(⊤, p_1) ?


0
Req1
Req2
Ret1
Ret2
1
3
Grt1
Req2
Req1
Grt2
2
p_1
Ret1
5
Ret2
4
Req2
Grt1
Grt2
Req1
7
p_1
6

M, 0 ⊨ AU(⊤, p_1) ?

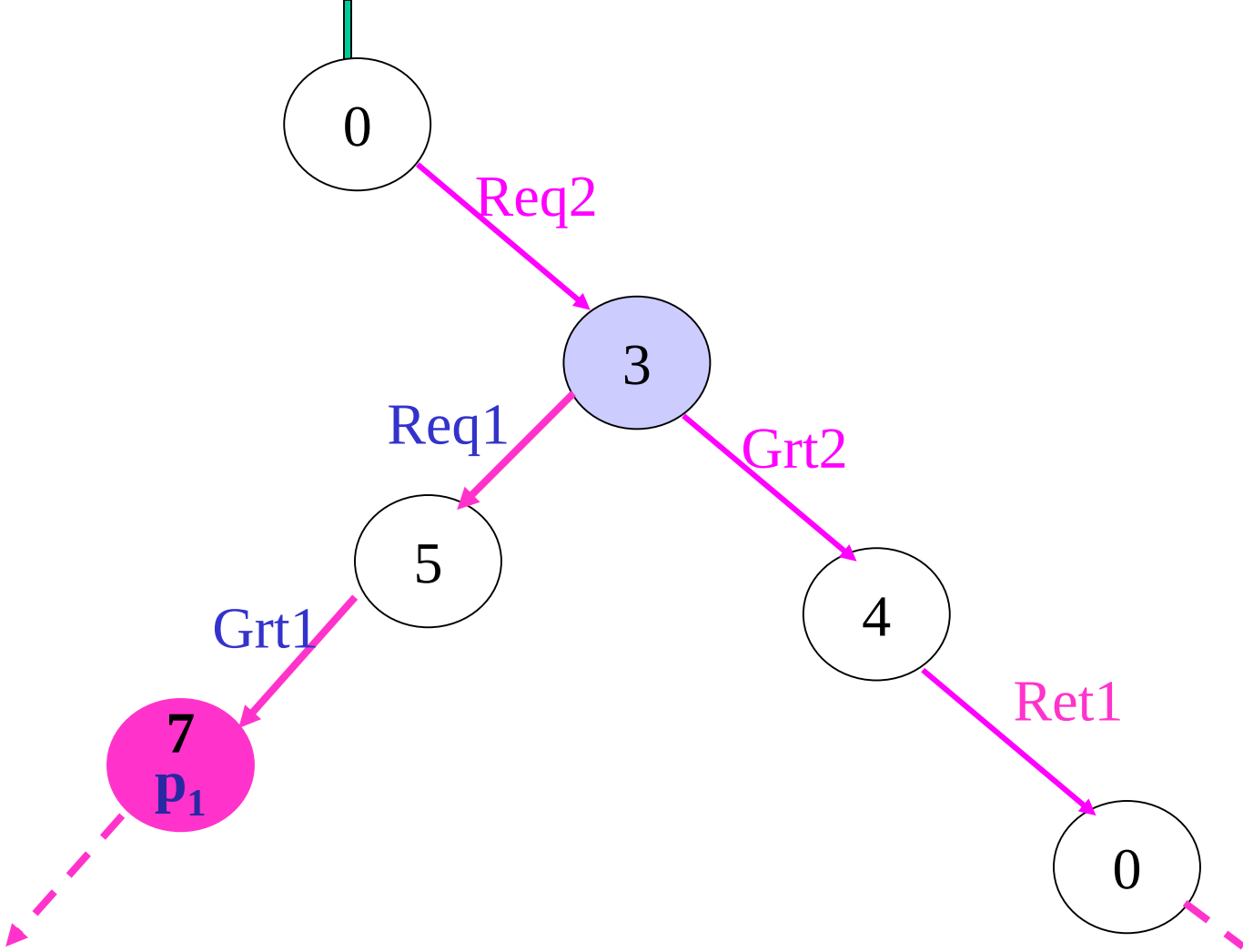


M, 0 ⊨ AU(⊤, $p_1 \vee p_2$) ?



M, 0 ⊨ AU(⊤, EU(⊤, p_1)) ?

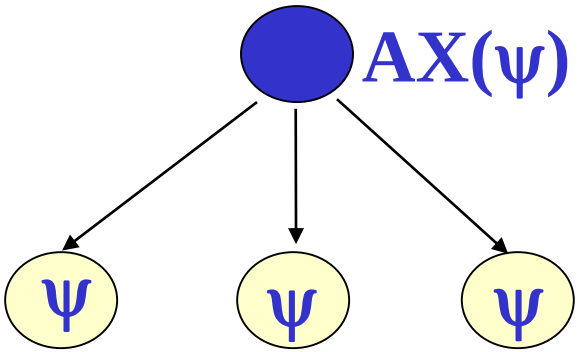
From s_0, ***all*** the computations will reach a point, where it is ***possible*** for 1 to print ***eventually.***



M, 0 ⊨ AU(⊤, EU(⊤, p_1)) ?

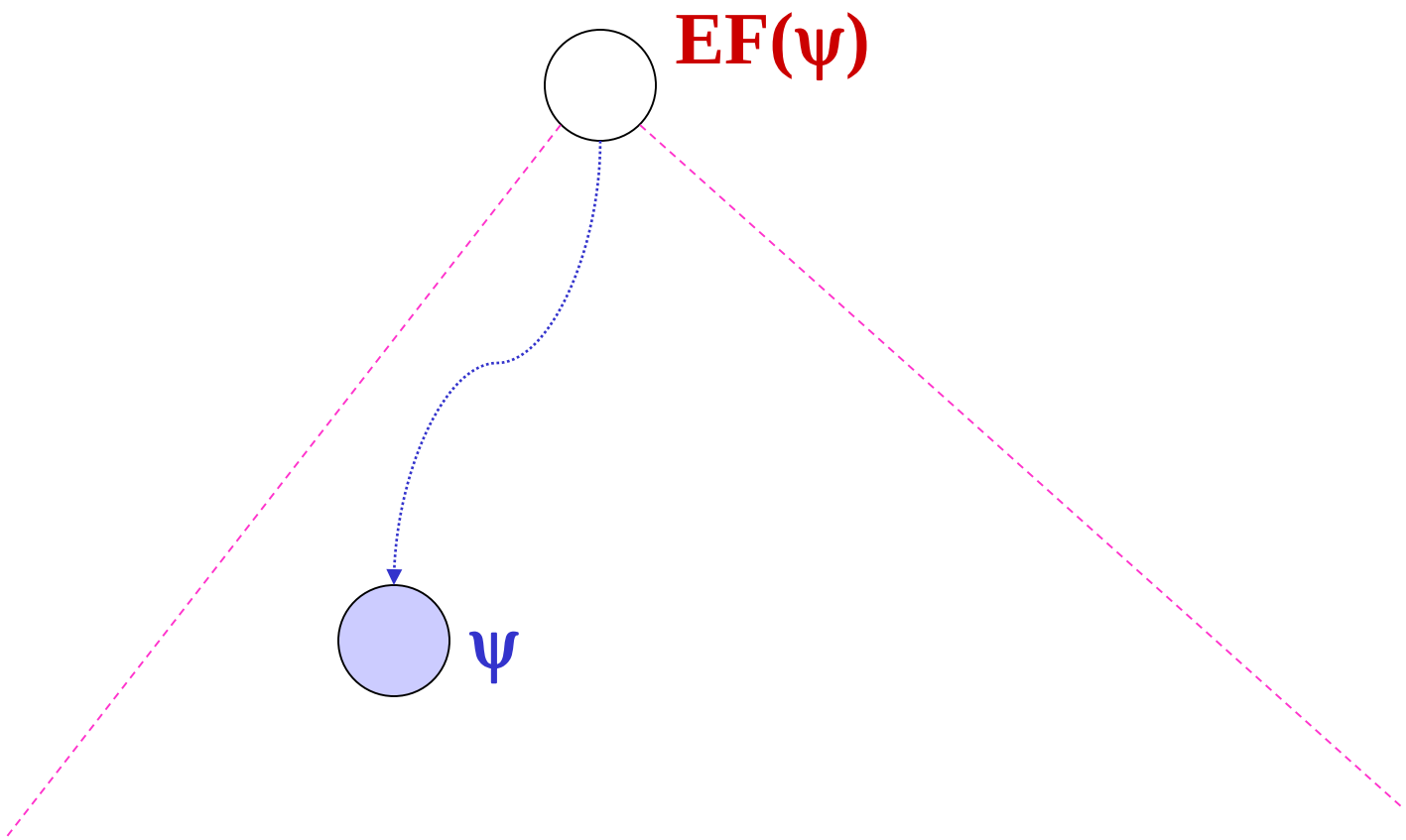
Derived Operators

- **AX(ψ) = ¬EX(¬ψ)**
  - It is not the case there exists a next state at which ψ does not hold, equivalent to
  - ***For every next state*** ψ holds.



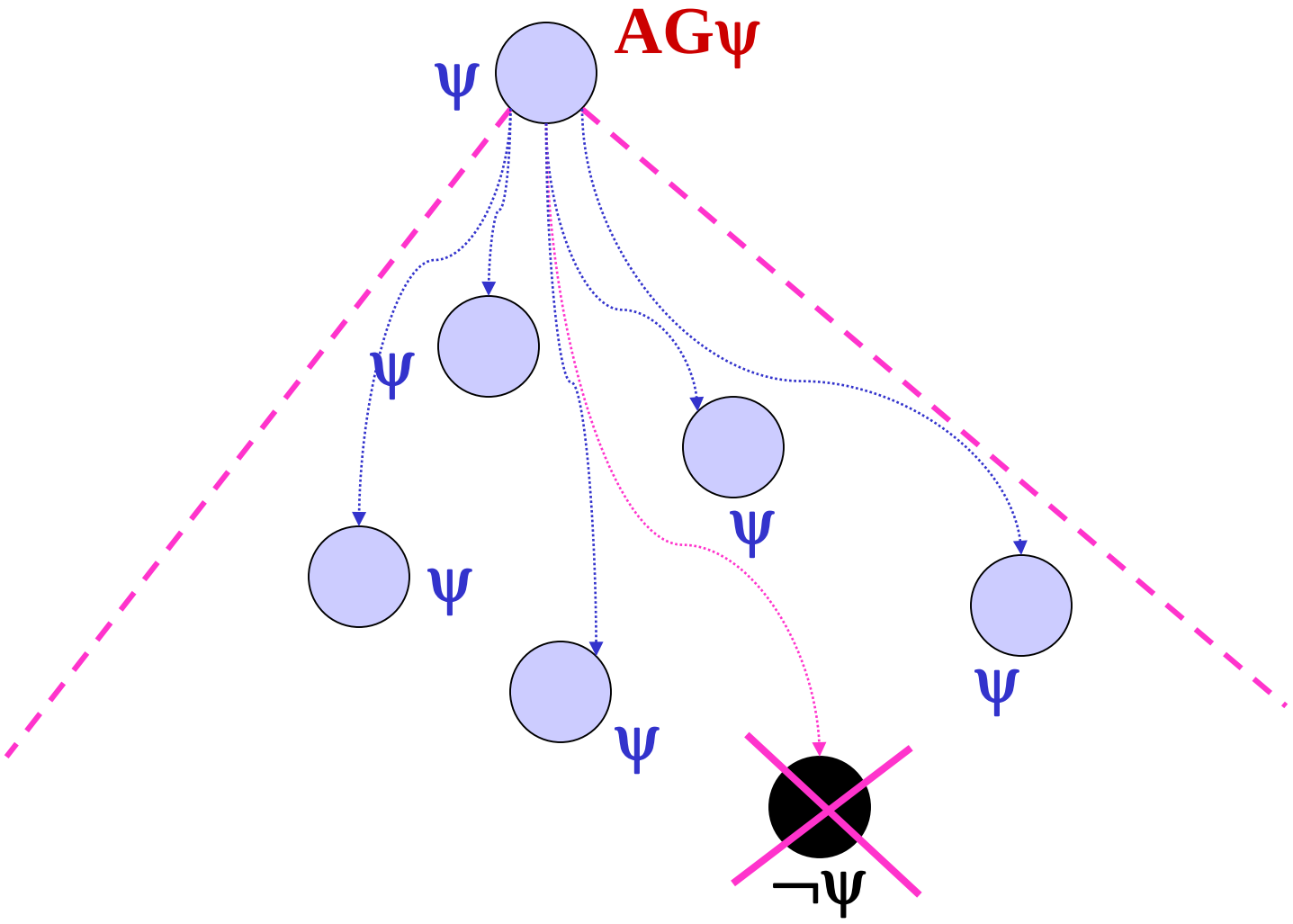
Derived Operators

- **K, s ⊨ EF(ψ)**
- **EF(ψ) = EU(⊤, ψ)**
  - There exists a path **π** (from **s**) and **k ≥ 0** such that:
    - **K, π(k) ⊨ ψ.**


EF(ψ)
ψ

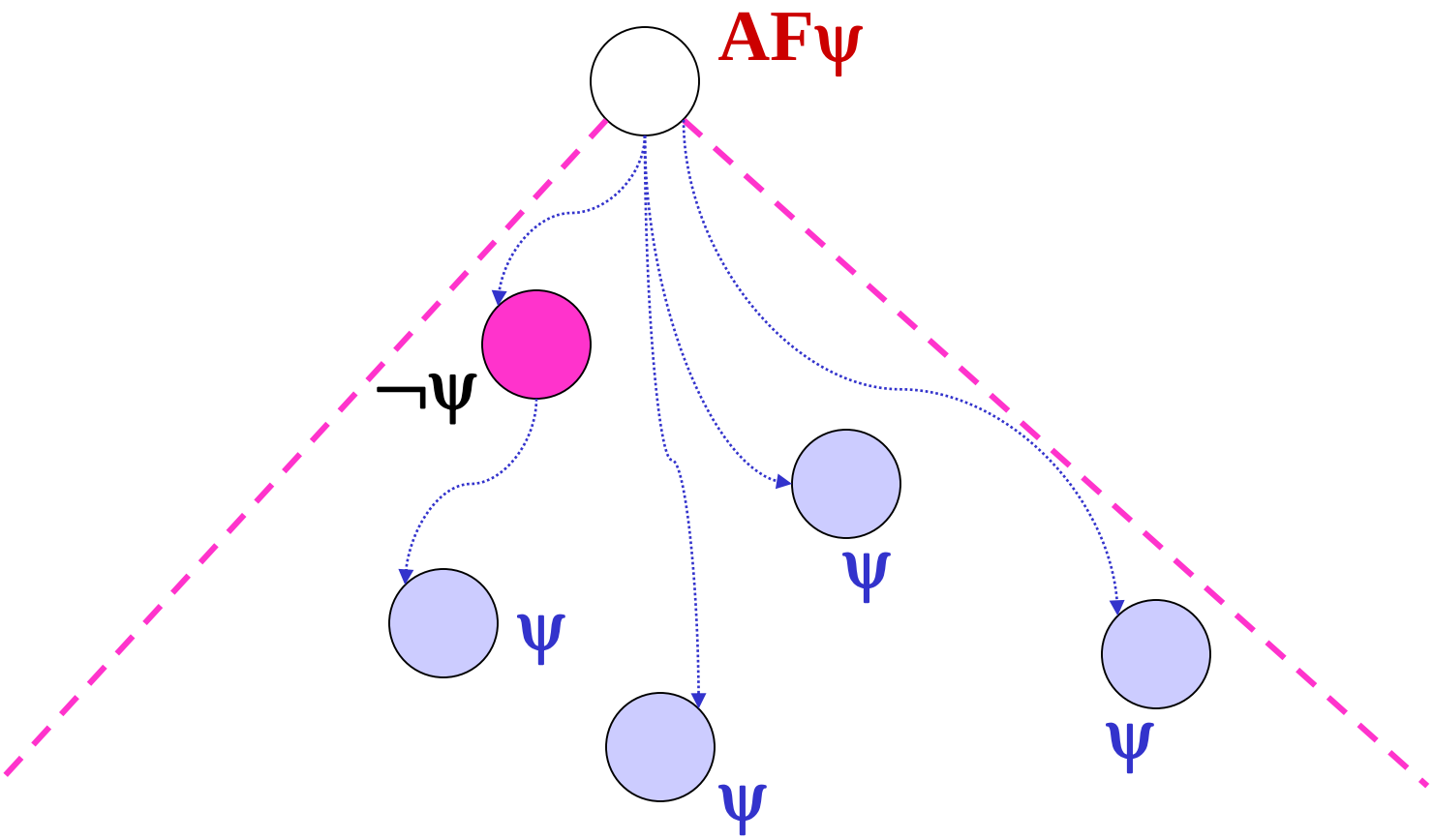
Derived Operators

- **K, s ⊨ AG(ψ)**
- **AG(ψ) = ¬EF(¬ψ)**
  - It is ***not*** the case ***there exists a path π*** (from **s**) and **k ≥ 0** such that:
    - **K, π(k) ⊨ ¬ ψ**
  - ***For every path π*** (from **s**) and ***every*** **k ≥ 0**:
    - **K, π(k) ⊨ ψ**


