Tecniche di Specifica e di Verifica

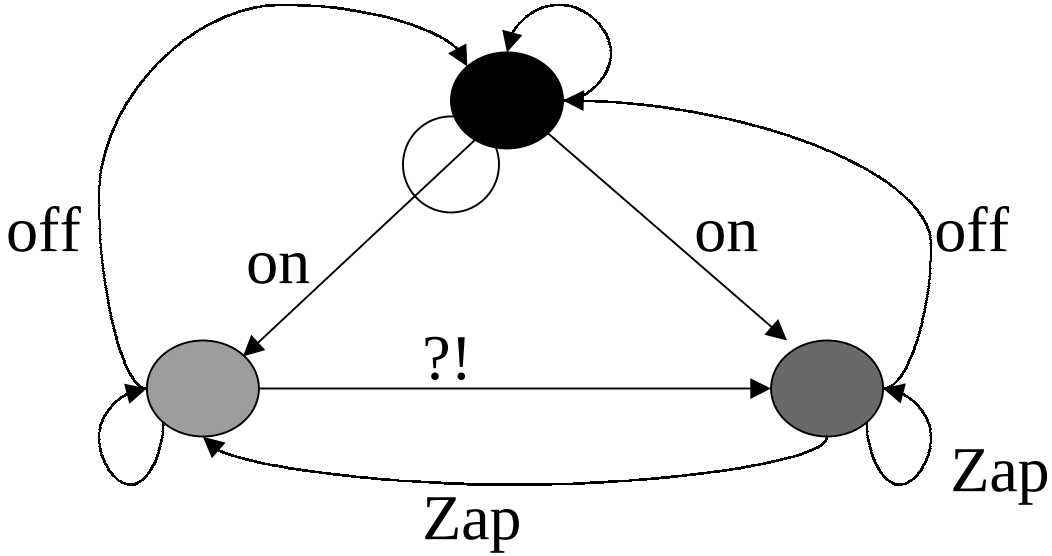
Branching Time Temporal Logics

Outline

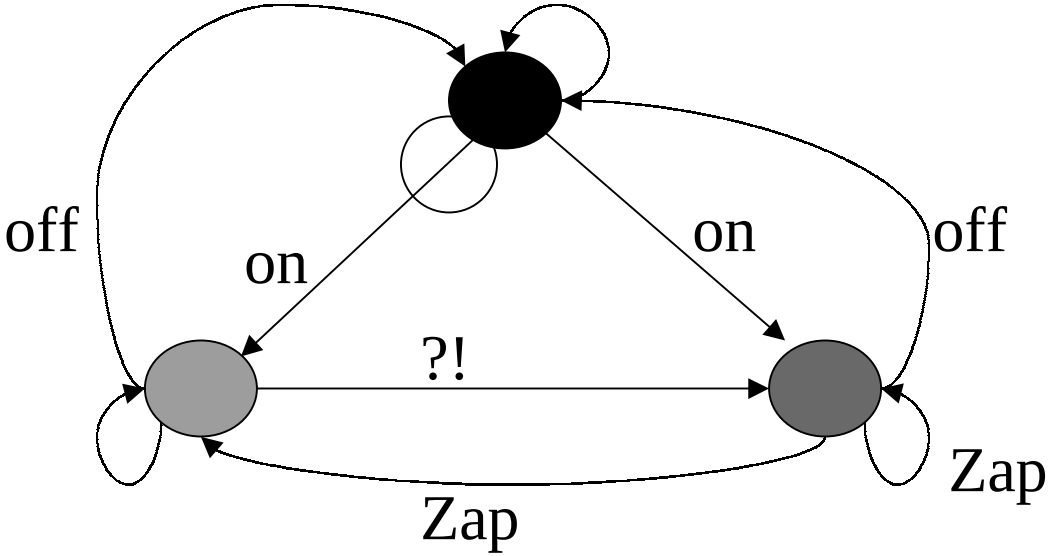
- ***CTL*** (**C**omputation **T**ree **L**ogic)
  - **Branching Time**
  - Unwindings --- computation trees
  - Syntax and semantics of CTL.

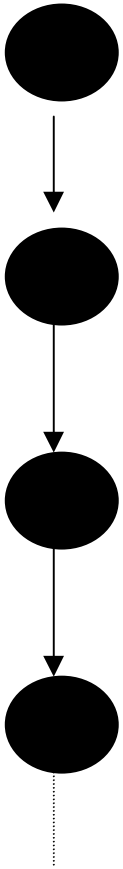
Branching Time Structures

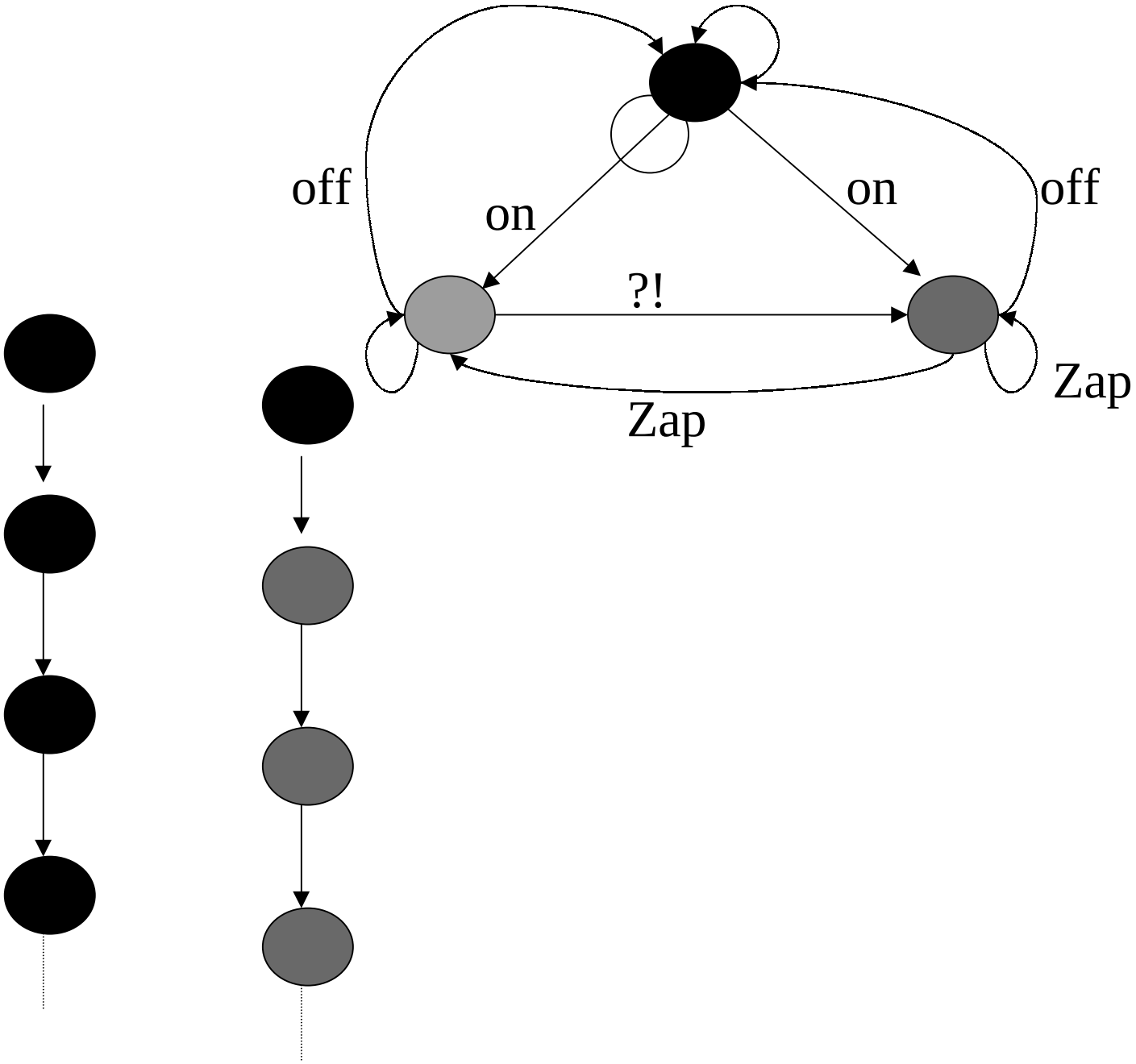
- **Linear Time**:
  - A ***computation*** at its first state satisfies a property.
  - Property **---- LTL** formula
- **Branching Time**
  - The ***computation tree*** at its root satisfies a property.
  - Property: **CTL** (**CTL*, μ-calculus**) formula.
  - **Computation Tree**
    - **All *computations***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

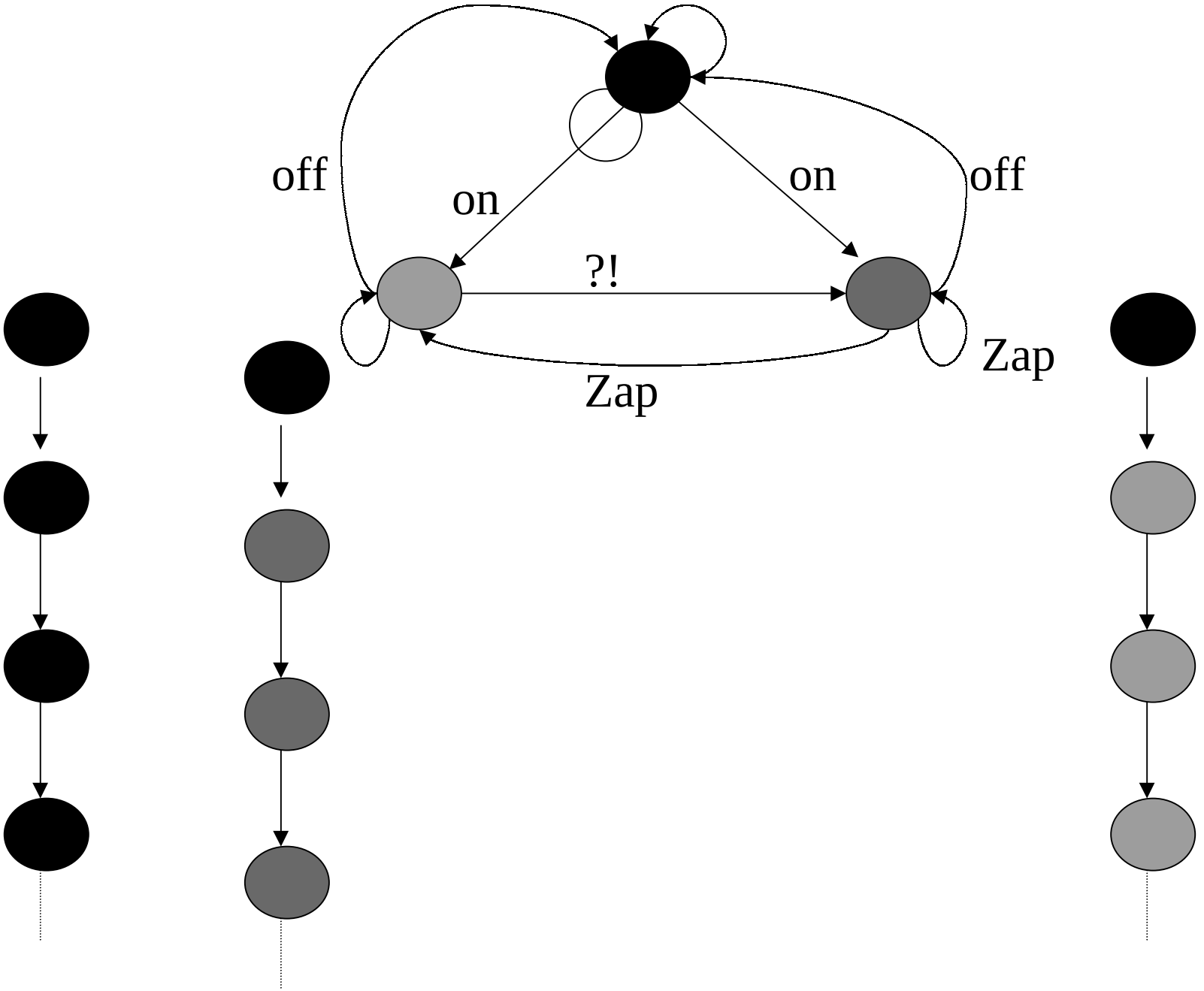


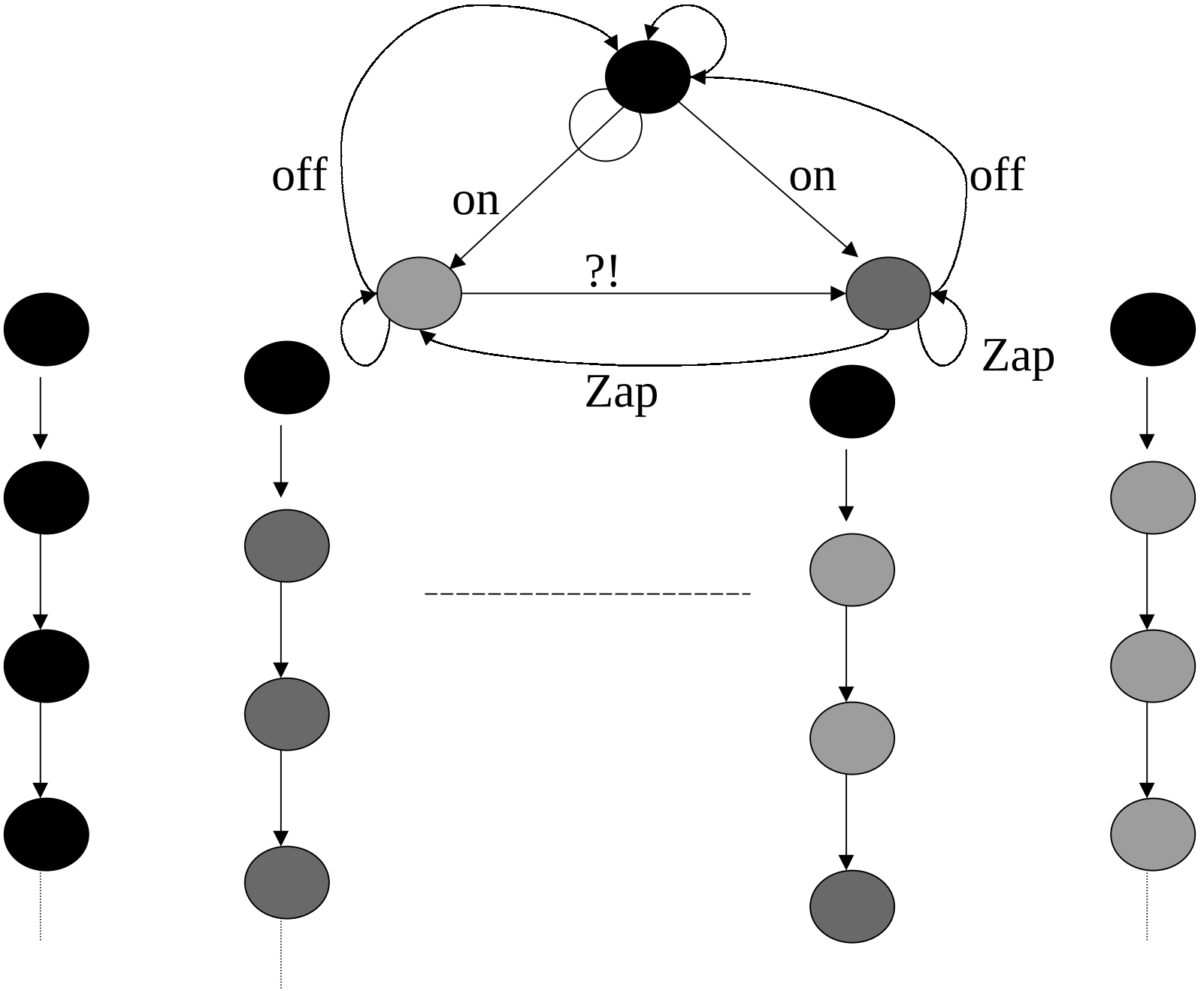
The TV Example


off
on
on
off
?!
Zap
Zap

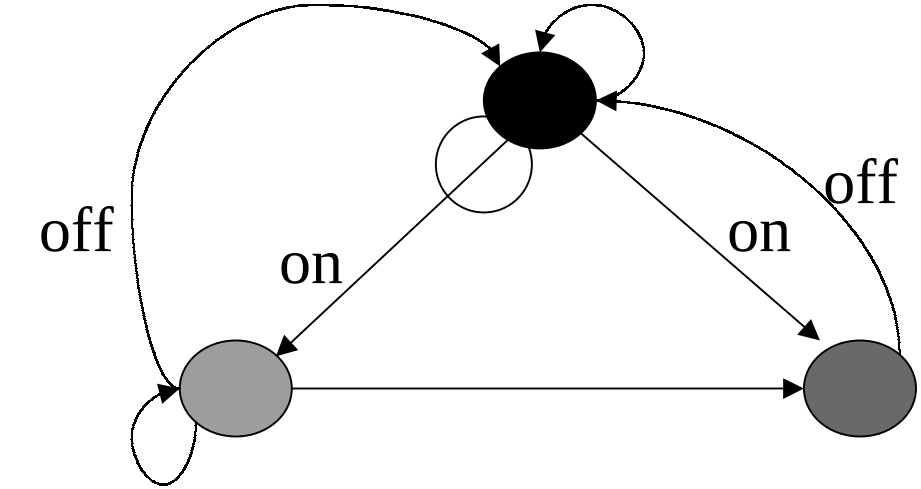


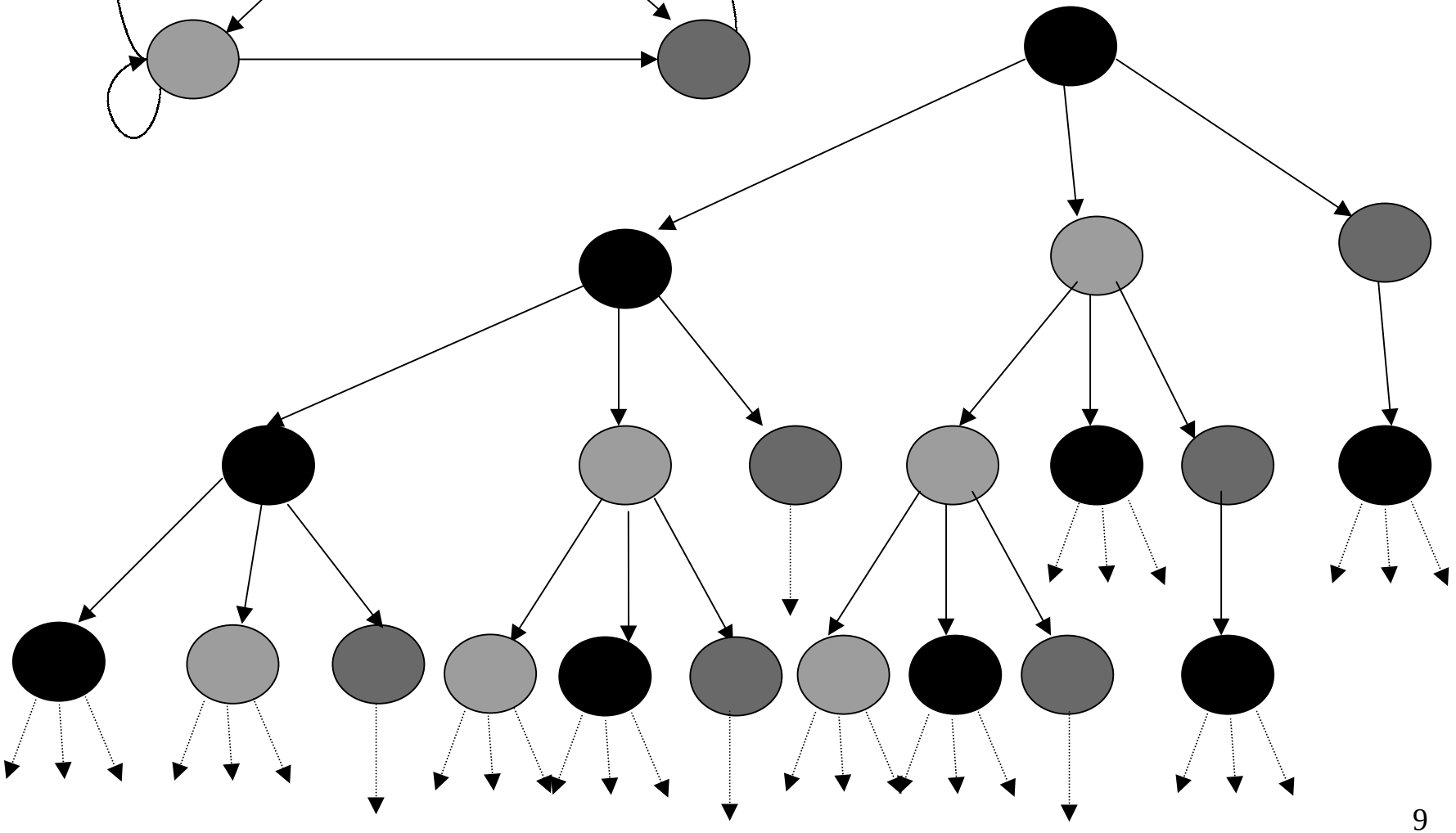

off
on
on
off
?!
Zap
Zap


off
on
on
off
?!
Zap
Zap

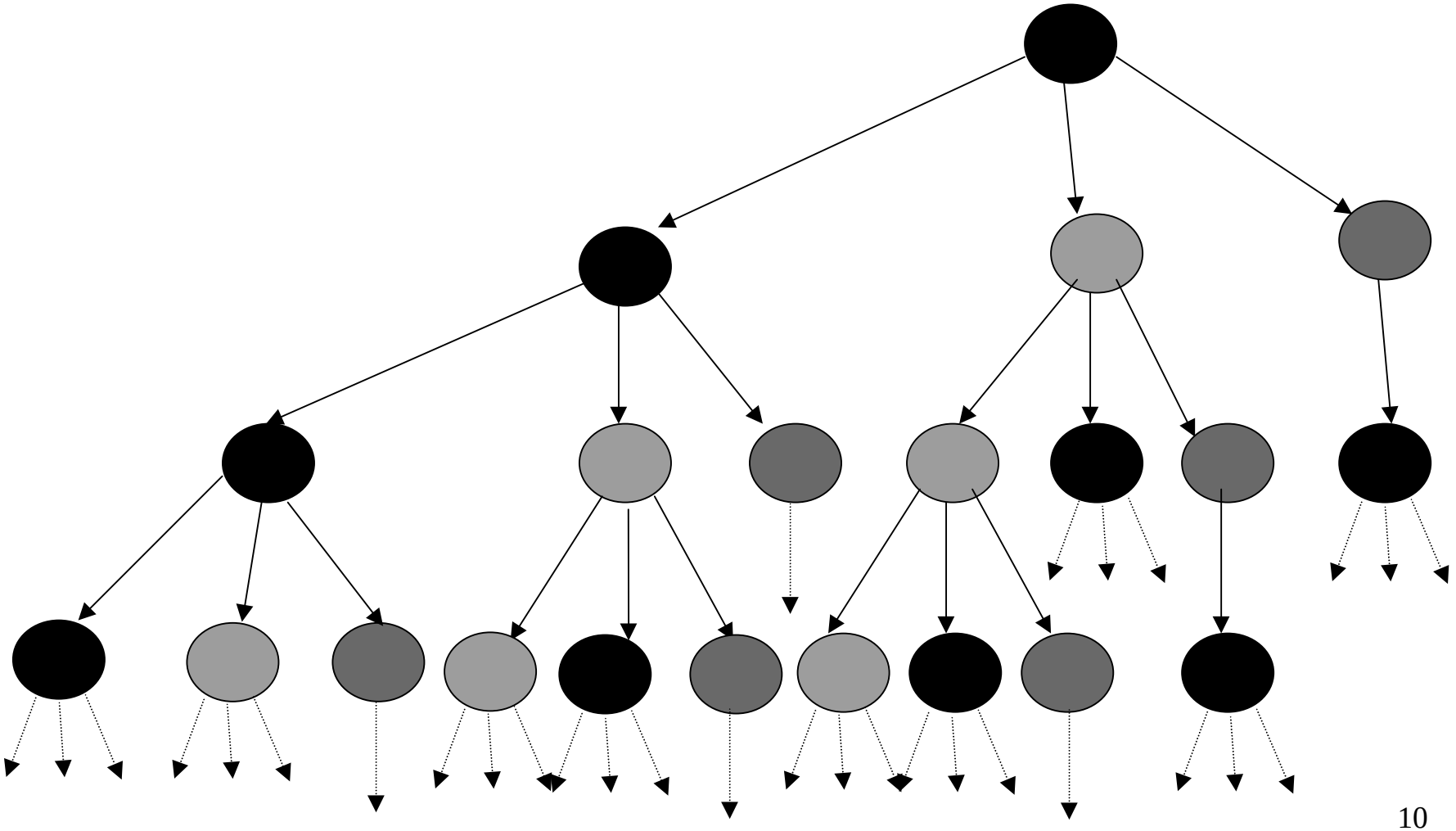

off
on
on
off
?!
Zap
Zap