AGψ
ψ
ψ
ψ
ψ
ψ
ψ
¬ψ

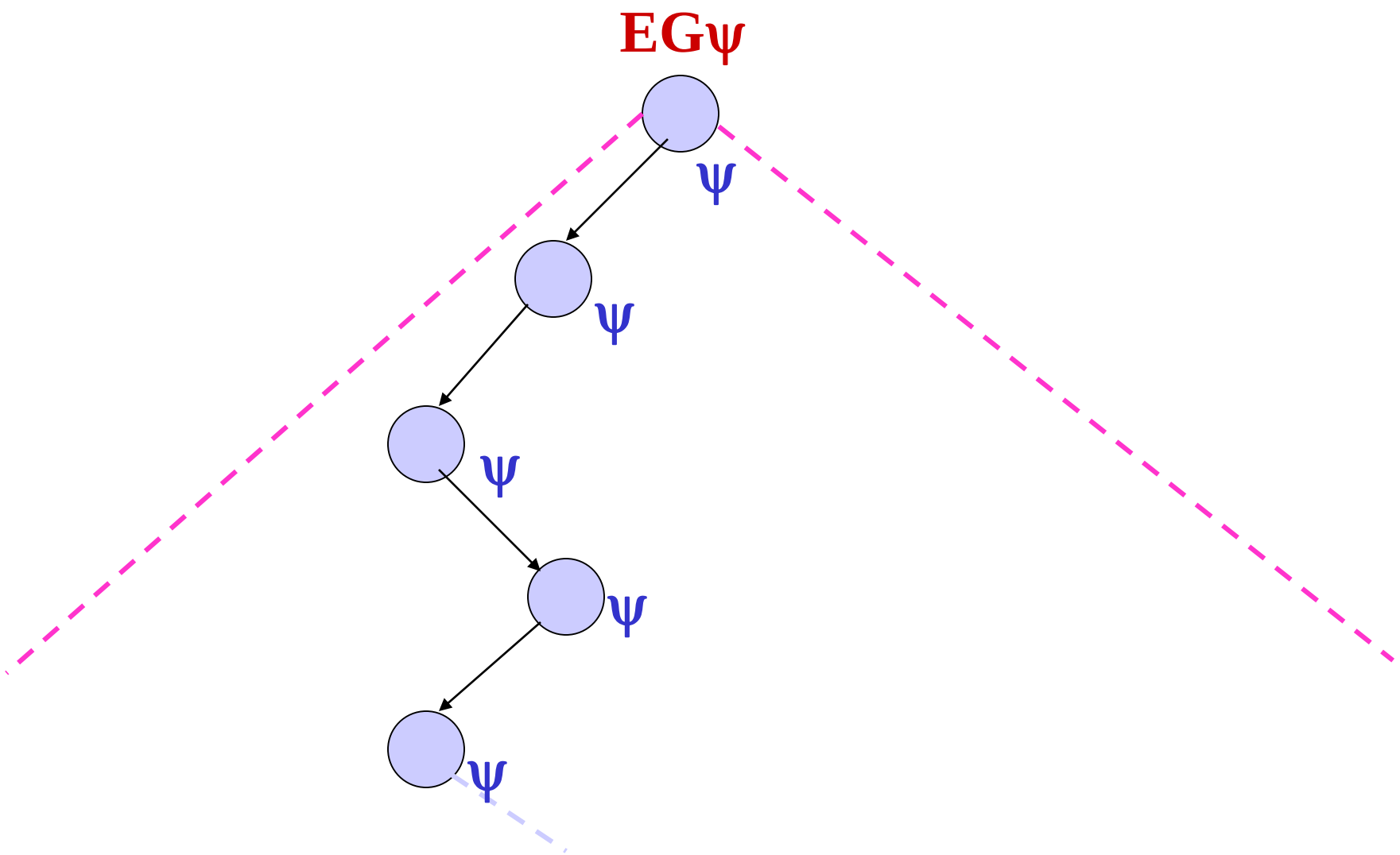
Derived Operators

- **K, s ⊨ AF(ψ)**
- **AF(ψ) = AU(⊤, ψ)**
  - ***For every path π*** from **s**, there ***exists*** **k ≥ 0** such that:
    - **K, π(k) ⊨ ψ.**


AFψ
¬ψ
ψ
ψ
ψ
ψ

Derived Oparators

- **K, s ⊨ EG(ψ)**
- **EG(ψ) = ¬AF(¬ψ)**
  - It is **not** the case that ***for every path π*** from **s** there is a **k ≥ 0** such that **K, π(k) ⊨ ¬ ψ**.
  - ***There exists a path π*** from **s** such that, for every **k ≥ 0**:
    - **K, π(k) ⊨ ψ**.


EGψ
ψ
ψ
ψ
ψ
ψ

A more convenient CTL

- **NCTL ::= p | $\neg\psi$ | $\psi_1 \vee \psi_2$ | EX(ψ) |**
  | EU(ψ_1, ψ_2) | EG(ψ)
- **CTL ::= p | $\neg\psi$ | $\psi_1 \vee \psi_2$ | EX(ψ) |**
  | EU(ψ_1, ψ_2) | AU(ψ_1, ψ_2)
- **NCTL** is more convenient for model checking!
- Clearly **NCTL** can be defined in terms of **CTL**.

A more convenient CTL

- **NCTL ::= p | $\neg\psi$ | $\psi_1 \vee \psi_2$ | EX(ψ) |**
  | EU(ψ_1, ψ_2) | EG(ψ)
- **CTL ::= p | $\neg\psi$ | $\psi_1 \vee \psi_2$ | EX(ψ) |**
  | EU(ψ_1, ψ_2) | AU(ψ_1, ψ_2)
- **CTL** can be defined in terms of **NCTL**!
- The semantics of **NCTL** is given in the obvious way.

A more convenient CTL

- **NCTL ::= $p \mid \neg\psi \mid \psi_1 \vee \psi_2 \mid EX(\psi) \mid$**

  $\mid EU(\psi_1, \psi_2) \mid EG(\psi)$

- **$K, s \models EG(\psi)$** iff ***there exists a path π*** from **s** such that for every **$k \geq 0$**:
  - **$K, \pi(k) \models \psi$**

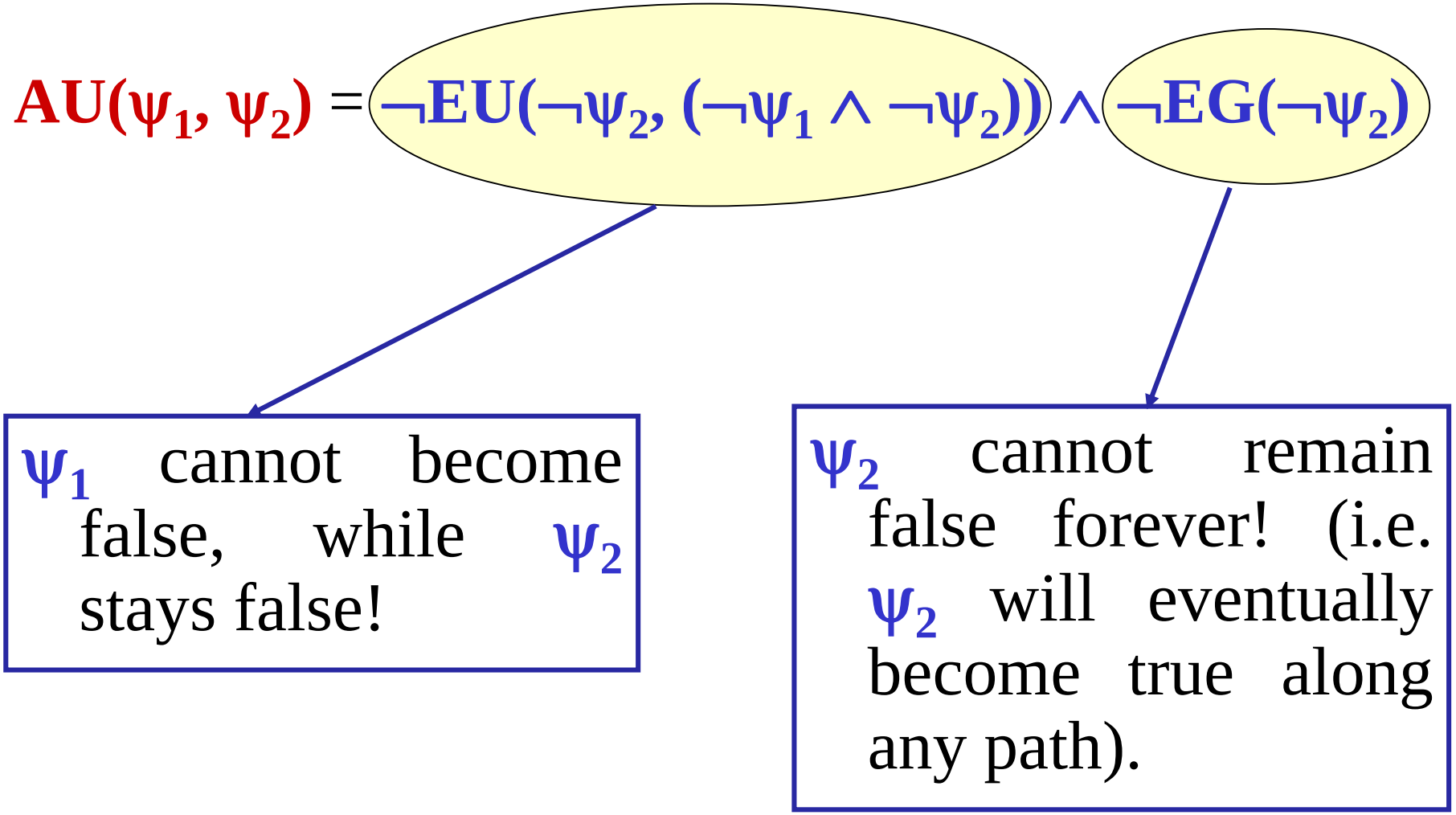
A more convenient CTL

- **NCTL ::= $p \mid \neg\psi \mid \psi_1 \vee \psi_2 \mid EX(\psi) \mid$**
  $\mid EU(\psi_1, \psi_2) \mid EG(\psi)$
- **CTL ::= $p \mid \neg\psi \mid \psi_1 \vee \psi_2 \mid EX(\psi) \mid$**
  $\mid EU(\psi_1, \psi_2) \mid AU(\psi_1, \psi_2)$
- **$AU(\psi_1, \psi_2) = \neg EU(\neg\psi_2, (\neg\psi_1 \wedge \neg\psi_2)) \wedge \neg EG(\neg\psi_2)$**

i.e., along any path: ψ_2 must hold eventually and $(\neg\psi_1 \wedge \neg\psi_2)$ can only happen after ψ_2 (recall the *before operator* of LTL)

A more convenient CTL

$$\mathbf{AU}(\psi_1, \psi_2) = \neg\mathbf{EU}(\neg\psi_2, (\neg\psi_1 \wedge \neg\psi_2)) \wedge \neg\mathbf{EG}(\neg\psi_2)$$



A more convenient CTL

$\mathbf{AU(\psi_1, \psi_2) = \neg EU(\neg\psi_2, (\neg\psi_1 \wedge \neg\psi_2)) \wedge \neg EG(\neg\psi_2)}$

$\Rightarrow$ Assume $\mathbf{K, s \vDash AU(\psi_1, \psi_2)}$

- Let π be a path from **s**. Then there exists $\mathbf{k \geq 0}$ with:
  - $\mathbf{K, s \vDash \psi_2}$
- Hence, **not** $\mathbf{K, s \vDash EG(\neg\psi_2)}$
- Equivalent to $\mathbf{K, s \vDash \neg EG(\neg\psi_2)}$

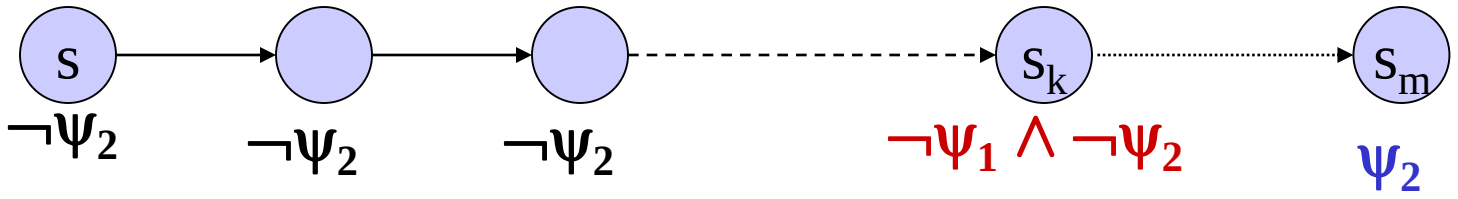
A more convenient CTL

- **$AU(\psi_1, \psi_2)$ = NewAU(ψ_1, ψ_2) =**

  $\neg EU(\neg\psi_2, (\neg\psi_1 \wedge \neg\psi_2)) \wedge \neg EG(\neg\psi_2)$

- Clearly **$K, s \models AU(\psi_1, \psi_2)$** implies **$K, s \models \neg EG(\neg\psi_2)$**

- Let **$K, s \models AU(\psi_1, \psi_2)$**
  - Suppose now **$K, s \models EU(\neg\psi_2, \neg\psi_1 \wedge \neg\psi_2)$**
  - Let **π** be any path from **s** witnessing the above:
  - Let now **k *be the least integer*** such that:
    - **$K, \pi(k) \models \neg\psi_1 \wedge \neg\psi_2$**
    - **$K, \pi(j) \models \neg\psi_2$** for **$0 \leq j < k$.**



- Suppose **K, π(m) ⊨ ψ_2,** required by **K, s ⊨ AU(ψ_1,ψ_2)**
- Take **m** to be the least such number.
- Then **k < m**, since **K, s ⊨ EU($\neg\psi_2$, $\neg\psi_1 \wedge \neg\psi_2$)**
- This implies that **K, π(k) ⊨ $\neg\psi_1$**, for some **0 ≤ k < m**
- Hence **not K, s ⊨ AU(ψ_1, ψ_2)**. ***Contradiction!***
- Thus **K, s ⊨ AU(ψ_1, ψ_2)** also implies:
  - **K, s ⊨ $\neg$EU($\neg\psi_2$, $\neg\psi_1 \wedge \neg\psi_2$)**
- So **K, s ⊨ AU(ψ_1, ψ_2)** implies **K, s ⊨ NewAU(ψ_1, ψ_2)**

From CTL to NCTL

- In a similar way we can argue that:

> ***if*** **K, s ⊨ newAU(ψ_1, ψ_2)**
>
> ***then*** **K, s ⊨ AU(ψ_1, ψ_2).**

- Hence ***CTL*** can be expressed in terms of ***NCTL***.

A more convenient CTL

- **NCTL ::= p | $\neg\psi$ | $\psi_1 \vee \psi_2$ | EX(ψ) |**
  | EU(ψ_1, ψ_2) | EG(ψ)
- **CTL ::= p | $\neg\psi$ | $\psi_1 \vee \psi_2$ | EX(ψ) |**
  | EU(ψ_1, ψ_2) | AU(ψ_1, ψ_2)
- **AU(ψ_1, ψ_2) = NewAU(ψ_1, ψ_2) =**

  $\neg$(EU($\neg\psi_2$, ($\neg\psi_1 \wedge \neg\psi_2$)) $\wedge$ AF(ψ_2)
- **NewAU$_1$ = $\neg$ EU($\neg\psi_2$, ($\neg\psi_1 \wedge \neg\psi_2$)**
- **NewAU$_2$ = AFψ_2**

$\neg$EG$\neg\psi_2$ = AFψ_2

From CTL to NCTL

- Let $\mathbf{K = (S, S_0, R, AP, L)}$ and $\mathbf{s \in S}$.
- We need to argue:
  - $\mathbf{K, s \vDash AU(\psi_1, \psi_2)}$ iff

    $\mathbf{K, s \vDash NewAU_1 \wedge NewAU_2}$
- We already argued that:
  - If $\mathbf{K, s \vDash AU(\psi_1, \psi_2)}$ then

    $\mathbf{K, s \vDash NewAU_1 \wedge NewAU_2}$

From CTL to NCTL

$\mathbf{AU(\psi_1, \psi_2) = \neg EU(\neg\psi_2, (\neg\psi_1 \wedge \neg\psi_2)) \wedge \neg EG(\neg\psi_2)}$

$\Leftarrow$ We need to argue that:

– If $\mathbf{K, s \vDash NewAU_1 \wedge NewAU_2}$ then

$\mathbf{K, s \vDash AU(\psi_1, \psi_2)}$

- So assume $\mathbf{K, s \vDash NewAU_1 \wedge NewAU_2}$.