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**

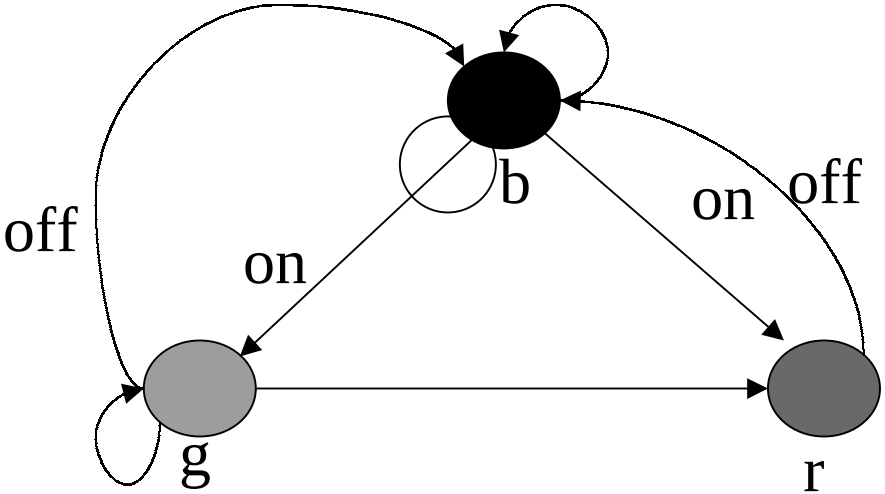
Unwinding

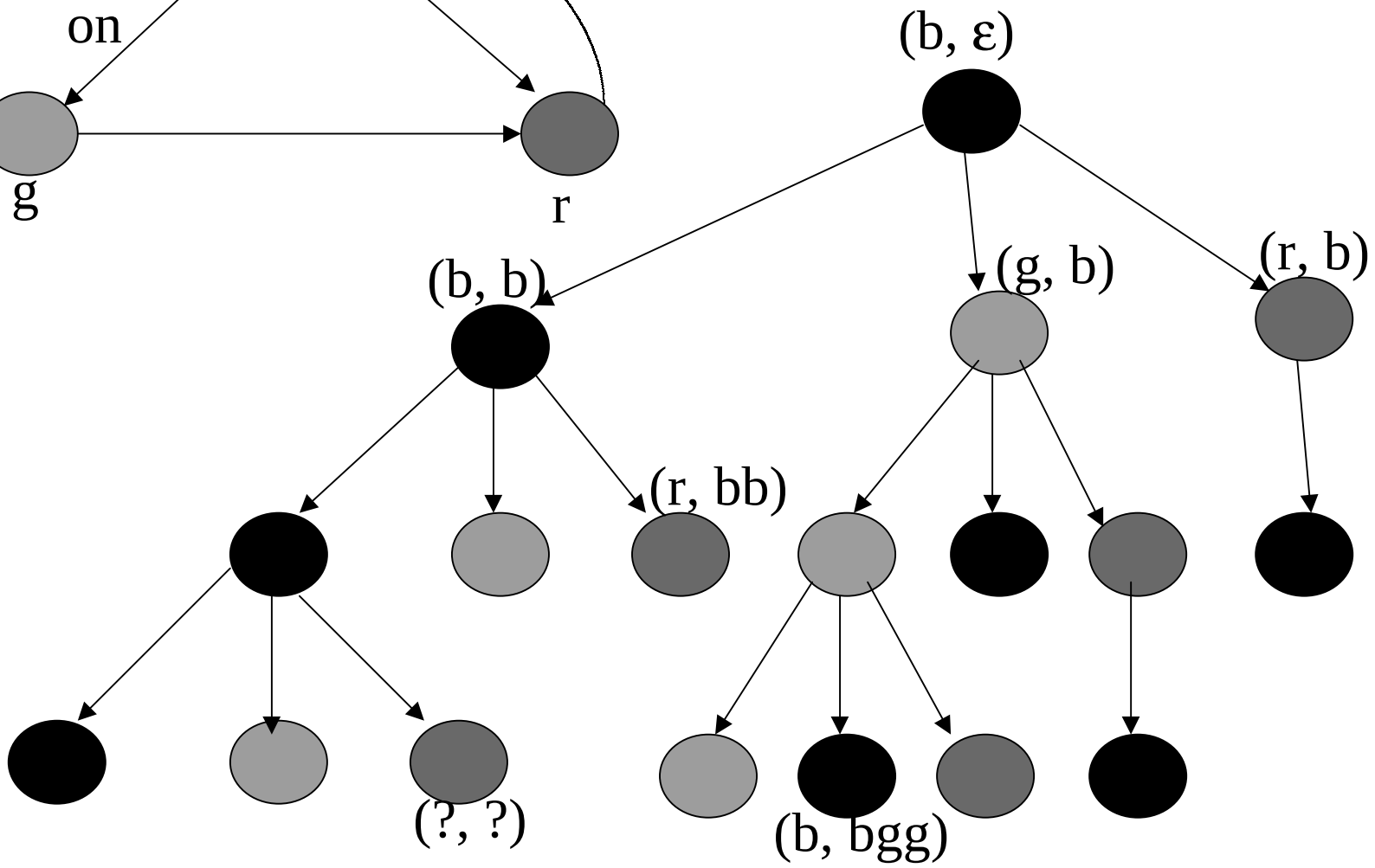
- **K = (S, S_0, R, AP, L)** **s** $\in$ **S**
- **TR(K, s)** --- The computation tree rooted at **s**.
- **TR(K, s) = ($\mathcal{S}_s$, (s,ε), $\mathcal{R}_s$, AP, $\mathcal{L}_s$).**
  - **(s, ε)** $\in \mathcal{S}_s$;
  - For any **(s', σ)** $\in \mathcal{S}_s$, **s'**$\in$ **S** and **σ = s s_1 … s_n** is a path in **K** leading from **s** to **s'** (i.e. **σ.s'** is a path from **s** to **s'**);
  - If **(s_1, σ)** $\in \mathcal{S}_s$ and **R(s_1, s_2)** then **(s_2, σ.s_1)** $\in \mathcal{S}_s$ and $\mathcal{R}_s$**((s_1, σ), (s_2, σ.s_1))** ;
  - $\mathcal{L}$**((s_1, σ)) = L(s_1)**.

Unwinding

- **TR(K, s)** is almost a Kripke structure.
  - $\mathcal{S}_s$ may 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*** at **s**.

A Modified Example





Linear time Vs Branching time

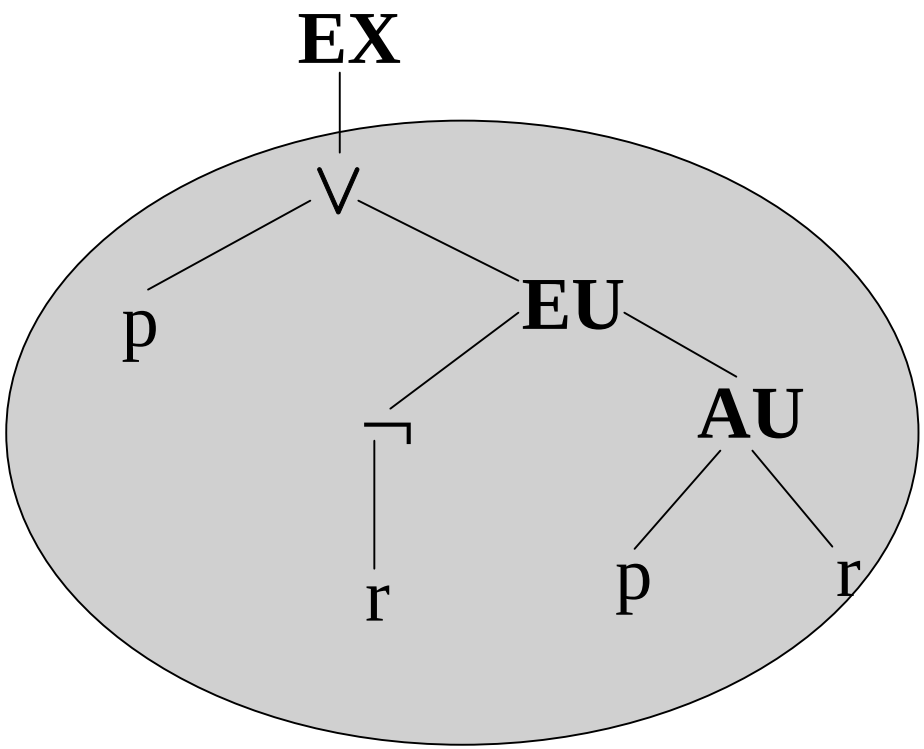
- There are ***properties*** that can be ***expressed in* LTL** but which ***can not be expressed in* CTL**. (sloppy statement!)
- There are ***properties*** that can be ***expressed in* CTL** but ***not in* LTL**.
- The **LTL *model checking problem*** can be ***converted*** into a ***restricted*** 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 $\rightarrow$ 2^{AP}**
- ψ a **CTL** formula **s** $\in$ **S**
- **K, s** $\models$ ψ
- ψ (holds) ***is satisfied*** at **s**.
- **FACT**:

  K, s $\models$ ψ iff **TR(K, s), (s,** ε**)** $\models$ ψ.

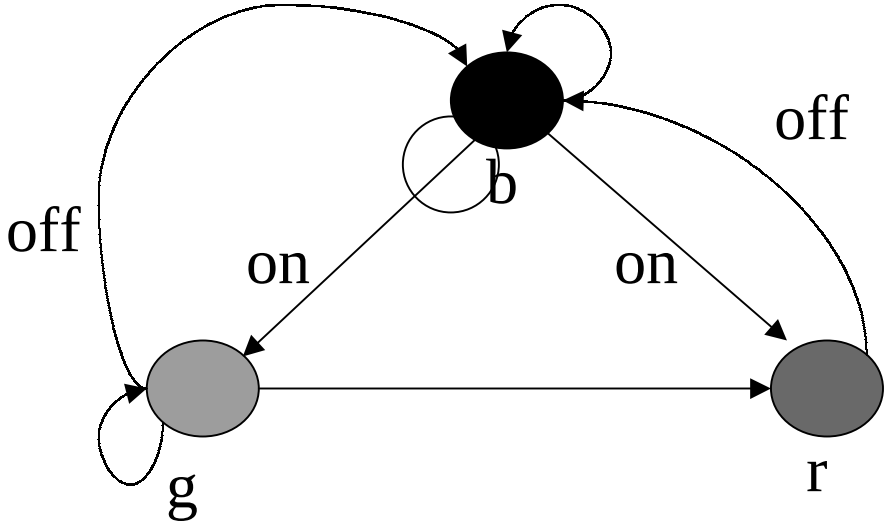
Semantics

- **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: $\rightarrow$ 2^{AP} ; s $\in$ S**
- **K, s $\models$ p** iff **p $\in$ L(s).**
- **K, s $\models$ $\neg\psi$** iff **K, s $\not\models$ ψ**
- **K, s $\models$ $\psi_1 \vee \psi_2$** iff
  K, s $\models$ ψ_1 or **K, s $\models$ ψ_2.**

Semantics

- **CTL ::= p** $\mid \neg\psi \mid \psi_1 \vee \psi_2 \mid$ **EX(**ψ**)** $\mid$
  $\mid$ **EU(**ψ_1**,** ψ_2**)** $\mid$ **AU(**ψ_1**,** ψ_2**)**
- **K = (S, S$_0$, R, AP, L)** ; **L:** $\rightarrow$ 2^{AP} ; $s \in S$
- **K, s** $\models$ **EX(**ψ**)** iff there exists **s'** such that:
  - **s** $\rightarrow$ **s'** (i.e. **R(s, s')**) and **K, s'** $\models \psi$

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



AP = {n, h, uh}

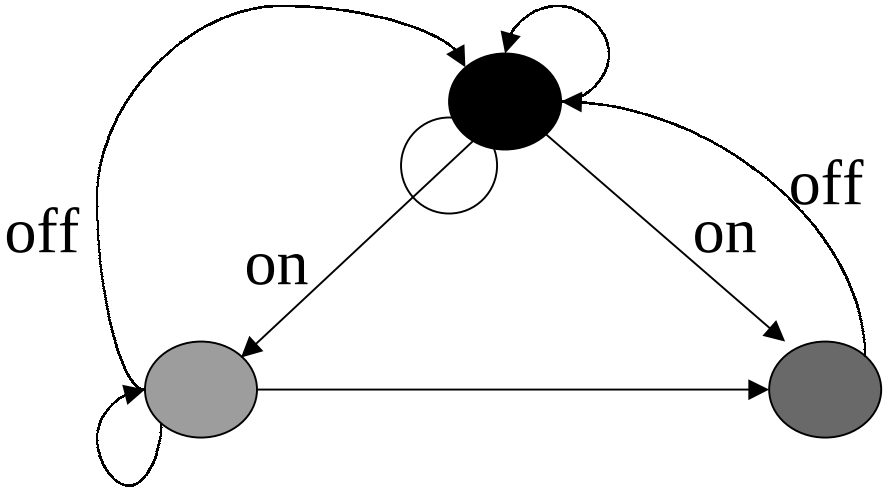
K, b ⊨ EX(uh) ? K, b ⊨ EX(¬uh) ?