- $\mathbf{NewAU_1 = \neg EU(\neg\psi_2, (\neg\psi_1 \wedge \neg\psi_2)).}$
- $\mathbf{NewAU_2 = \neg EG\neg\psi_2 = AF\psi_2}$

From CTL to NCTL

- Let π be some path from **s**.
- We need to show that there exists $k \geq 0$ such that:
  - $K, \pi(k) \models \psi_2$
  - $K, \pi(j) \models \psi_1$ if $0 \leq j < k$.
- But $K, s \models AF\ \psi_2$ implies that along any path (and also along π) there exists $k \geq 0$ such that:
  - $K, \pi(k) \models \psi_2$
- Assume **k** is the ***least*** such number along π.

From CTL to NCTL

Now consider an arbitrary **m** with **$0 \leq m < k$**.

CLAIM: **$K, \sigma(m) \models \psi_1$**

- If the **CLAIM** is true then we are done.
- Suppose instead that **$K, \sigma(m) \models \neg\psi_1$**.
  - Then **$K, \sigma(m) \models \neg\psi_1 \wedge \neg\psi_2$** (**$m < k$**) ***WHY???***
  - and **$K, \sigma(j) \models \neg\psi_2$** if **$0 \leq j < m$**, since **$j < m < k$**
  - Hence **$K, \sigma(0) \models EU(\neg\psi_2, \neg\psi_1 \wedge \neg\psi_2)$**
  - Therefore, **not** **$K, s \models NewAU_1$** which is a ***contradiction***!

CTL Model Checking

- **$K \models \psi$** ***iff***

  $K, s_0 \models \psi$ for every **$s_0 \in S_0$**.

- The ***CTL model checking problem***.
  - **$K = (S, S_0, R, AP, L)$** (system model)
  - **ψ** a ***CTL*** formula (spec. of the property)

- Given **K** and **ψ** ***determine whether or not*** **$K \models \psi$**

CTL Model Checking

- The actual model checking problem:
  - Given **$K = (S, S_0, R, AP, L)$**
  - Given **$s \in S$**
  - Given **ψ**, **an NCTL formula**.
  - Determine whether:

$K, s \models \psi$

The Sub-formulas of ψ

- **SF(ψ)** is the ***least set of formulas*** satisfying:
  - $\psi \in SF(\psi)$
  - If $\neg\alpha \in SF(\psi)$ then $\alpha \in SF(\psi)$.
  - If $\alpha \vee \beta \in SF(\psi)$ then $\alpha, \beta \in SF(\psi)$
  - If $EX\alpha \in SF(\psi)$ then $\alpha \in SF(\psi)$.
  - If $EU(\alpha, \beta) \in SF(\psi)$ then $\alpha, \beta \in SF(\psi)$
  - If $EG\alpha \in SF(\psi)$ then $\alpha \in SF(\psi)$.
- **SF(ψ)** ---- The ***set of sub-formulas*** of ψ.

The Labeling Procedure.

- **K = (S, S_0, R, AP, L)**
  - **s ∈ S**
  - **ψ** a ***NCTL*** formula (built out of **AP**).
- **Strategy**:
  - Construct **Labels: S ⟶ $2^{SF(\psi)}$**
  - **$2^{SF(\psi)}$**, the set of subsets of **SF(ψ)**.
  - Each state of **K** is assigned a subset of a **SF(ψ)** by the Labels function.
- **K, s ⊨ ψ** ***iff*** **ψ ∈ Labels(s)**.

The Labels function

- **Stage 0**: consider the atomic propositions only
  - For every **$t \in S$**:
  - **Labels(t) = L(t)** (**$K = (S, S_0, R, AP, L)$**)
- ….

Assume we have done up to stage **i** (all subfomrulae of length **i** already processed)

- **Stage i +1**: consider subfomrulae **α** of length **i+1**
  - For every **$t \in S$**:
  - If **$\alpha = \neg\beta$** then

    $\alpha \in$ Labels(t) ***iff*** **$\beta \notin$ Labels(t)**.

The Labels function

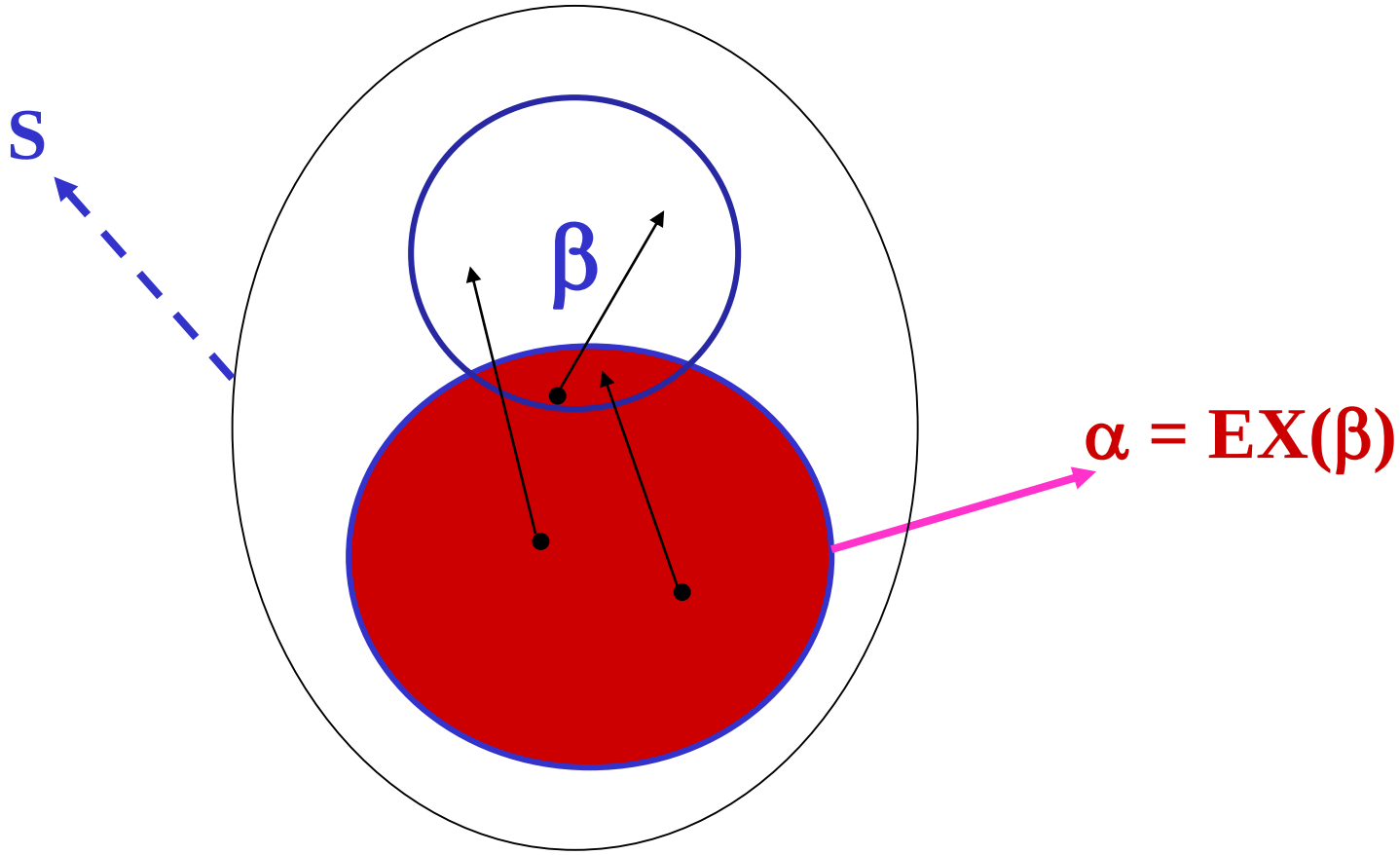
- **Stage i +1**: consider subfomrulae α of length **i+1**
  - For every $t \in S$:
  - If $\alpha = \beta_1 \vee \beta_2$ then

    $\alpha \in$ **Labels(t)** ***iff*** $\beta_1 \in$ **Labels(t) or** $\beta_2 \in$ **Labels(t)**
  - If $\alpha = EX\beta$ then

    $\alpha \in$ **Labels(t)** ***iff*** **there exists** $s \in S$ **such that**

    $\beta \in$ **Labels(s)** and **R(t, s) [i.e.** $t \rightarrow s$ **]**

The Labels Function



Computing the labeling for EX(β)

Complexity: O(\|M\|)

Algorithm Check_EX(β)

T := {**s** | β ∈ **Labels(s)**};

while **T** ≠ ∅ do

choose **s** ∈ **T**;

T := **T** \{**s**};

for each **t** ∈ **S** such that **t** → **s** do

Labels(**t**) := **Labels**(**t**) ∪ {**EX** β};

The Labels Function

- **Stage i +1**: consider subfomrulae α of length **i+1**
  - For every $t \in S$:
  - If $\alpha = EU(\beta_1,\beta_2)$ then

    $\alpha \in Labels(t)$ ***iff***
    - $\beta_2 \in Labels(t)$ **or**
    - $\beta_1 \in Labels(t)$ **and** $EU(\beta_1,\beta_2) \in Labels(s)$ **for some s with** $t \rightarrow s$.

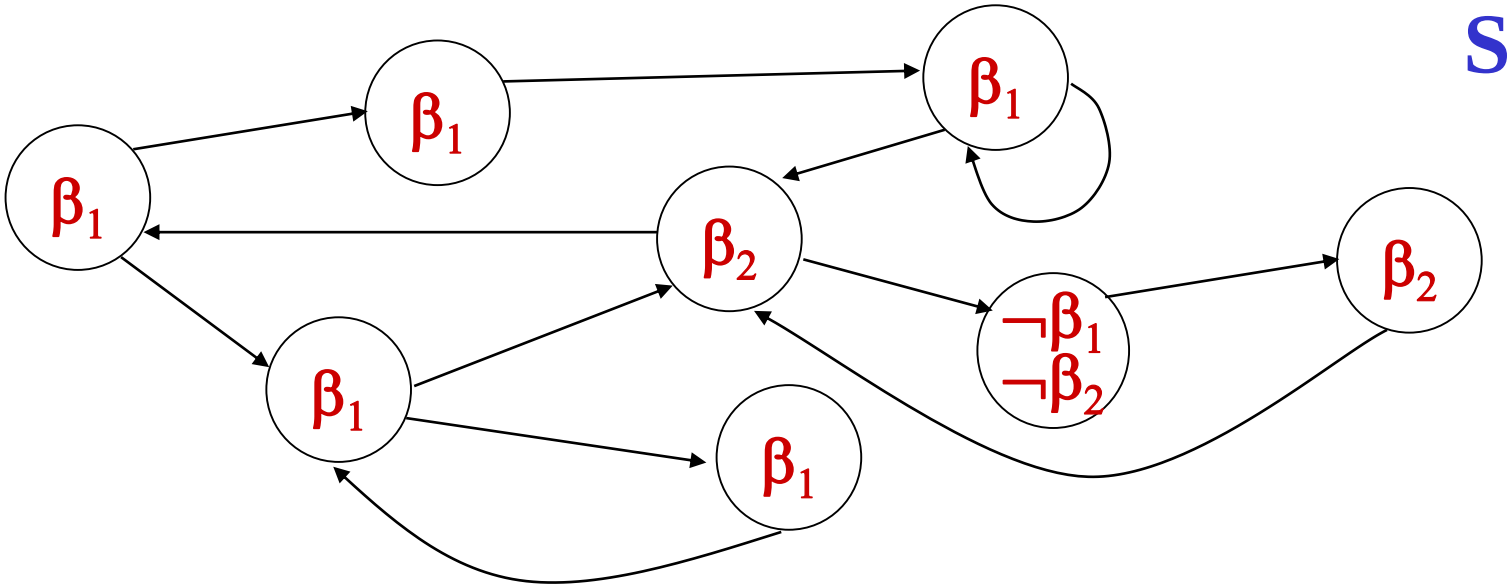
The Labels Function

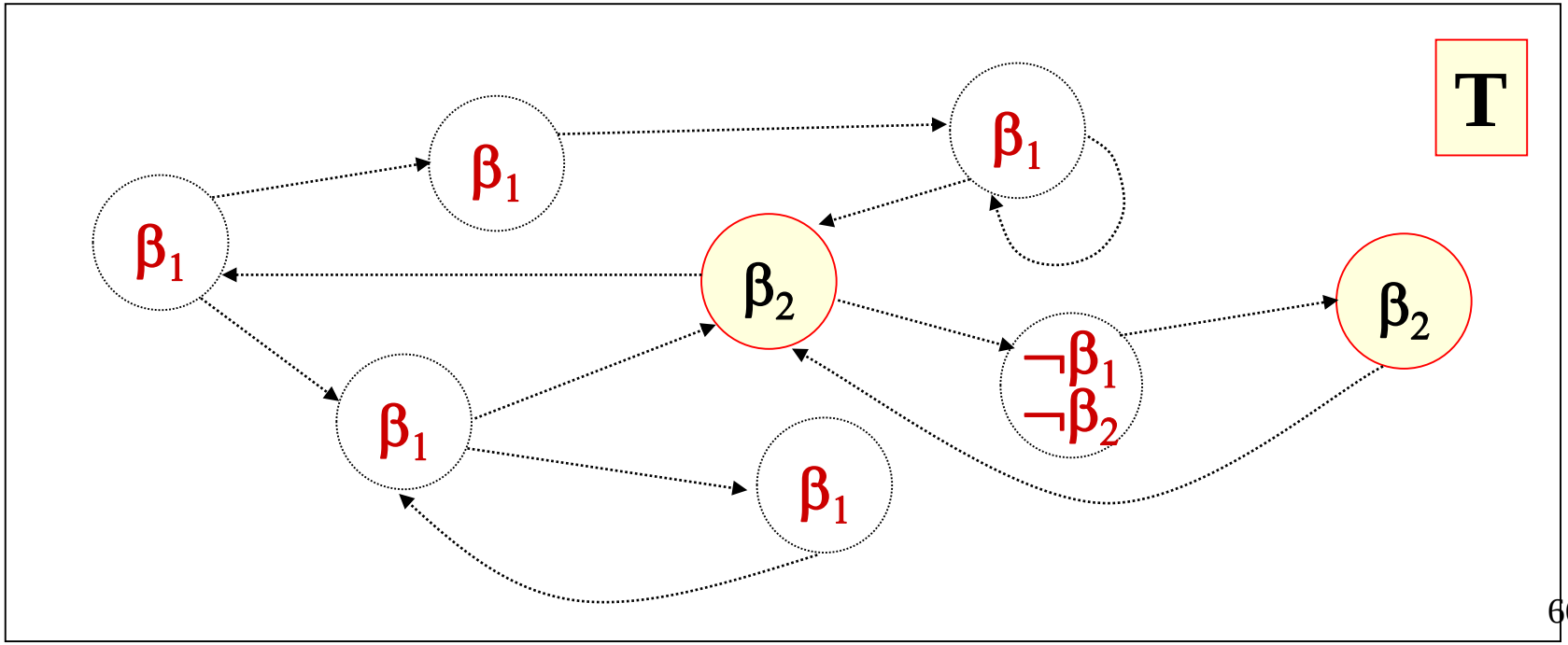
- Collect in **T** all the states satisfying β_2
  - **all these states do also satisfy $EU(\beta_1,\beta_2)$.**
- Traverse backward $\rightarrow$ from states in **T** and label with $EU(\beta_1,\beta_2)$ all the states **t** satisfying β_1 and reaching at least a state **s** labeled with $EU(\beta_1,\beta_2)$.