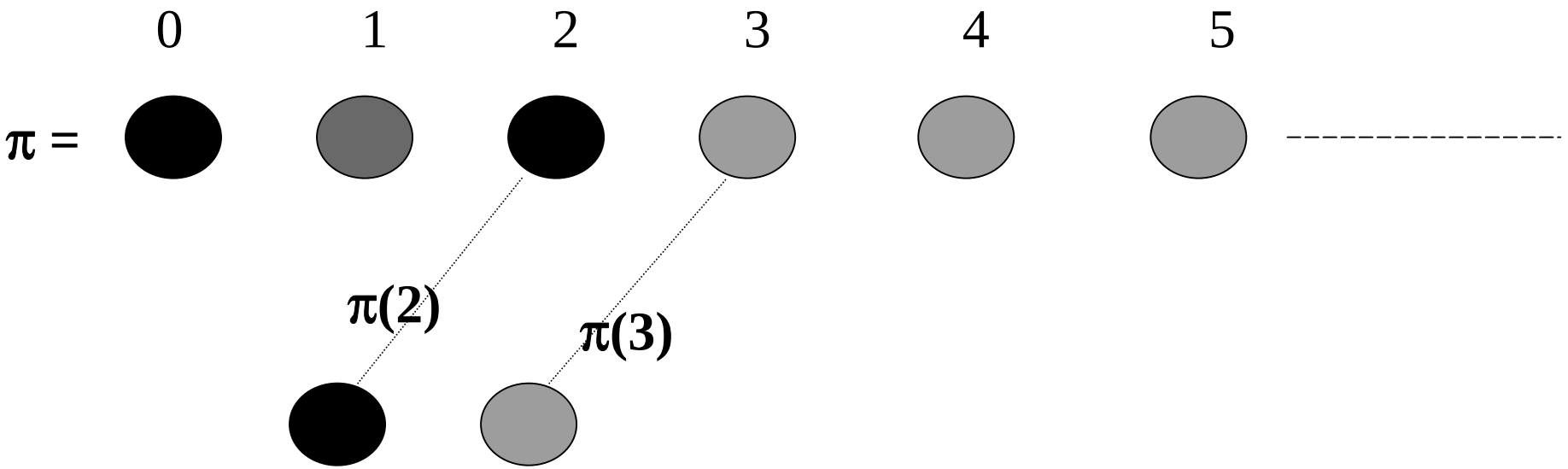
K, g ⊨ EX(uh) ?

K, r ⊨ EX(h) ?

Semantics

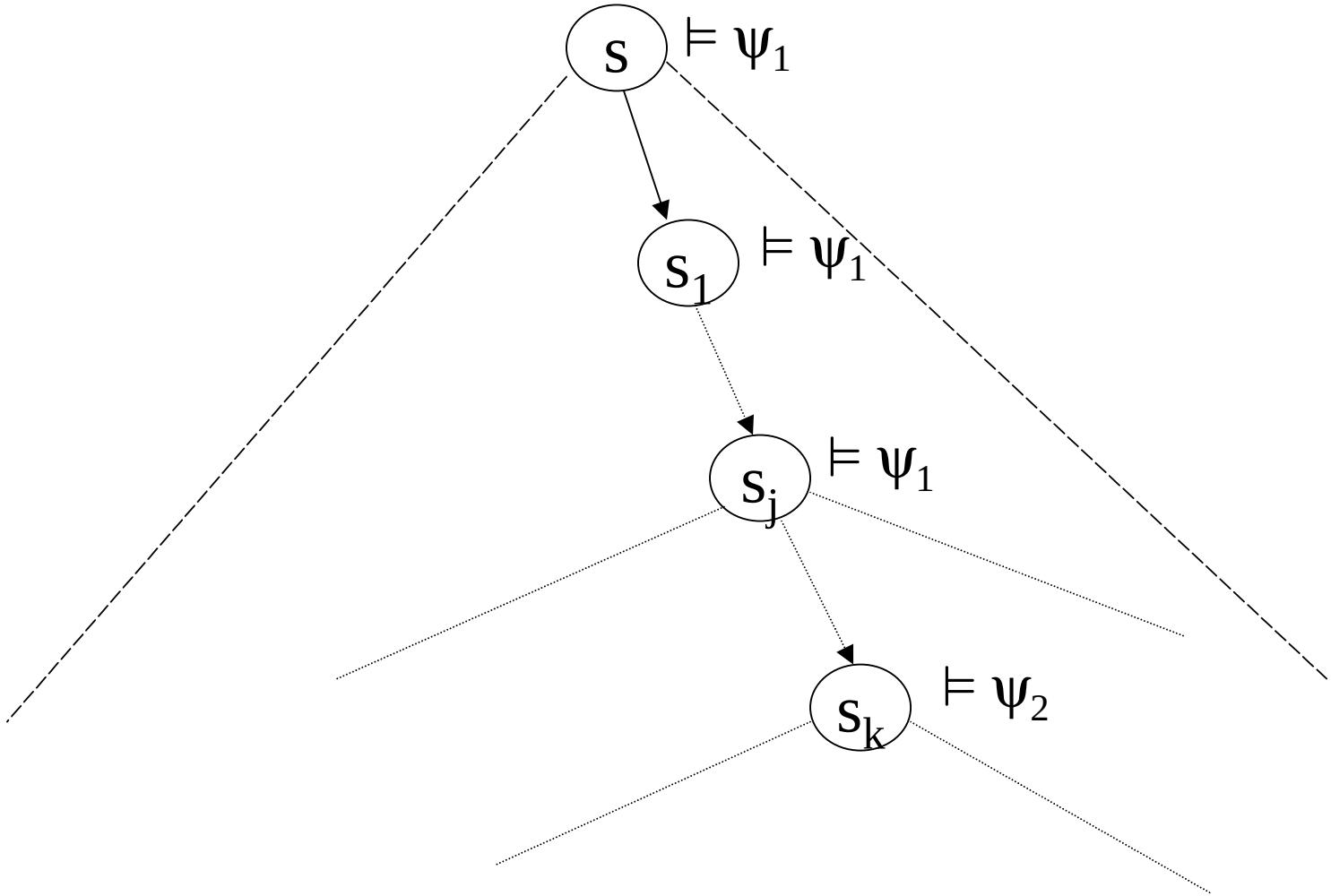
- $\mathbf{K = (S, S_0, R, AP, L)\ ;\ L: \rightarrow\ 2^{AP}\ ;\ s \in S}$
- ***A path from*** **s** is a (infinite) sequence of states $\pi = \mathbf{s_0, s_1, s_2, \ldots, s_i, s_{i+1}, \ldots}$ s.t:
  - $\mathbf{s = s_0}$
  - $\mathbf{s_i \rightarrow\ \ s_{i+1}}$ (i.e. $\mathbf{R(s_i, s_{i+1})}$)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


off
on
on
off


0
1
2
3
4
5
π =
π(2)
π(3)