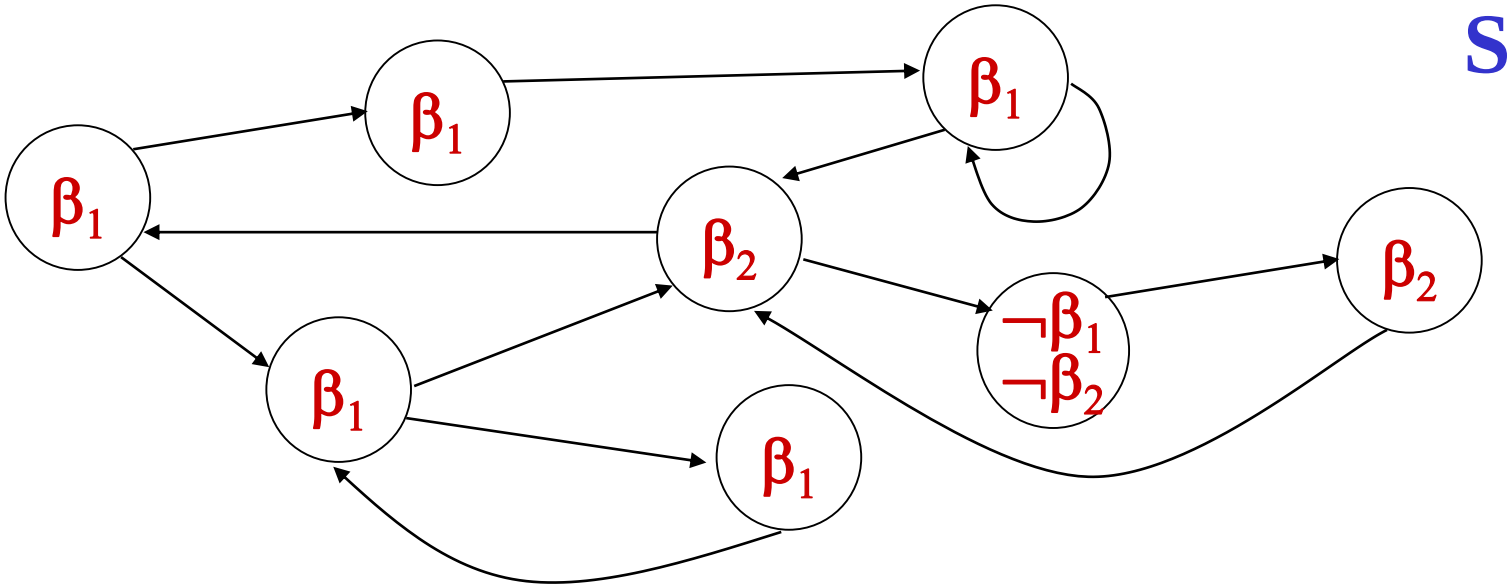
If $s \in T$, t with $t \rightarrow s$ and $\beta_1 \in Labels(t)$ then

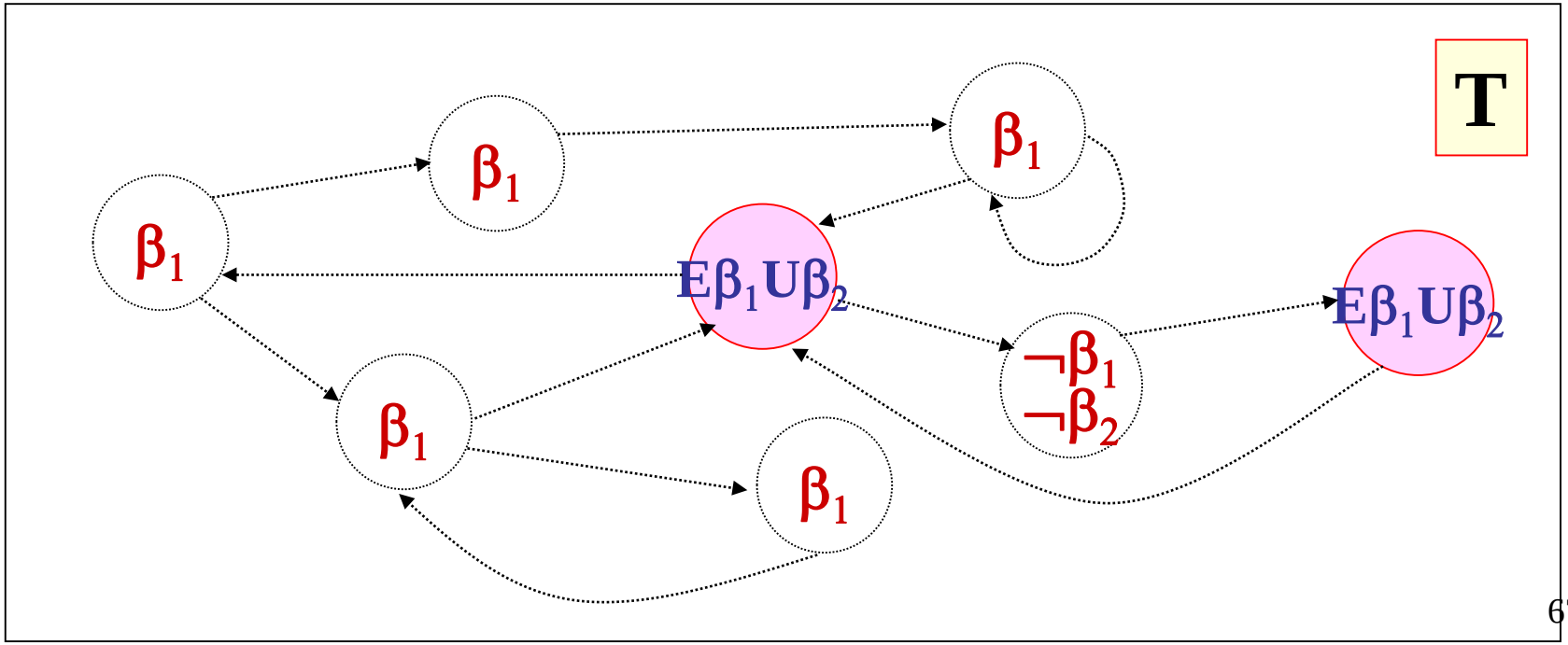
$EU(\beta_1,\beta_2) \in Labels(t)$

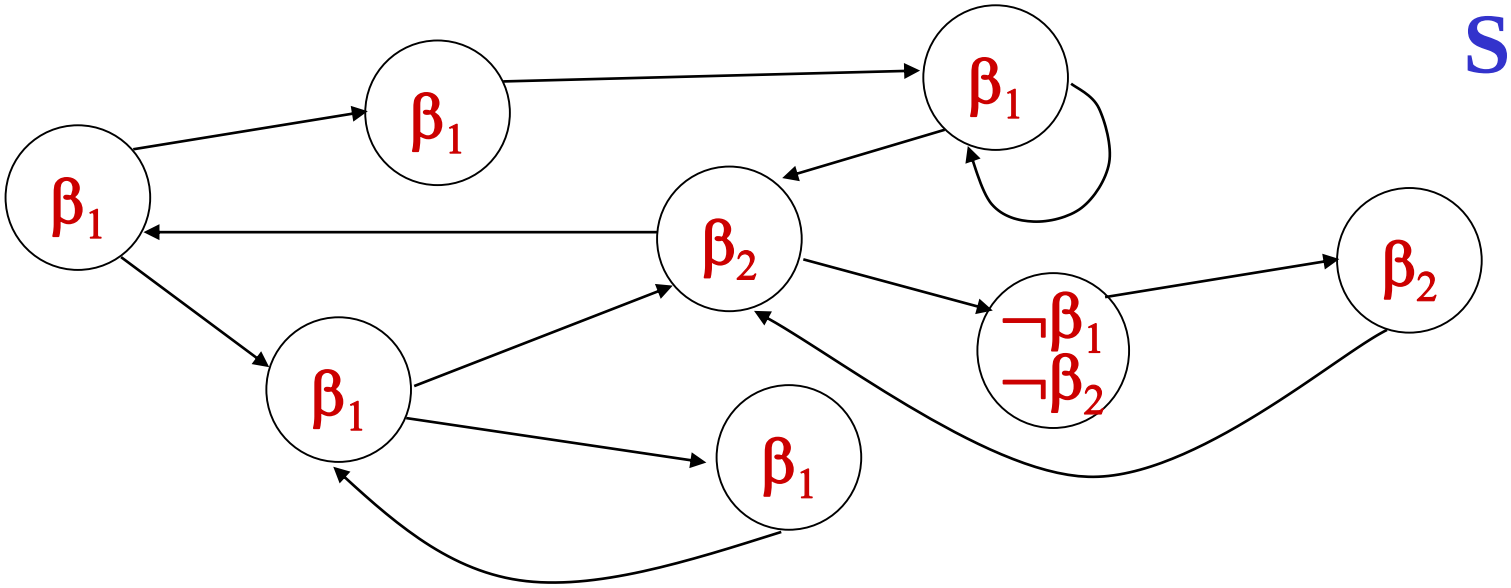
Recall that: $EU(\beta_1,\beta_2) = (\beta_2 \vee (\beta_1 \wedge EXEU(\beta_1,\beta_2)))$