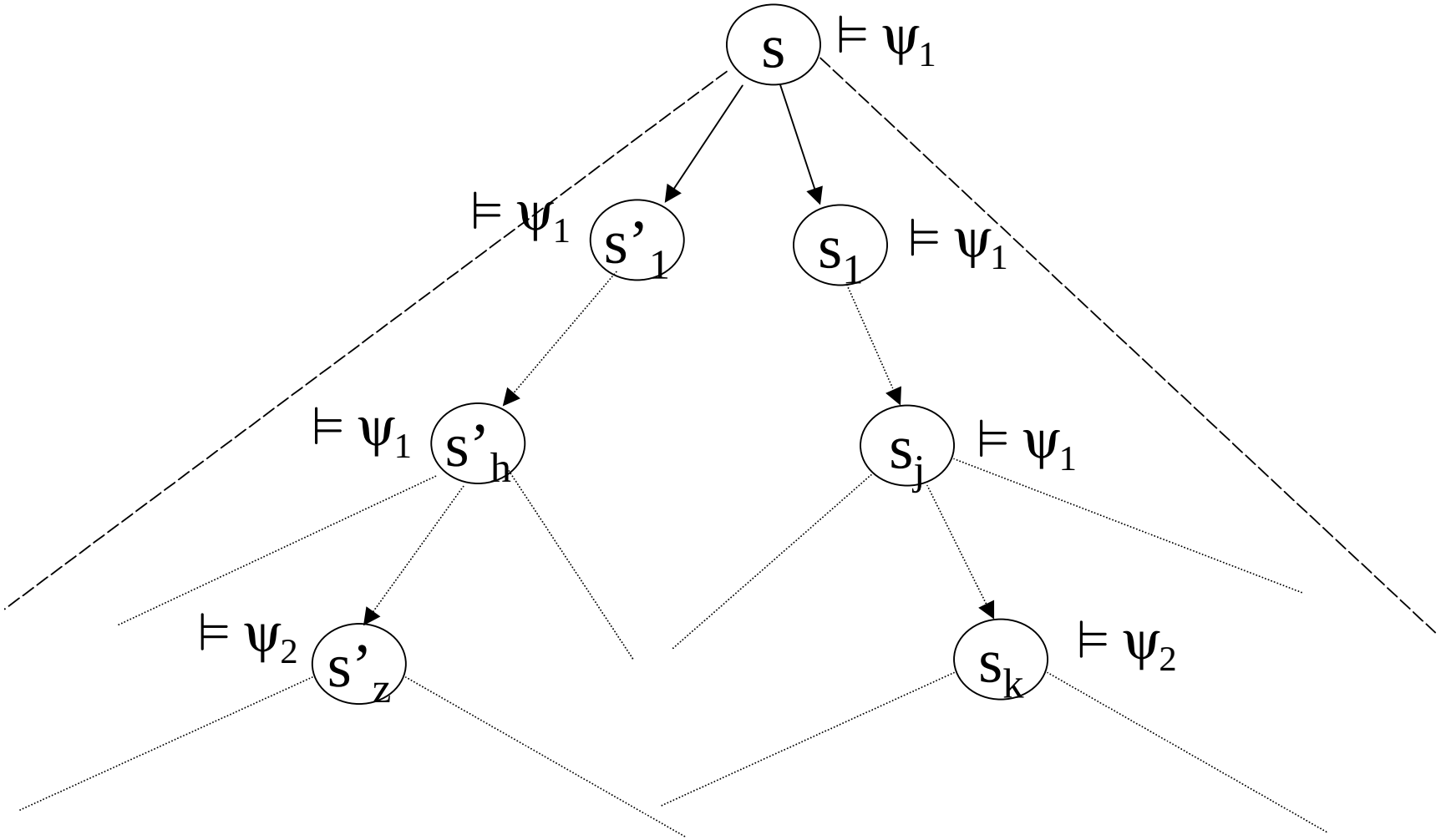
Semantics

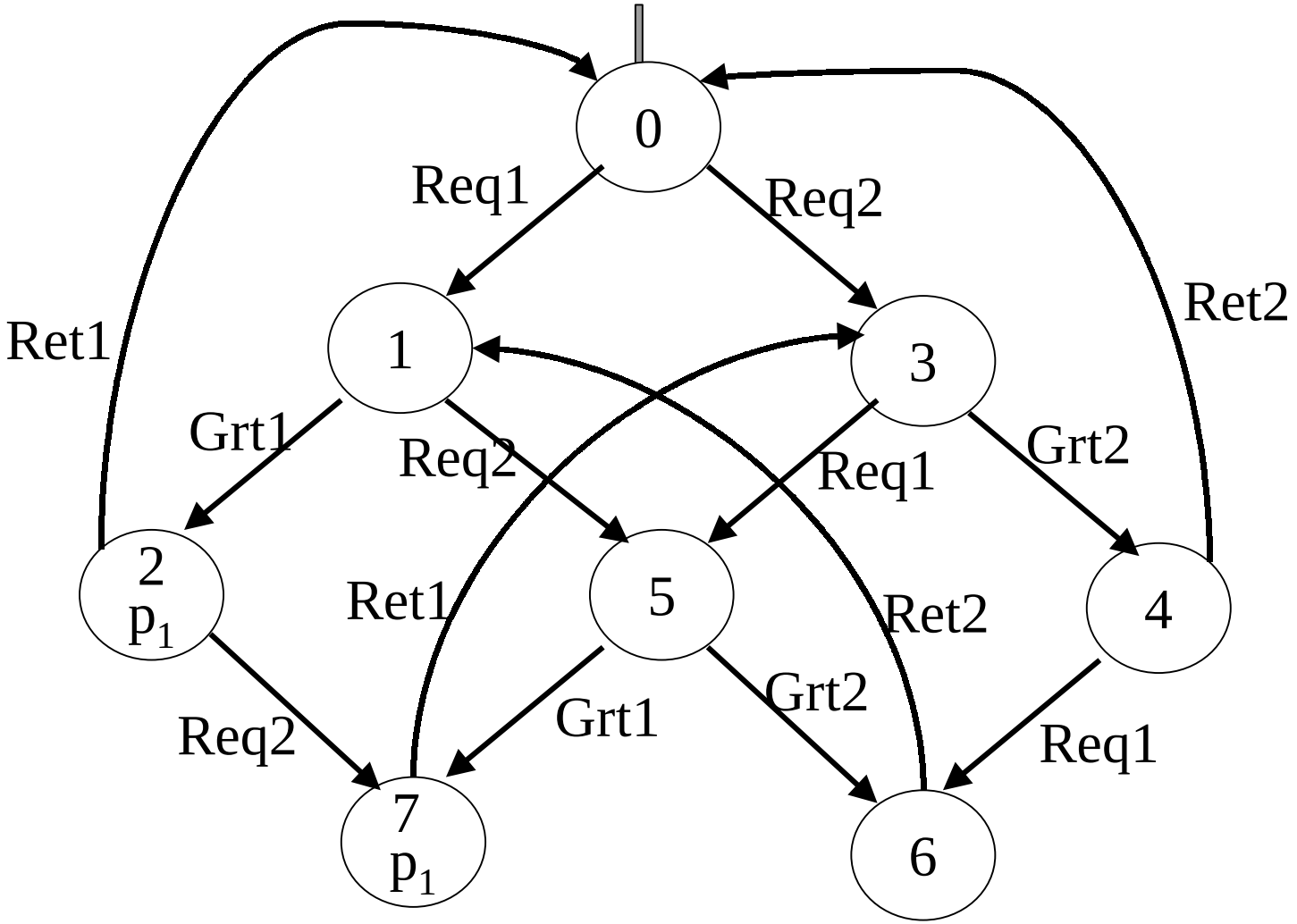
- **CTL ::= p** $\mid \neg\psi \mid \psi_1 \vee \psi_2 \mid$ **EX(**ψ**)** $\mid$
  $\mid$ **EU(**ψ_1**,** ψ_2**)** $\mid$ **AU(**ψ_1**,** ψ_2**)**
- **K = (S,** S_0**, R, AP, L)** ; **L:** $\rightarrow$ 2^{AP} ; $s \in S$
- **K, s** $\models$ **EU(**ψ_1**,** ψ_2**)** iff ***there exists a path*** $\pi = s_0, s_1, \ldots$ from **s** (i.e. s_0=**s**) and $k \geq 0$ such that:
  - **K,** π**(k)** $\models \psi_2$
  - **K,** π**(j)** $\models \psi_1$, for all $0 \leq j < k$.