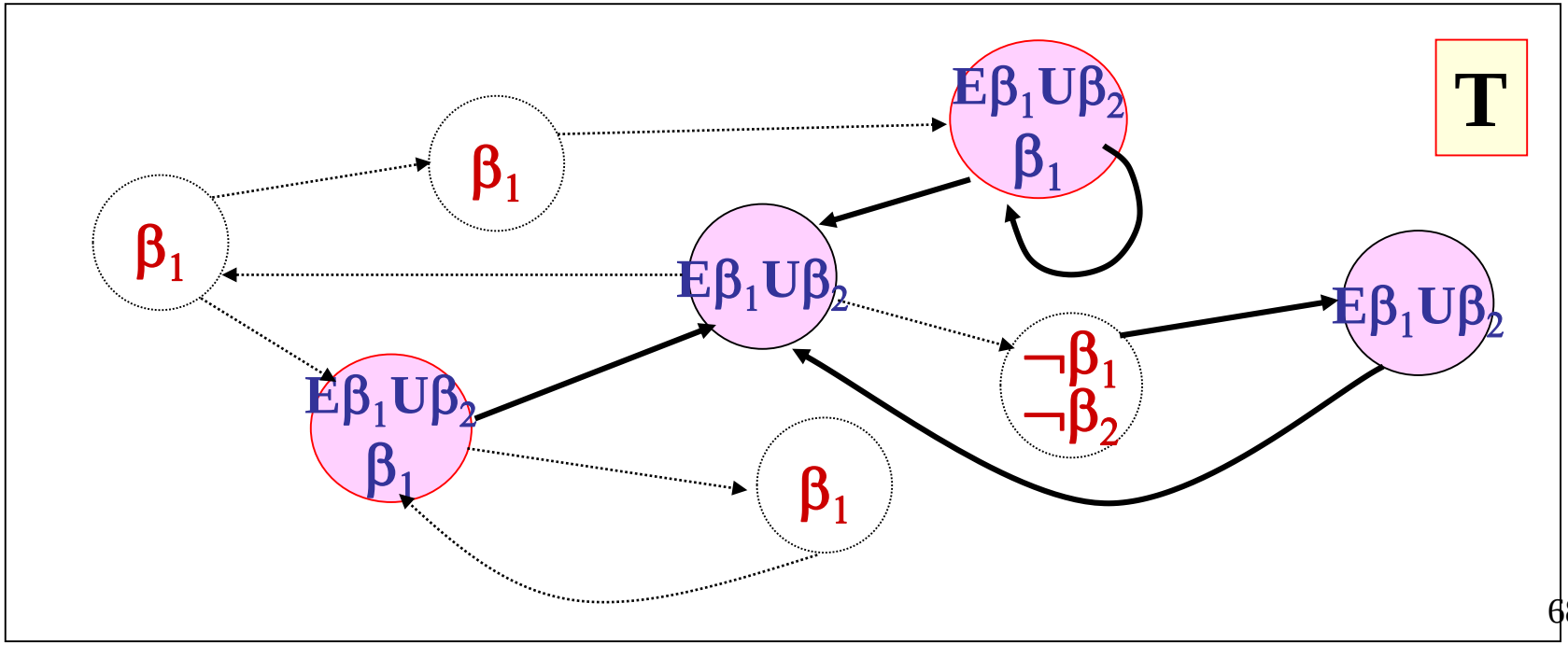

S
β1
β1
β1
β2
β2
¬β1
¬β2
β1
β1

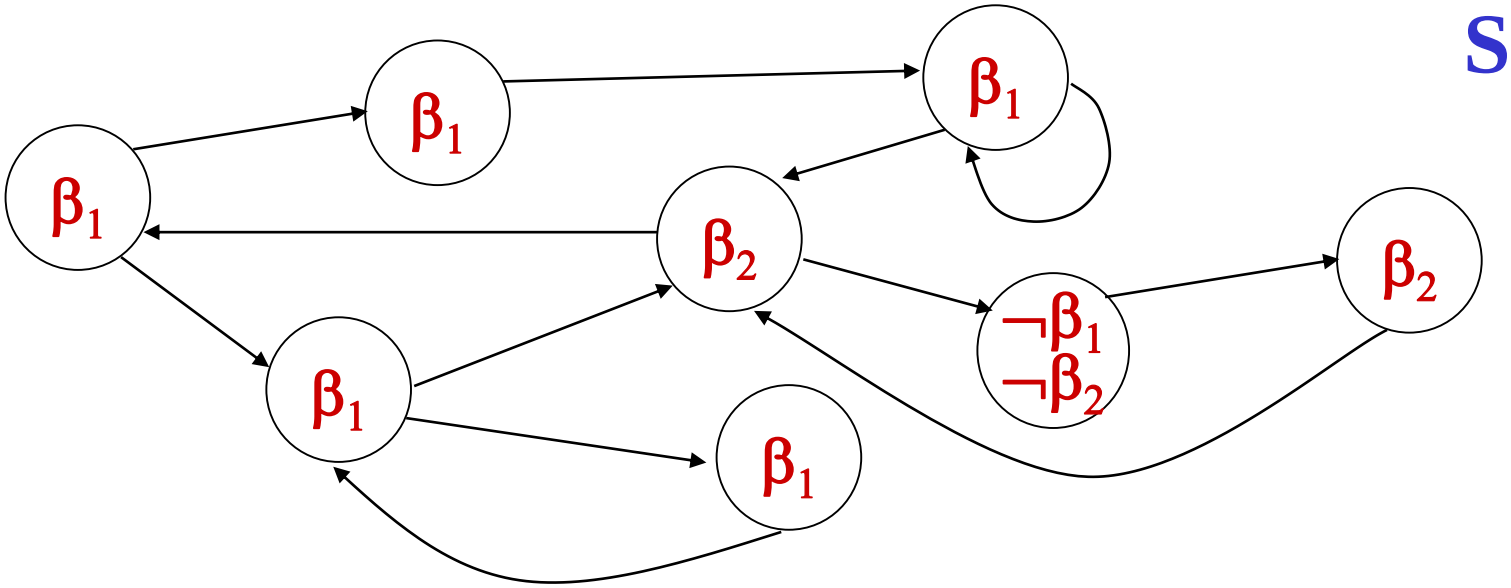

T
β1
β1
β1
β2
β2
¬β1
¬β2
β1
β1

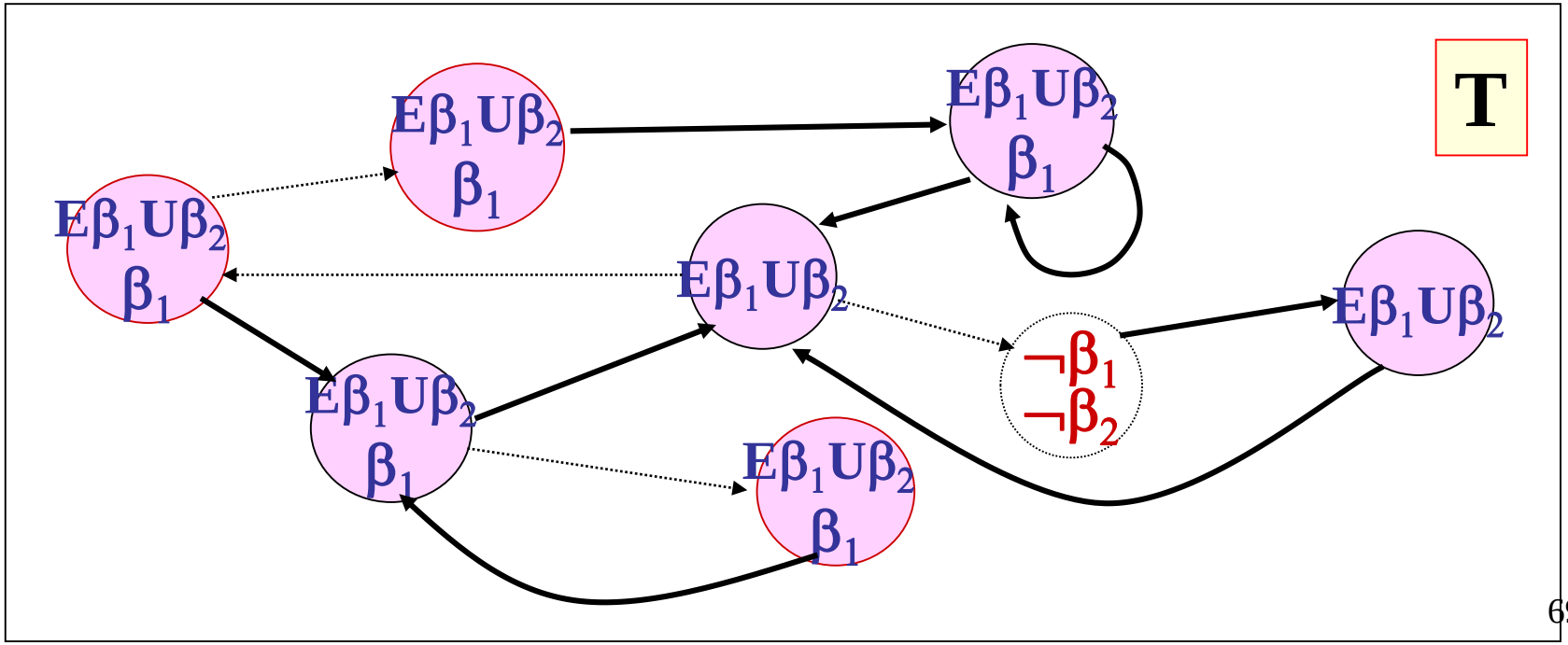

S
β1
β1
β1
β2
β2
β1
¬β1
¬β2
β1

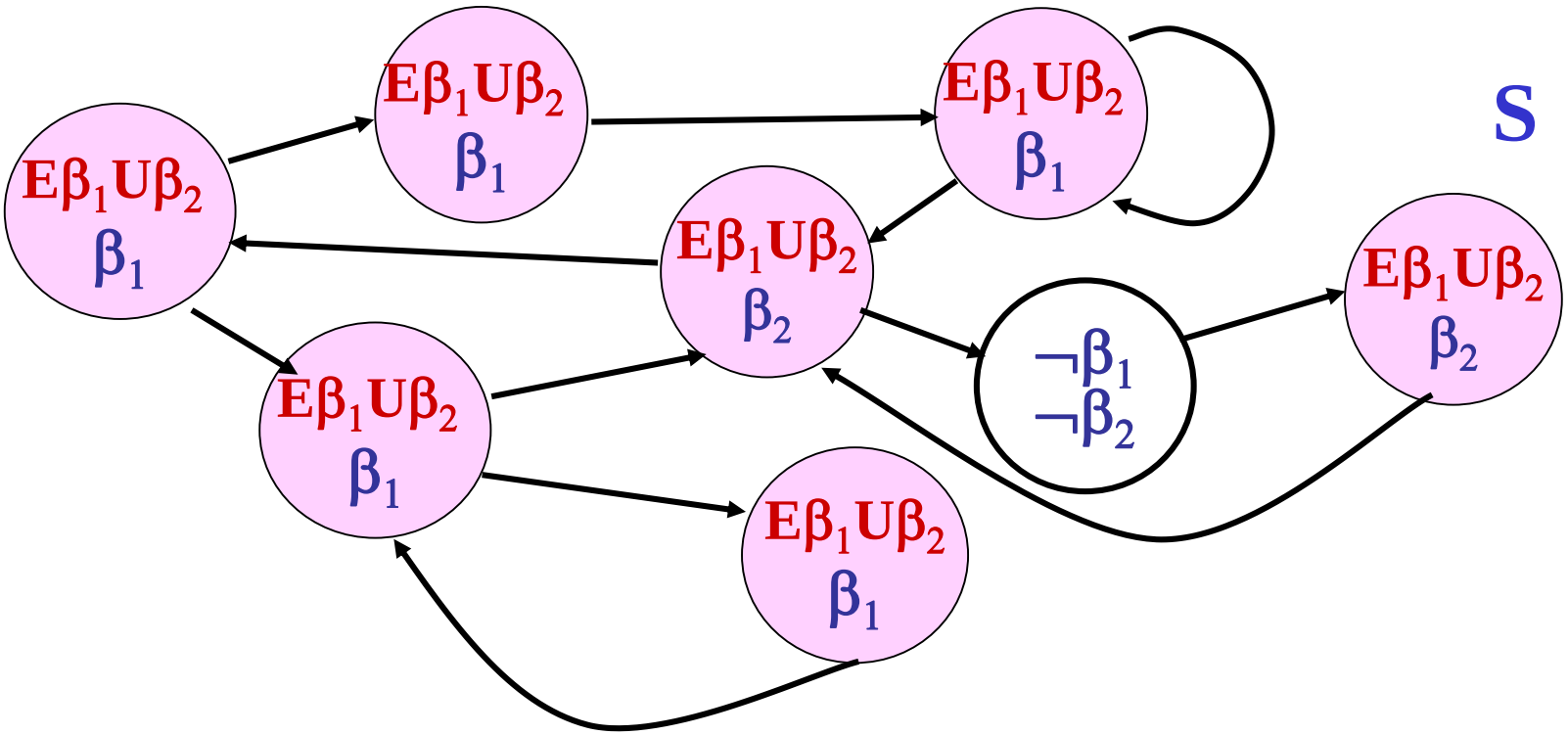

T
β1
β1
β1
Eβ1Uβ2
Eβ1Uβ2
¬β1
¬β2
β1
β1


S
β1
β1
β1
β2
β2
β1
β1
¬β1
¬β2


T
Eβ1Uβ2
β1
β1
β1
Eβ1Uβ2
Eβ1Uβ2
¬β1
¬β2
Eβ1Uβ2
β1
β1


S
β1
β1
β1
β2
β2
β1
¬β1
¬β2
β1


T
Eβ1Uβ2
β1
Eβ1Uβ2
β1
Eβ1Uβ2
β1
Eβ1Uβ2
Eβ1Uβ2
¬β1
¬β2
Eβ1Uβ2
β1
Eβ1Uβ2
β1


$E\beta_1 U \beta_2$ β_1
$E\beta_1 U \beta_2$ β_1
$E\beta_1 U \beta_2$ β_1
$E\beta_1 U \beta_2$ β_2
$E\beta_1 U \beta_2$ β_2
$E\beta_1 U \beta_2$ β_1
$E\beta_1 U \beta_2$ β_1
$\neg\beta_1$ $\neg\beta_2$
S

Computing the labeling for $EU(\beta_1,\beta_2)$

Algorithm Check_$EU(\beta_1,\beta_2)$

T := {**s** | $\beta_2 \in$ Labels(**s**)};

for each **s** ∈ **T** do

Labels(**s**) := Labels(**s**) ∪ {**EU**(β_1,β_2)};

while **T** ≠ ∅ do

chose **s** ∈ **T**;

T := **T** \{**s**};

for each **t** ∈ **S** with **t** → **s** do

if **EU**(β_1,β_2) ∉ Labels(**t**) and $\beta_1 \in$ Labels(**t**) then

Labels(**t**) := Labels(**t**) ∪ {**EU**(β_1,β_2)};

T := **T** ∪ {**t**};

Complexity: O(|M|)

The Labels Function

- **Stage i +1**: consider subfomrulae **α** of length **i+1**
  - For every **t ∈ S**:
  - If **α = EG(β)** then

    α ∈ Labels(t) ***iff***

    - **β ∈ Labels(t) and EG(β) ∈ Labels(s) for some s with t → s.**

Property of EG(β)

Let **M' = (S',R',L')** be the sub-graph of **M** where

- **S' = { s | M ,s ⊨ β }**
- **R' = $R|_{S' \times S'}$** (the restriction of **R** to **S'**)
- **L' = $L|_{S'}$** (the restriction of **L** to **S'**)

Lemma: M,s ⊨ EG(β) *iff*

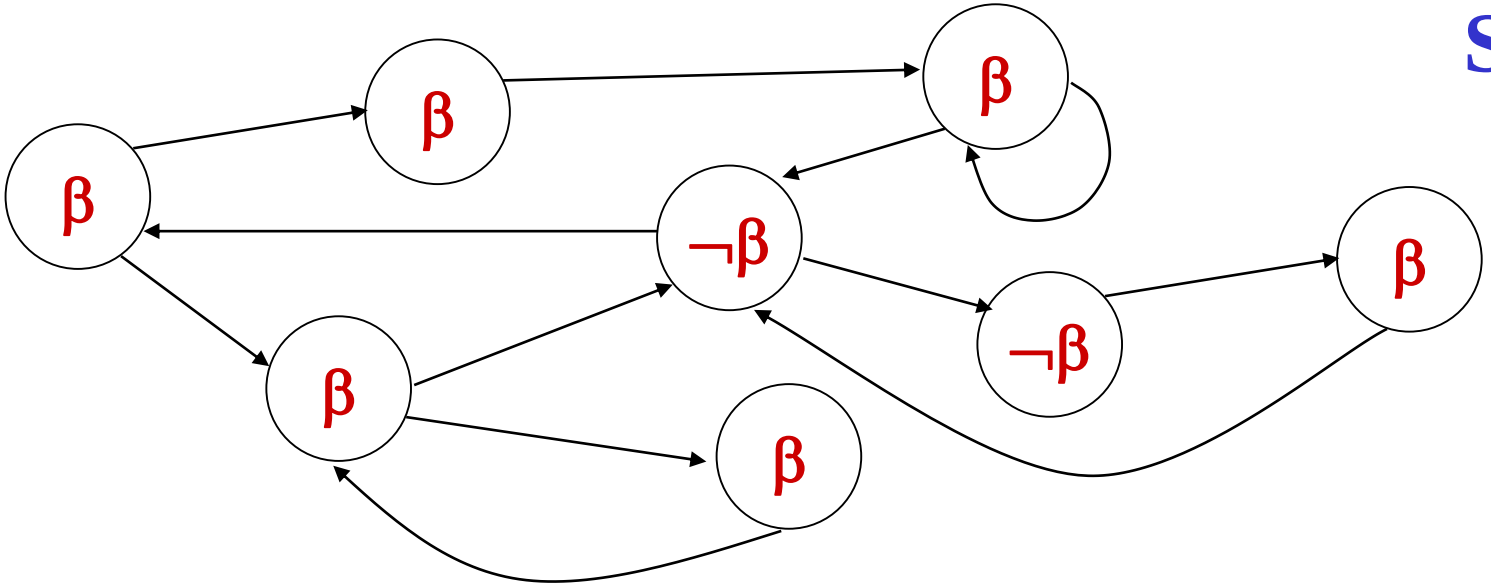
1. **s ∈ S' and**
2. ***there exists a path*** in **M'** leading from **s** to a ***non-trivial strongly connected component*** **C** of the graph **(S',R').**

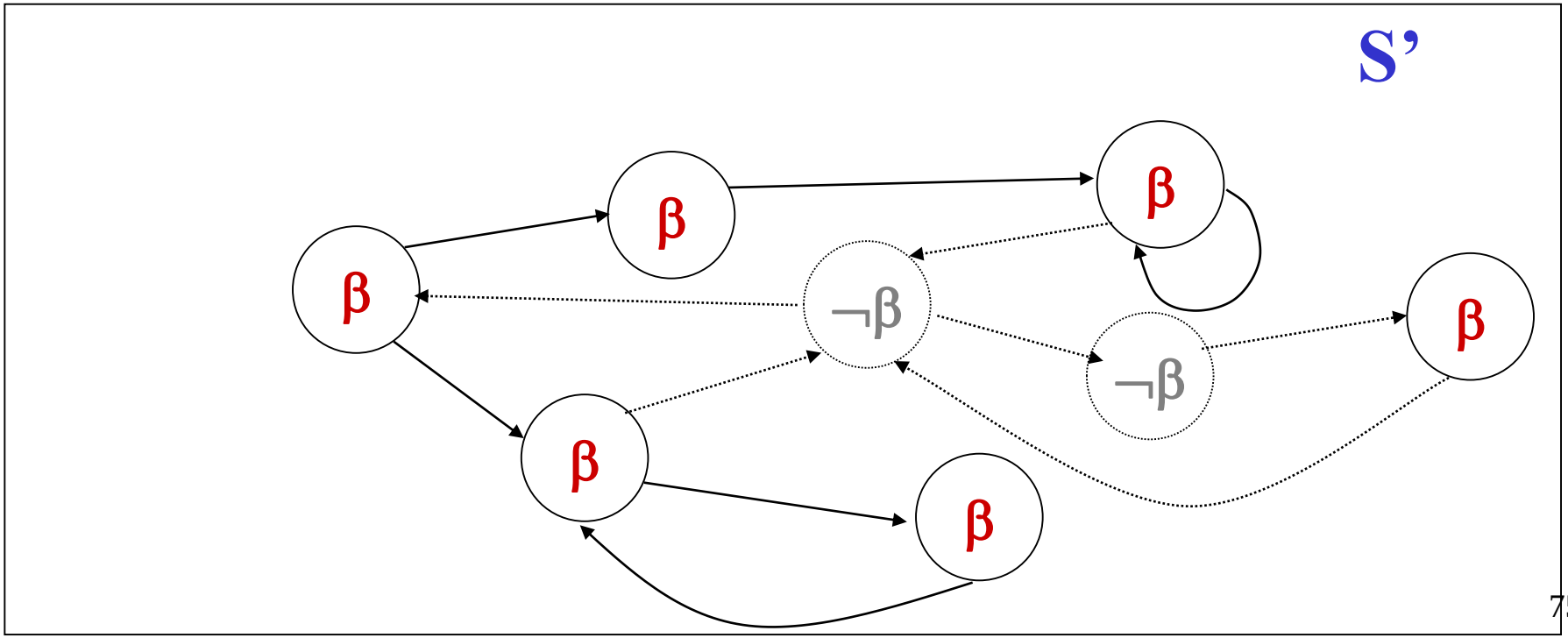
The Labels Function

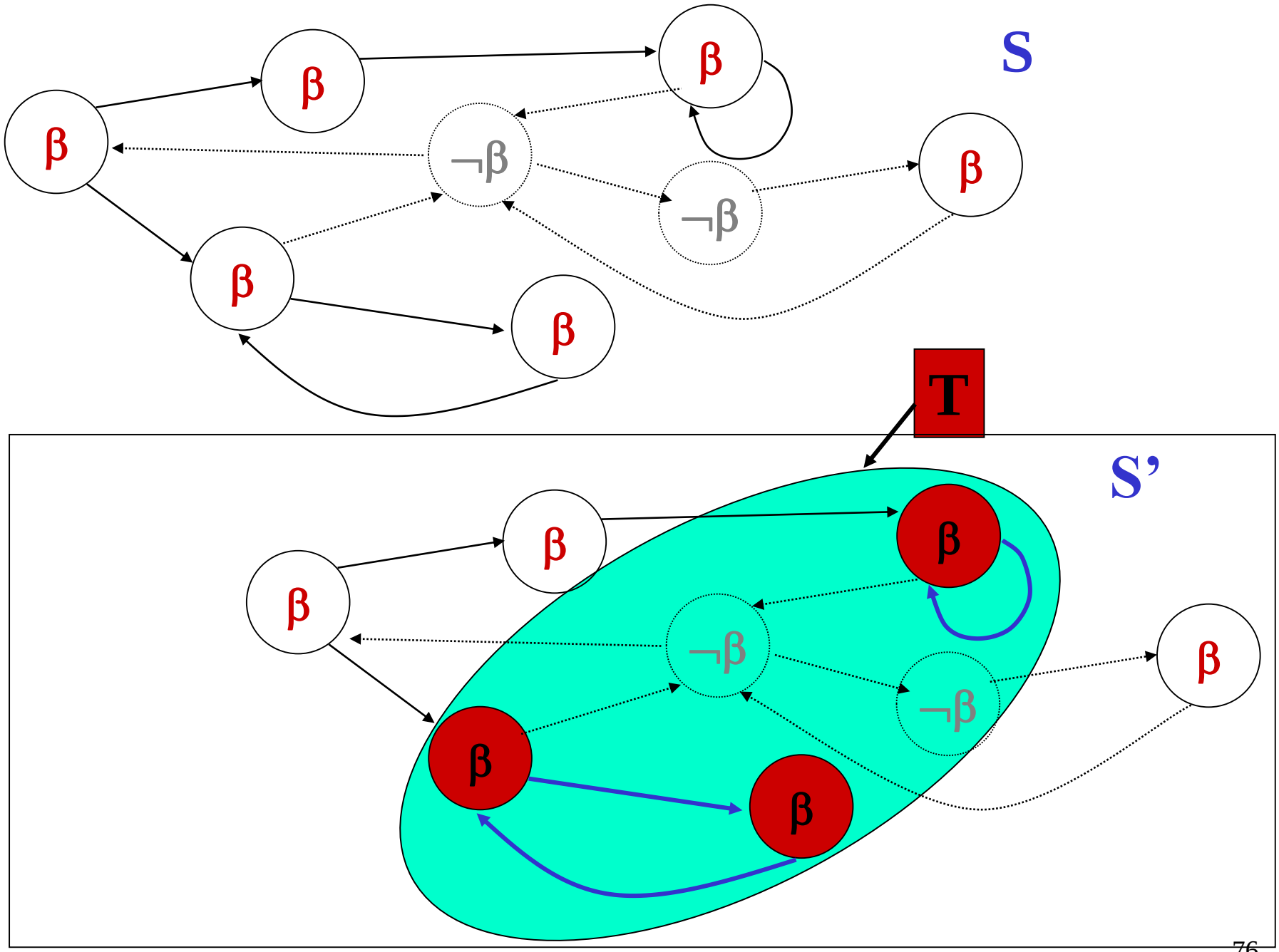
- Compute the subgraph **S'** whose states satisfy **β**
- Take ***non-trivial strongly connected components*** of **S'**
  - **all the states in these components do satisfy EG(β).**
- Traverse backward **→'** and label with **EG(β)** the ***states t reaching at least a state s*** labeled with **EG(β)** (note that both **t** and **s** belong to **S'**).

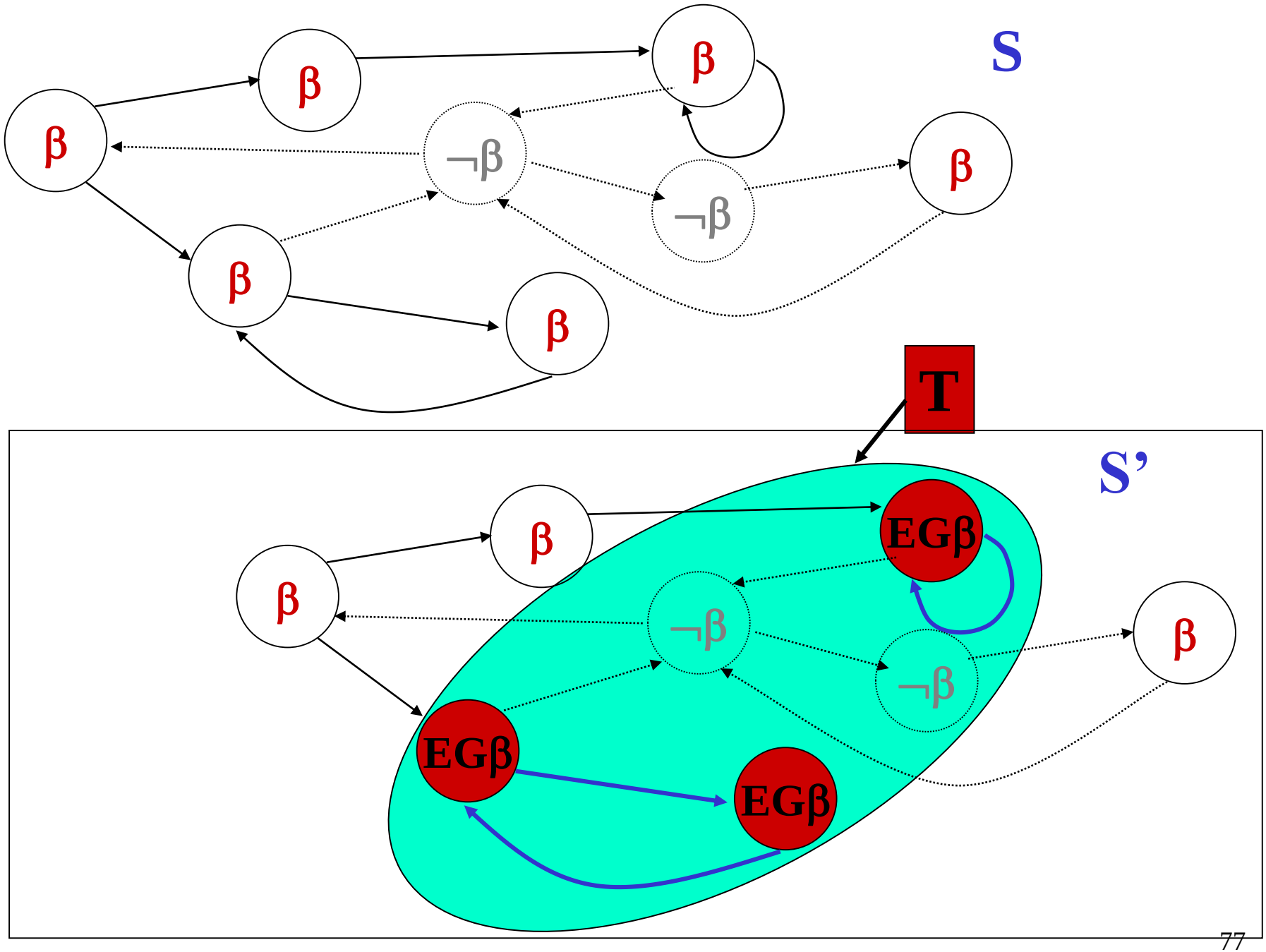
If t ∈ S' and R(t,s) then EG(β) ∈ Labels(t)

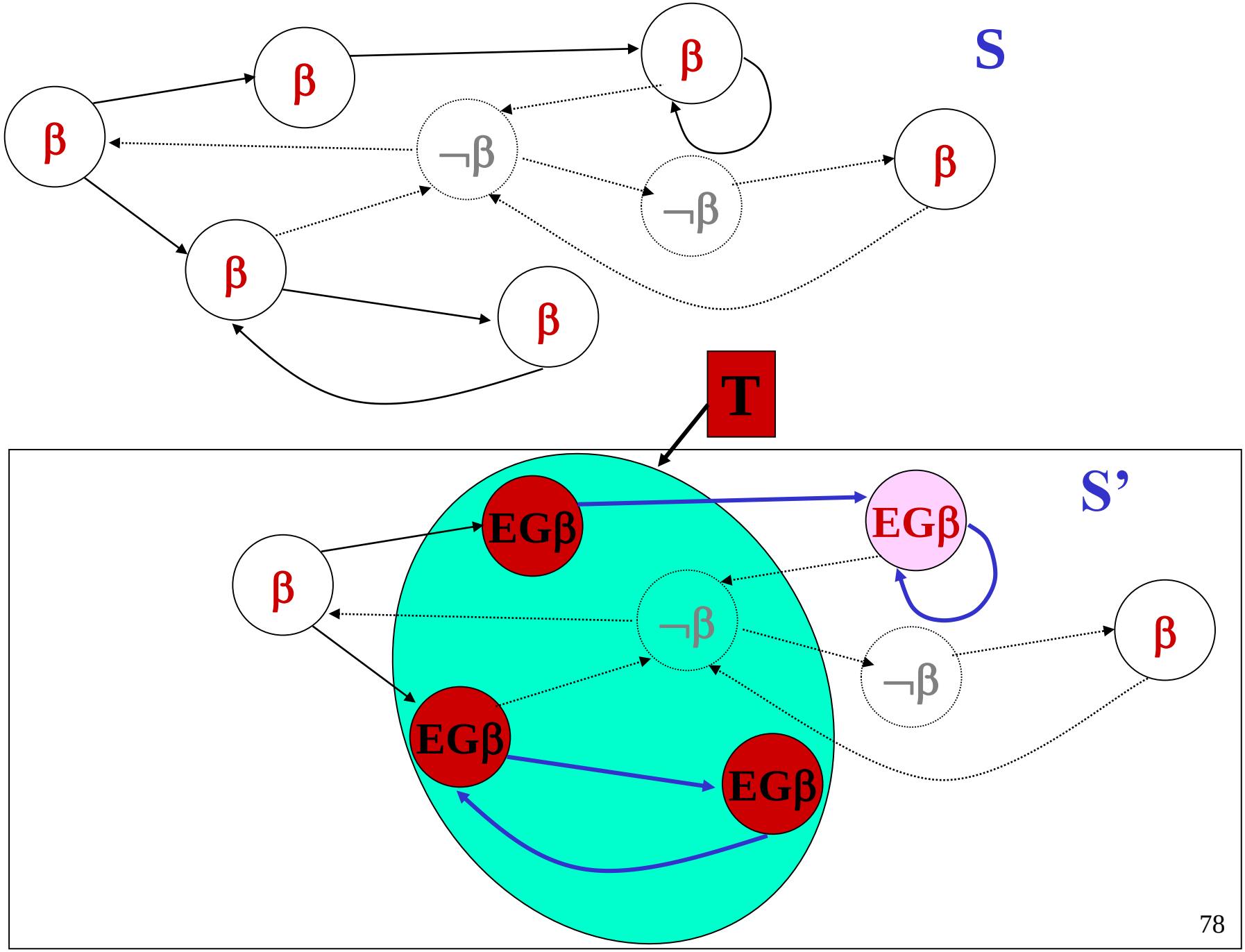
Recall that: **EG β = β ∧ EXEG β**

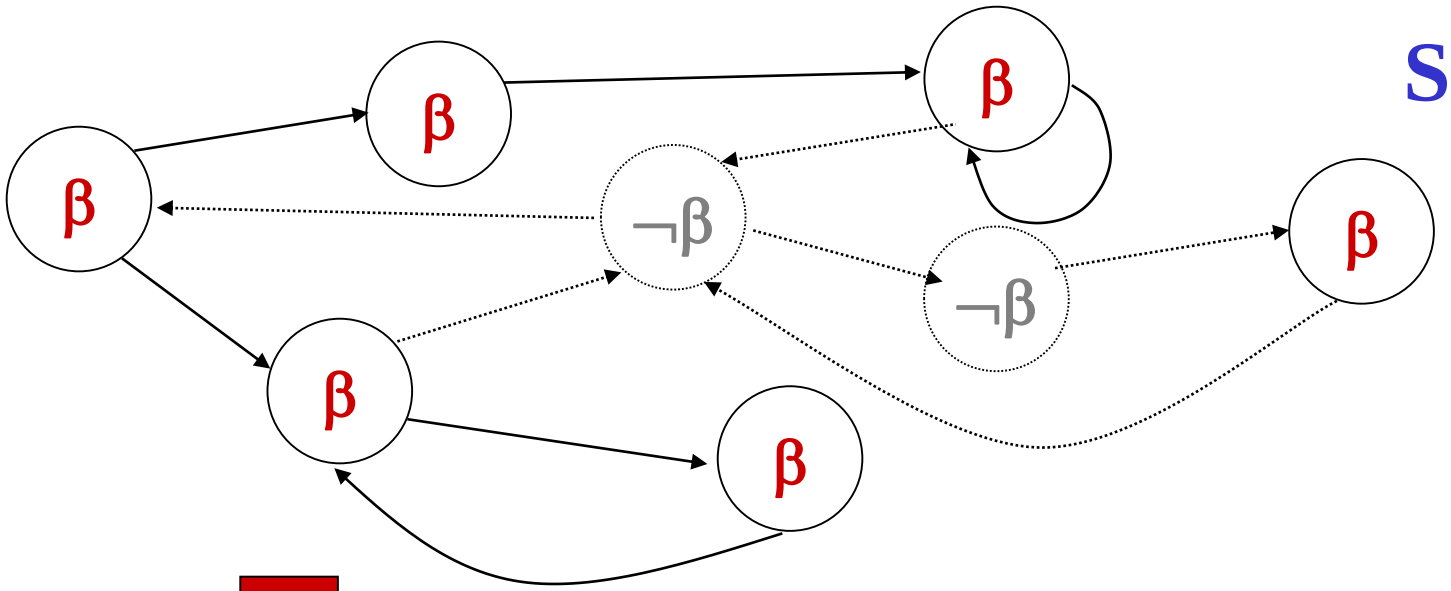

S
β
β
β
¬β
β
¬β
β
β

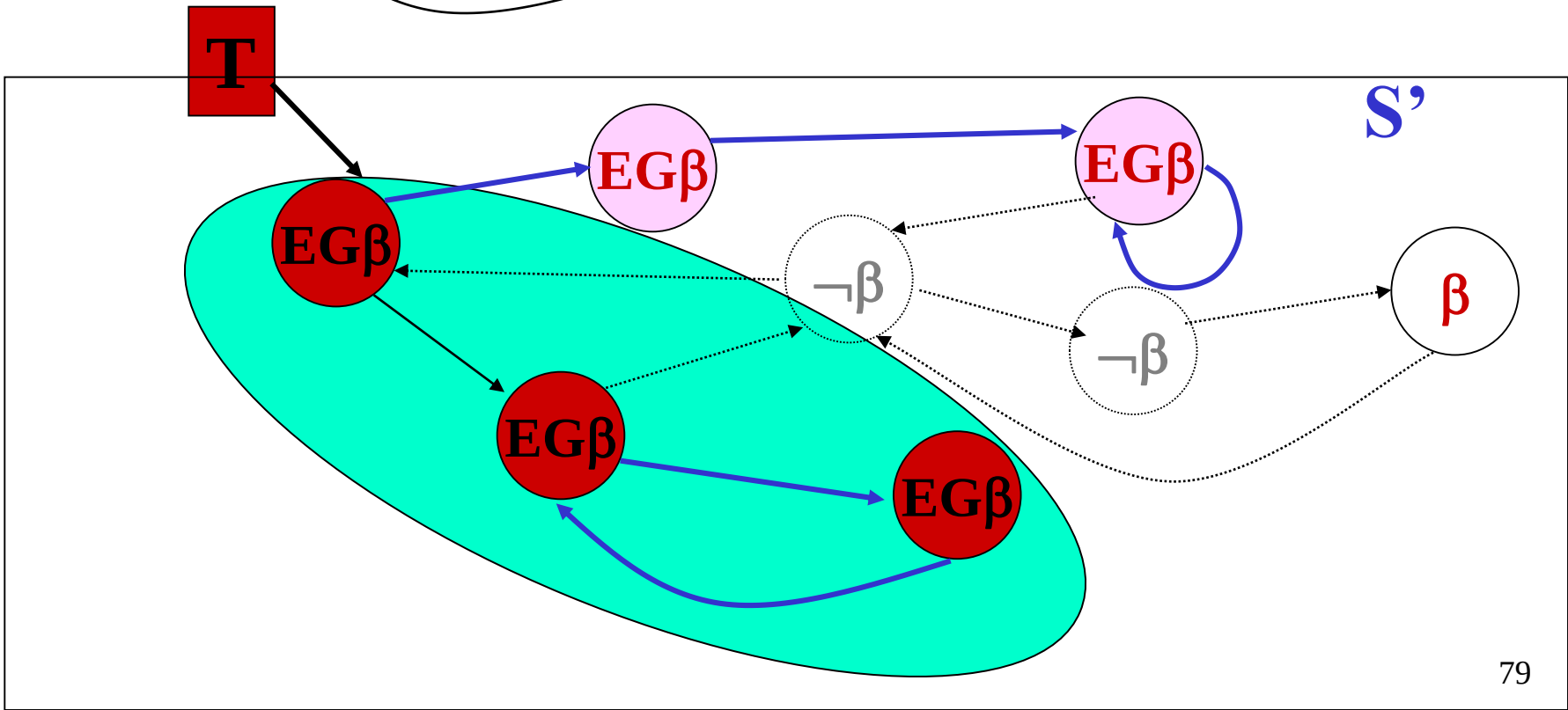

S’
β
β
β
¬β
¬β
β
β
β

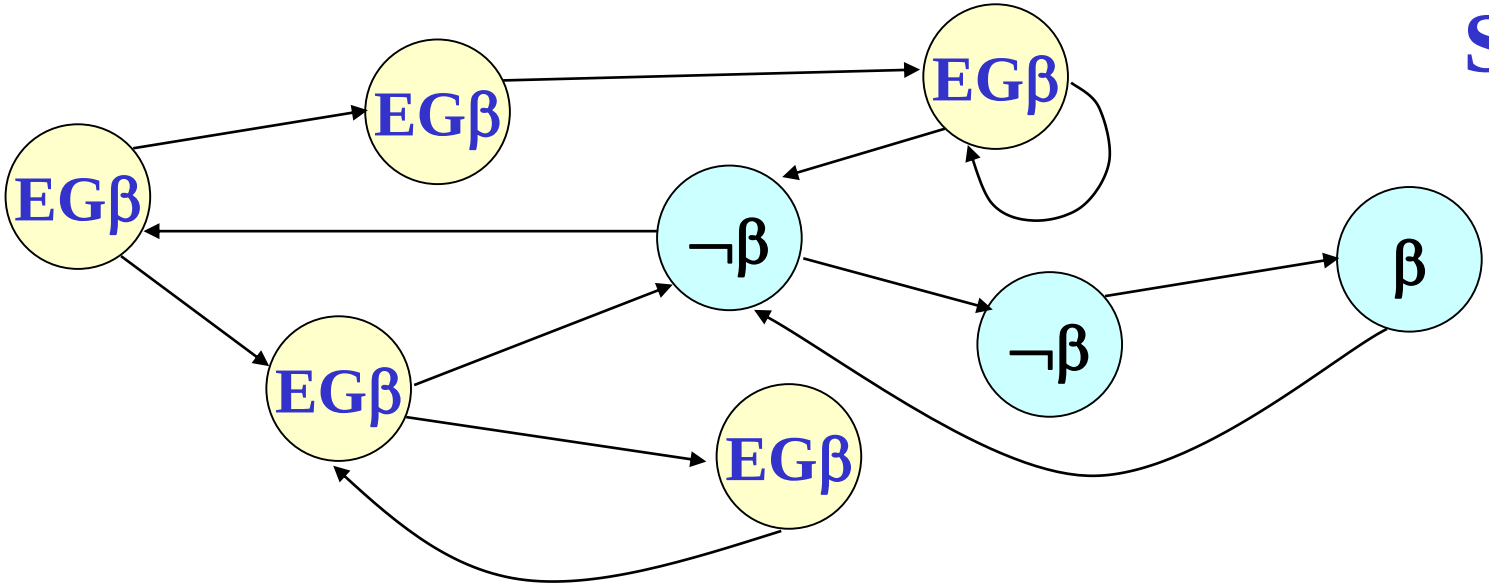

S
β
β
β
¬β
¬β
β
β
β
T
S'
β
β
β
¬β
¬β
β
β
β

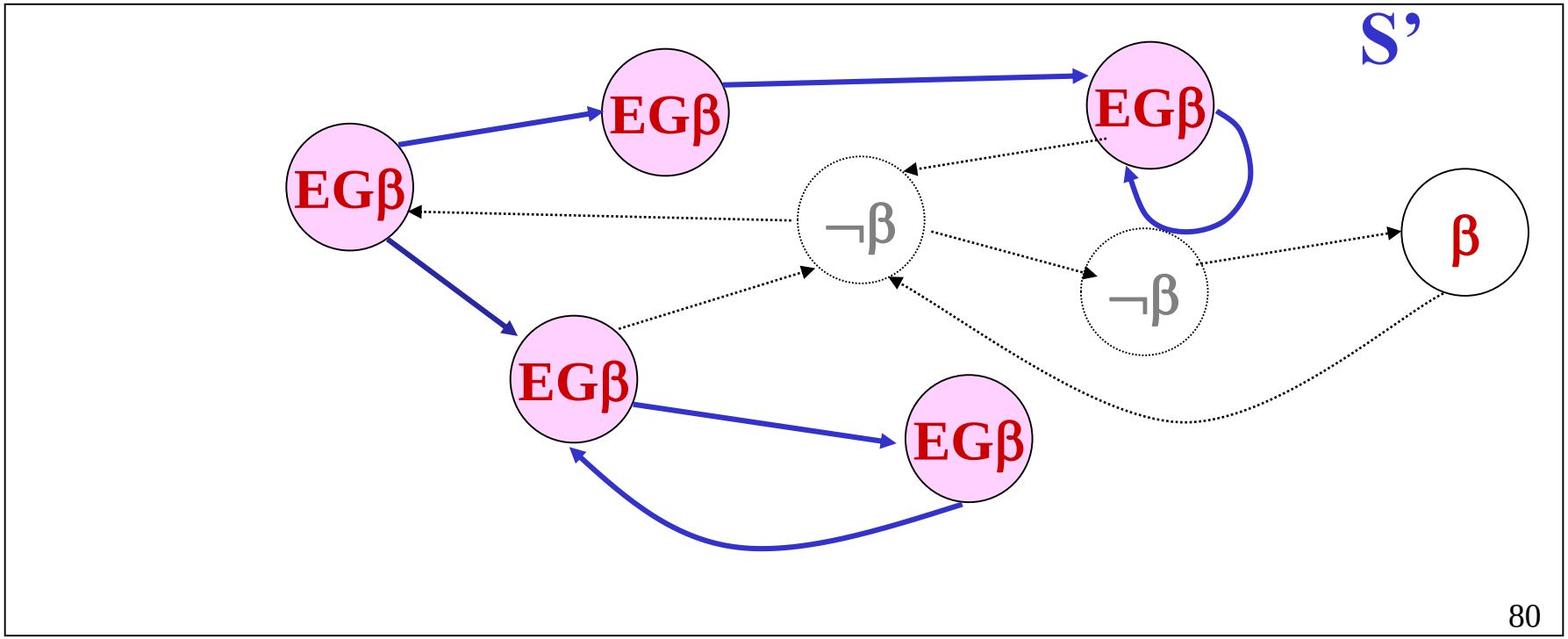

S
β
β
β
¬β
¬β
β
β
β
T
S'
β
β
EGβ
¬β
¬β
β
EGβ
EGβ


S
β
β
β
¬β
¬β
β
β
β
T
S'
EGβ
EGβ
β
¬β
¬β
β
EGβ
EGβ


S
β
β
β
¬β
¬β
β
β
β


T
S'
EGβ
EGβ
EGβ
EGβ
EGβ
¬β
¬β
β


S
EGβ
EGβ
EGβ
¬β
β
¬β
EGβ
EGβ


S'
EGβ
EGβ
EGβ
¬β
β
¬β
EGβ
EGβ

Computing the labeling for EG(β)

Algorithm Check_EG(β)

Complexity: O(|M|)

S' := {**s** | β ∈ Labels(**s**)};

SCC := {**C** | **C** is a non trivial **SCC** of **S'**};

T := $\bigcup_{C \in SCC}$ {**s** | **s** ∈ **C**};

for each **s** ∈ **T** do Labels(**s**) := Labels(**s**) ∪ {**EG(**β**)**};

while **T** ≠ ∅ do

chose **s** ∈ **T**;

T := **T** \{**s**};

for each **t** ∈ **S'** with **t** → **s** do

if **EG(**β**)** ∉ Lables(**t**) then

Labels(**t**) := Labels(**t**) ∪ {**EG(**β**)**};

T := **T** ∪ {**t**};

CTL model checking

- The algorithms just presented show that the ***model checking problem*** for ***CTL*** can be solved in ***time linear*** in the size of System **M** and the size of the Property ϕ, namely:

in time $\mathbf{O(|M| \cdot |\phi|)}$

where $|\mathbf{M}|$ is the ***size of the graph*** underlying **M** and $|\phi|$ is the ***number of subformulae*** of ϕ.

Fixed point characterization

- We will redefine the labeling function in terms of ***fixed point computation***.
- This is a ***nice*** and ***elegant*** algorithmic account.
- It will be used when ***efficient symbolic approach*** will be introduced.

Partial Orders

- A binary relation $\sqsubseteq$ on a set **A** is a ***partial order*** iff $\sqsubseteq$ is ***reflexive***, ***anti-symmetric*** and ***transitive***.
- The pair $\langle A, \sqsubseteq \rangle$ is called a ***partially ordered set*** (or ***poset***).
- **Example:** If **S** is any set and $\subseteq$ is the ordinary subset relation, then $\langle 2^S, \subseteq \rangle$ is a *partially ordered set*.

Upper Bounds

Given $\langle A, \sqsubseteq \rangle$ and $A' \subseteq A$

- $a \in A$ is an ***upper bound*** of A' iff $\forall a' \in A', a' \sqsubseteq a$
- $a \in A$ is a ***least upper bound*** (***lub***) of A', written $\sqcup A'$, iff
  - a is an ***upper bound*** of A' and
  - $\forall a' \in A$, if a' is an ***upper bound*** of A', then $a \sqsubseteq a'$

Lower Bounds

Given $\langle A, \sqsubseteq \rangle$ and $A' \subseteq A$

- $a \in A$ is a ***lower bound*** of A' iff $\forall a' \in A', a \sqsubseteq a'$
- $a \in A$ is a ***greatest lower bound*** (***glb***) of A', written $\sqcap A'$, iff
  - a is a ***lower bound*** of A' and
  - $\forall a' \in A$, if a' is a ***lower bound*** of A', then $a' \sqsubseteq a$

Complete Lattice

A ***poset*** **<A, ⊑>** is a ***complete lattice*** if, for each **A' ⊆ A**, the ***greatest lower bound*** **⊓A'** and the ***least upper bound*** **⊔A'** do exist.

A ***complete lattice*** **<A, ⊑>** has a unique ***greatest element*** **⊔A=⊤** and also a unique ***least element*** **⊓A = ⊥**.

Complete Lattice

The ***poset*** $\langle 2^S, \subseteq \rangle$ is a ***complete lattice*** where ***intersection*** $\cap$ and ***union*** $\cup$ correspond to $\sqcap$ and $\sqcup$, respectively.

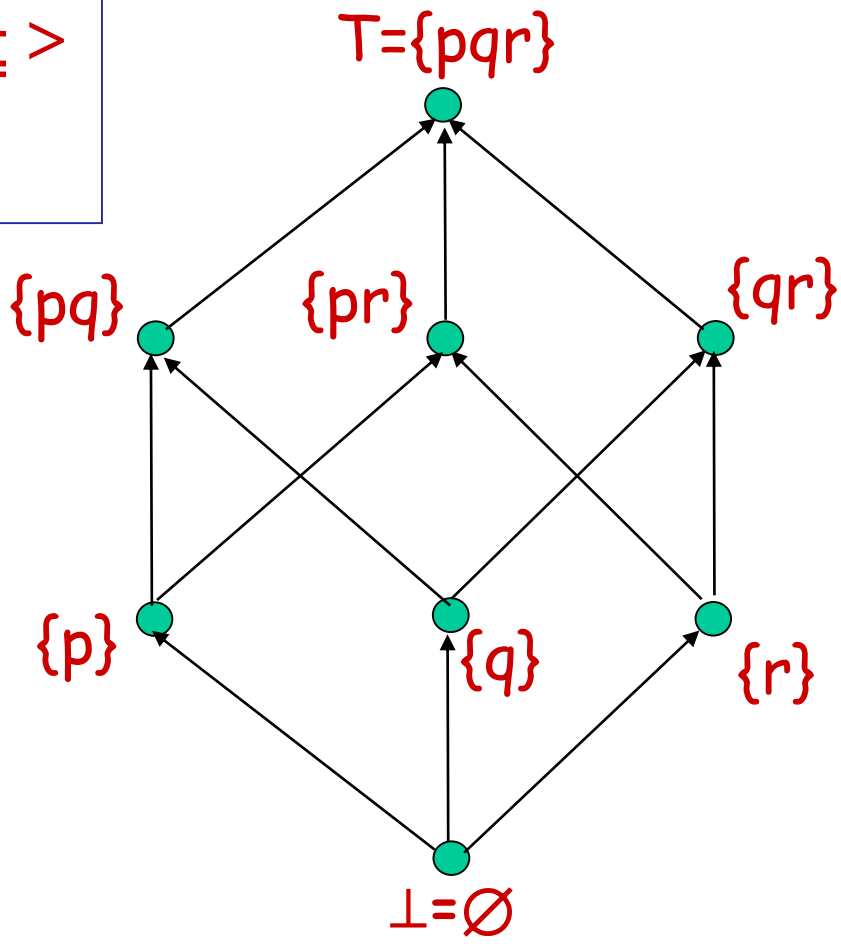
Any two subset of **S** have a ***least upper*** and a ***greatest lower bound***.

Example: **S={a,b,c,d}**. For **{a,c}** and **{b,c}** the ***glb*** is **{c}**, while the ***lub*** is **{a,b,c}**.

There is a unique ***greatest element*** $\cup 2^S = S$ and a unique ***least element*** $\cap 2^S = \varnothing$.

Example of a complete lattice

The complete lattice $<2^S, \subseteq>$ when S is the set {p,q,r}.



Monotonic functions

- A function **F: A ⟶ A** is ***monotonic*** if for each **a,b ∈ A**, **a ⊑ b** implies **F(a) ⊑ F(b)**.
- In other words, a function **F** is monotonic if it ***preserves the ordering*** ⊑.

Fixed points

- Given a function **F: A ⟶ A**, an element **a ∈ A** is a ***fixed point*** of **F** if **F(a) = a**.
- **a ∈ A** is called the ***least fixed point*** of **F (μx.F(x))**, if for all **a' ∈ A** such that **F(a') = a'**, then **a ⊑ a'**.
- **a ∈ A** is called the ***greatest fixed point*** of **F (νx.F(x))**, if for all **a' ∈ A** such that **F(a') = a'**, then **a' ⊑ a**.

Tarski's Fixed Point theorem

THEOREM: Let **<A, ⊑>** be a ***complete lattice***, and **F: A ⟶ A** a monotonic function. Then **F** has a ***least*** and a ***greatest fixed point*** given, respectively, by:

- **μx.F(x) = ⊓{x ∈ A | F(x) ⊑ x}**
- **νx.F(x) = ⊔{x ∈ A| x ⊑ F(x)}**

Fixed point in finite lattices

Let $\langle A, \sqsubseteq \rangle$ be a ***finite complete lattice***, and $F: A \rightarrow A$ be a monotonic function.

The ***least element*** of **A**

Then the ***least fixed point*** for **F** is obtained as

$$\mu x.F(x) = F^m(\bot)$$

for some **m**, where $F^0(\bot) = \bot$, and $F^{n+1}(\bot) = F(F^n(\bot))$.

Moreover, the ***greatest fixed point*** for **F** is obtained as

$$\nu x.F(x) = F^k(\top)$$

for some **k**, where $F^0(\top) = \top$, and $F^{n+1}(\top) = F(F^n(\top))$.

The ***greatest element*** of **A**

Generic fixed point algorithm

Algorithm Compute_lfp(**F**:function)

$\mathbf{X_0} := \bot$;

$\mathbf{X_1} := \mathbf{F}(\mathbf{X_0})$;

j=1;

while $\mathbf{X_j} \neq \mathbf{X_{j-1}}$

$\mathbf{j} := \mathbf{j+1}$;

$\mathbf{X_j} := \mathbf{F}(\mathbf{X_{j-1}})$;

return $\mathbf{X_j}$

CTL and complete lattices

- Given a Kripke structure $\mathbf{M=<S,S_0,R,L,AP>}$. We will then consider the ***poset*** $\mathbf{<2^S,\subseteq>}$.
- $\mathbf{<2^S,\subseteq>}$ is clearly a ***complete lattice*** (with respect to intersection and union).
- We will identify a ***CTL formula*** with the ***set of states*** which ***satisfy it***.
- In this way we can define ***temporal operators*** as ***functions*** on the ***complete lattice*** $\mathbf{<2^S,\subseteq>}$.

Denotation of a CTL formula

- Given a formula ϕ, let us define its ***denotation*** (in **M**), in symbols $|[\phi]|$, as the set of states satisfying the formula:

$$|[\phi]| = \{ s \mid M,s \vDash \phi \}$$

- We could then define the cpo $<CTL, \sqsubseteq>$ by:

$$\phi \sqsubseteq \psi \quad \textit{iff} \quad |[\phi]| \subseteq |[\psi]|$$

Denotation of a CTL formula

- Given the ***denotation*** of a formula

$$|[\phi]| = \{ s \mid M,s \models \phi \}$$

- We could then define the cpo $\langle CTL, \sqsubseteq \rangle$ by:

$$\phi \sqsubseteq \psi \quad \textit{iff} \quad |[\phi]| \subseteq |[\psi]|$$

- Then $|[\bot]| = \varnothing$; $|[\top]| = S$;
- $|[p]| = \{ s \mid p \in L(s) \}$;
- $|[\neg\phi]| = S \setminus |[\phi]|$;
- $|[\phi \vee \psi]| = |[\phi]| \cup |[\psi]|$;
- $|[\phi \wedge \psi]| = |[\phi]| \cap |[\psi]|$;

CTL is closed under ***conjunction*** and ***disjunction***, therefore for any pair of formulae the ***upper*** and ***lower bound*** do exist.

Denotation of a CTL formula

- Given a formula ϕ, let us define its ***denotation*** (in **M**), in symbols $|[\phi]|$, as the set of states satisfying the formula:

$$|[\phi]| = \{ s \mid M,s \vDash \phi \}$$

- ….
- $|[EX\phi]| = \{ s \mid \exists t. (t \in |[\phi]| \cap R(s)) \}$
- for the other **temporal operators** we would need to use **fixed points….**

Fixed point characterization of $\mathbf{EU(\beta_1,\beta_2)}$

- $\mathbf{EU(\beta_1,\beta_2) \equiv \beta_2 \vee (\beta_1 \wedge EX\ EU(\beta_1, \beta_2))}$
- $\mathbf{|[EU(\beta_1,\beta_2)]| = \mu Z.(|[\beta_2]| \cup (|[\beta_1]| \cap |[EX\ Z]|))}$
- $\mathbf{|[EU(\beta_1,\beta_2)]| =}$

$$\mathbf{\mu Z.(|[\beta_2]| \cup (|[\beta_1]| \cap \{ s \mid \exists t \in Z \cap R(s) \}))}$$

Fixed point characterization of $\mathbf{EU(\beta_1,\beta_2)}$

Lemma: Let

$$\mathbf{F(Z) = (|[\beta_2]| \cup (|[\beta_1]| \cap \{ s \mid \exists t \in Z \cap R(s) \}))}$$

then **F** is a ***monotonic function***, i.e.

$$\mathbf{Z_1 \subseteq Z_2 \text{ implies } F(Z_1) \subseteq F(Z_2)}$$

Fixed point characterization of $\mathbf{EU(\beta_1,\beta_2)}$

Theorem:

$$|[\mathbf{EU}(\beta_1,\beta_2)]| = \mu Z.(|[\beta_2]| \cup (|[\beta_1]| \cap \{ s \mid \exists t \in Z \cap R(s) \}))$$

in other words:

$$\mu Z.(|[\beta_2]| \cup (|[\beta_1]| \cap \{ s \mid \exists t \in Z \cap R(s)\})) \subseteq |[\mathbf{EU}(\beta_1,\beta_2)]|$$

and

$$|[\mathbf{EU}(\beta_1,\beta_2)]| \subseteq \mu Z.(|[\beta_2]| \cup (|[\beta_1]| \cap \{ s \mid \exists t \in Z \cap R(s)\}))$$

Computing fixed point for $EU(\beta_1,\beta_2)$

Algorithm Compute_EU(β_1,β_2)

$X_0 := |[\bot]|$; /* i.e. $X_0 := \emptyset$ */

$X_1 := |[\beta_2]| \cup (|[\beta_1]| \cap X_0)$; /* i.e. $X_1 := |[\beta_2]|$ */

j=1;

while $X_j \neq X_{j-1}$

$j := j+1$; $T := X_{j-1}$; $X := \emptyset$

while $T \neq \emptyset$ do

chose $s \in T$;

$T := T \setminus \{s\}$;

forall t such that $s \in R(t)$ do

$X := X \cup \{t\}$;

$X_j := |[\beta_2]| \cup (|[\beta_1]| \cap X)$

This computes $X = EX\ X_{j-1}$

Computing fixed point for $\mathbf{EU(\beta_1,\beta_2)}$

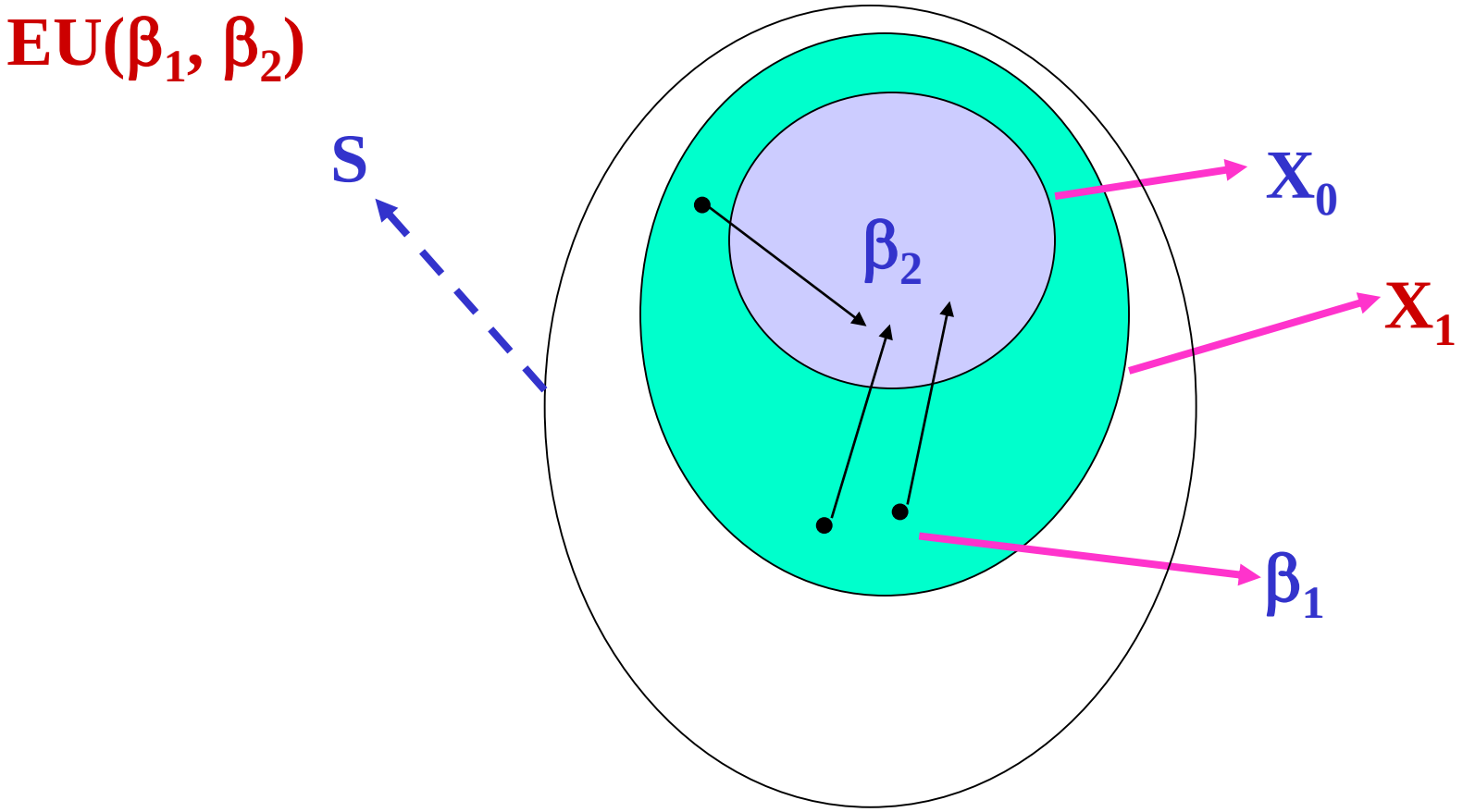
To compute $\mathbf{|[EU(\beta_1,\beta_2)]|}$ we can ***construct inductively the set*** of states $\mathbf{X_j}$ as follows:

- $\mathbf{X_1 = |[\beta_2]|}$.
- $\mathbf{X_{j+1} = X_j \cup \{ s \mid s \in |[\beta_1]|}$ and $\mathbf{R(s,t)}$ for some $\mathbf{t \in X_j\}}$

$\mathbf{|[EU(\beta_1,\beta_2)]|}$ is then the set $\mathbf{X}$ such that $\mathbf{X = X_n}$ for $\mathbf{n}$ such that $\mathbf{X_{n+1} = X_n}$.

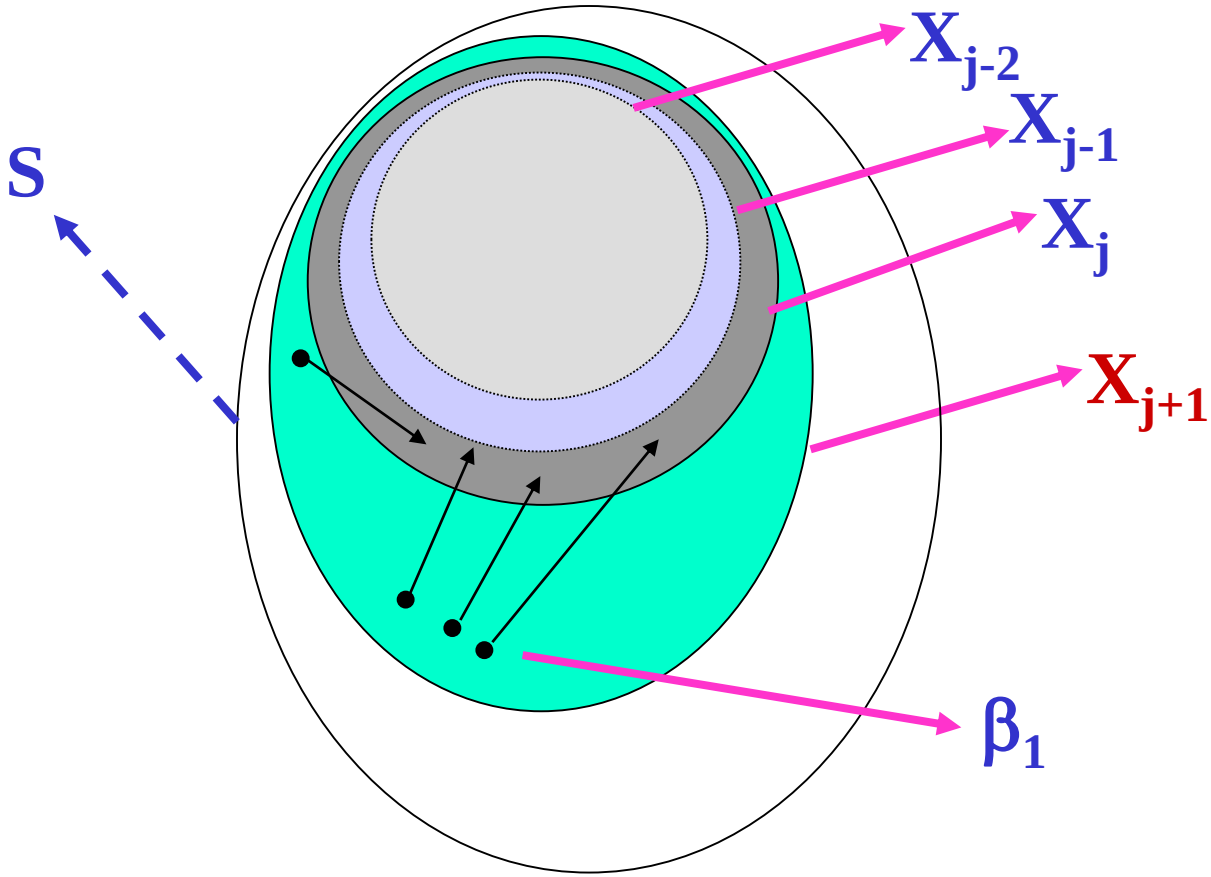
Notice that $\mathbf{n}$ ***must exist*** by ***Tarski's Theorem*** since $\mathbf{X_j \subseteq X_{j+1} \subseteq S}$ (and $\mathbf{S}$ is finite!)

From X_0 to X_1



From X_j to X_{j+1}

$EU(\beta_1, \beta_2)$



Computing fixed point for $EU(\beta_1,\beta_2)$

Algorithm Compute_EU(β_1,β_2)

$X_1 := |[\beta_2]|$;

j=1;

repeat

$j := j+1$; $T := X := X_{j-1}$;

while $T \neq \emptyset$ do

chose $s \in T$;

$T := T \setminus \{s\}$;

forall t such that $s \in R(t)$ do

if $t \in |[\beta_1]|$ then /* $t \in |[\beta_1]| \cap EX\, X_{j-1}$ */

$X := X \cup \{t\}$;

$X_j = X$;

until $X_{j-1} = X_j$

Fixed point characterization of EG(β)

- **$EG(\beta) \equiv \beta \wedge EX\ EG(\beta)$**
- **$|[EG(\beta)]| = \nu Z.((|[\beta]| \cap |[EX\ Z]|)\ )$**
- **$|[EG(\beta)]| =$**

  $\nu Z.((|[\beta]| \cap \{\ s \mid \exists t \in Z \cap R(s)\ \})$

Computing the fixed point for EG(β)

Algorithm Compute_EG(β)

$X_0 := |[\top]|$; /* i.e. $X_0 := S$ */

$X_1 := |[\beta]| \cap X_0$; /* i.e. $X_1 := |[\beta]|$ */

$j=1$;

while $X_j \neq X_{j-1}$

$j := j+1$; $T := X_{j-1}$; $X := \emptyset$;

while $T \neq \emptyset$ do

chose $s \in T$;

$T := T \setminus \{s\}$;

forall t such that $s \in R(t)$

$X_j := X_j \cup \{t\}$;

$X_j := |[\beta]| \cap X_j$

$X = EX\ X_{j-1}$

The Labels function

- To compute $|[EG\beta]|$ we can ***construct inductively the set*** of states X_j as follows:

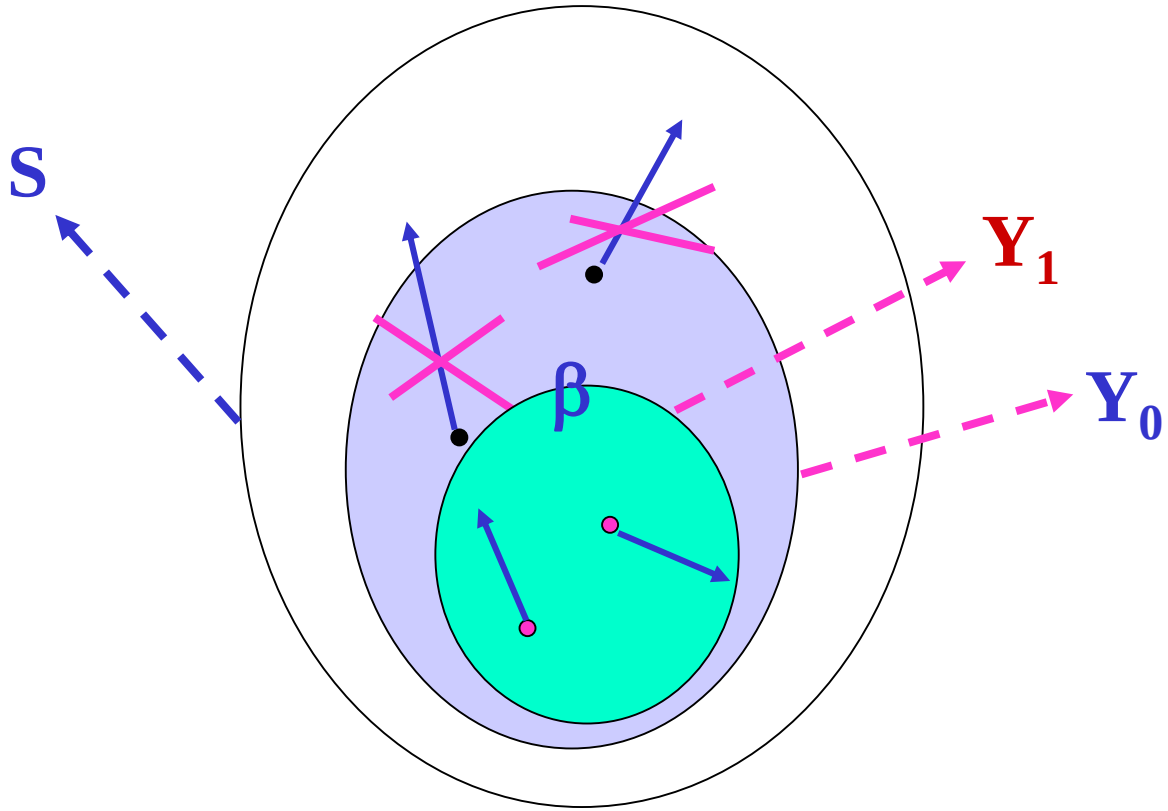
  - $X_1 = |[\beta]|$.
  - $X_{j+1} = X_j - \{ s \mid s \in X_j$ and ***there does not exist*** $t \in X_j$ such that $R(s, t)\}$

$|[EG\beta]|$ is then the set X such that $X = X_n$ for m such that $X_{m+1} = X_m$.

- Notice that m ***must exists*** by ***Tarski's Theorem*** since $\emptyset \subseteq X_{j+1} \subseteq X_j$

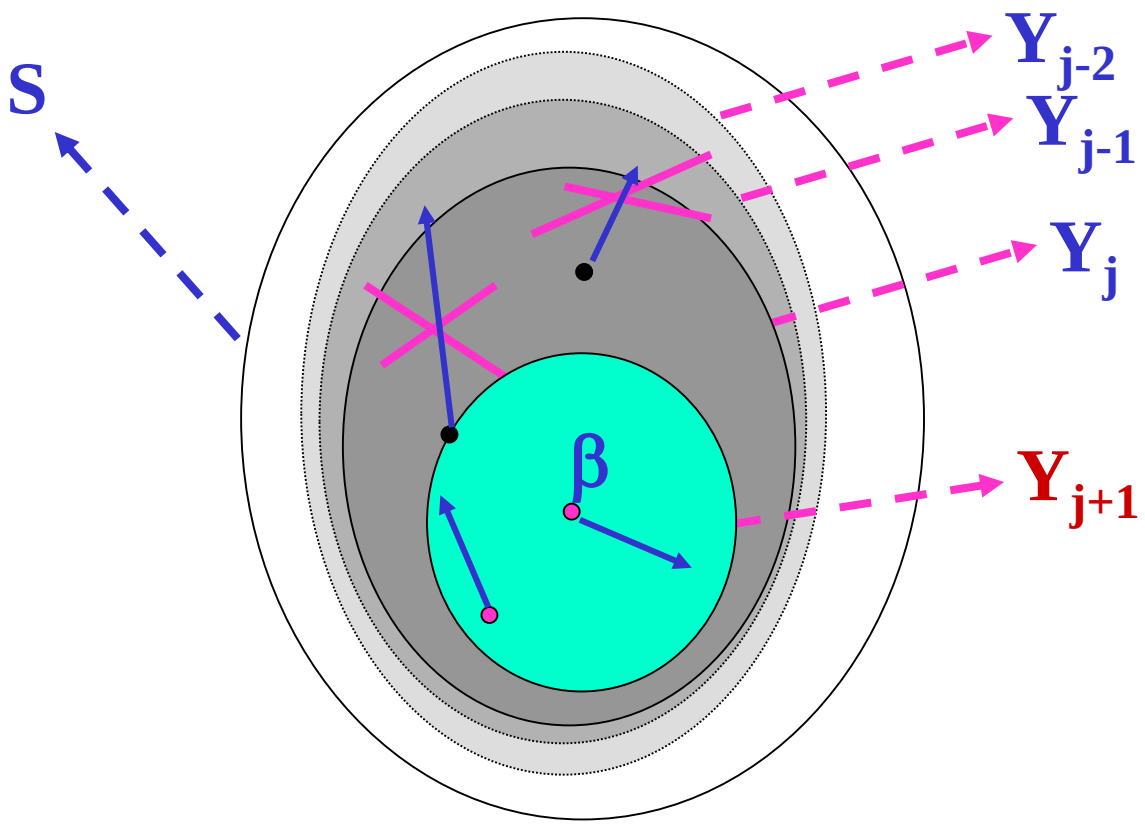
From Y_0 to Y_1

EGβ



From Y_j to Y_{j+1}

EGβ



Computing the fixed point for EG(β)

Algorithm Compute_EG(β)

$X_1 := |[\beta]|$;

$j=1$;

repeat

$j := j+1$; $T := X_j := X_{j-1}$;

while $T \neq \emptyset$ do

chose $s \in T$;

$T := T \setminus \{s\}$;

if for all $t \in R(s)$, $t \notin X_{j-1}$ then

$X_j := X_j - \{s\}$;

until $X_j = X_{j-1}$;