s
⊨ ψ1
s1
⊨ ψ1
sj
⊨ ψ1
sk
⊨ ψ2

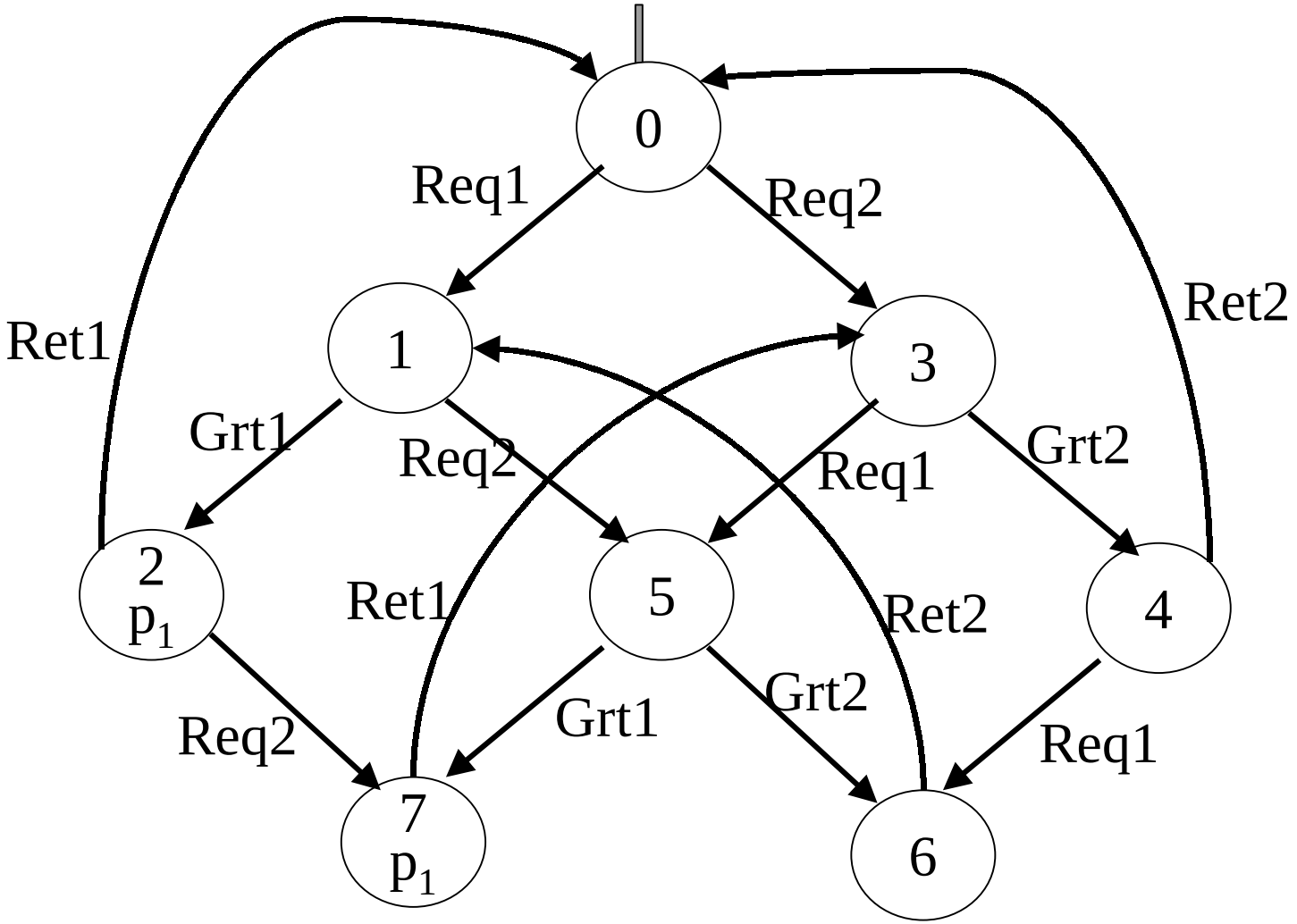
Semantics

- **CTL ::= p** $\mid \neg\psi \mid \psi_1 \vee \psi_2 \mid$ **EX(**ψ**)** $\mid$
  $\mid$ **EU(**ψ_1**,** ψ_2**)** $\mid$ **AU(**ψ_1**,** ψ_2**)**
- **K = (S,** S_0**, R, AP, L)** ; **L:** $\rightarrow 2^{AP}$; $s \in S$
- **K,** $s \models$ **AU(**ψ_1**,** ψ_2**)** iff ***for every path*** $\pi = s_0, s_1, \ldots$ from **s** there exists $k \geq 0$ such that:
  - **K,** $\pi(k) \models \psi_2$
  - **K,** $\pi(j) \models \psi_1$, for all $0 \leq j < k$.

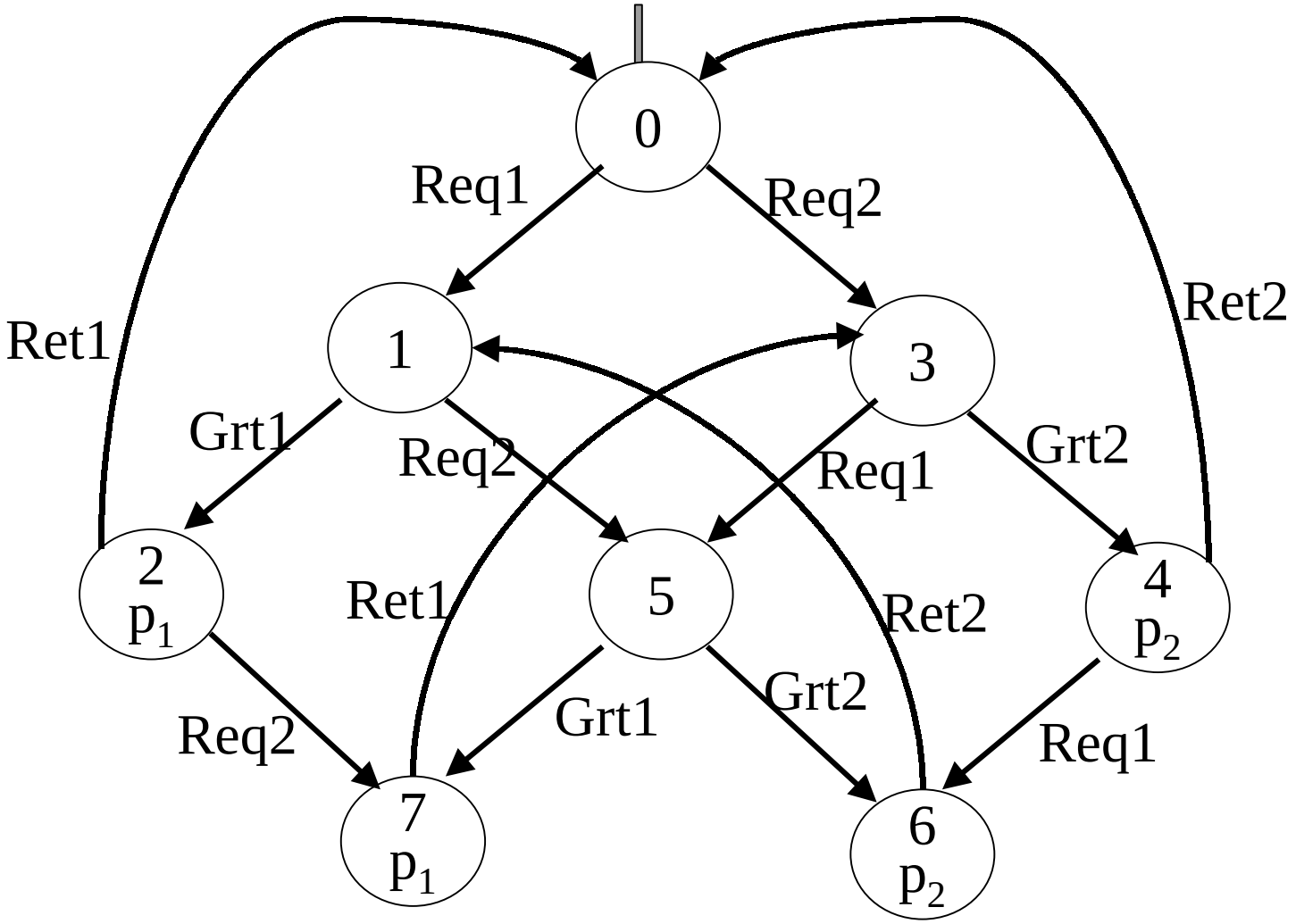

s
⊨ ψ1
⊨ ψ1
s'1
s1
⊨ ψ1
⊨ ψ1
s'h
sj
⊨ ψ1
⊨ ψ2
s'z
sk
⊨ ψ2


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

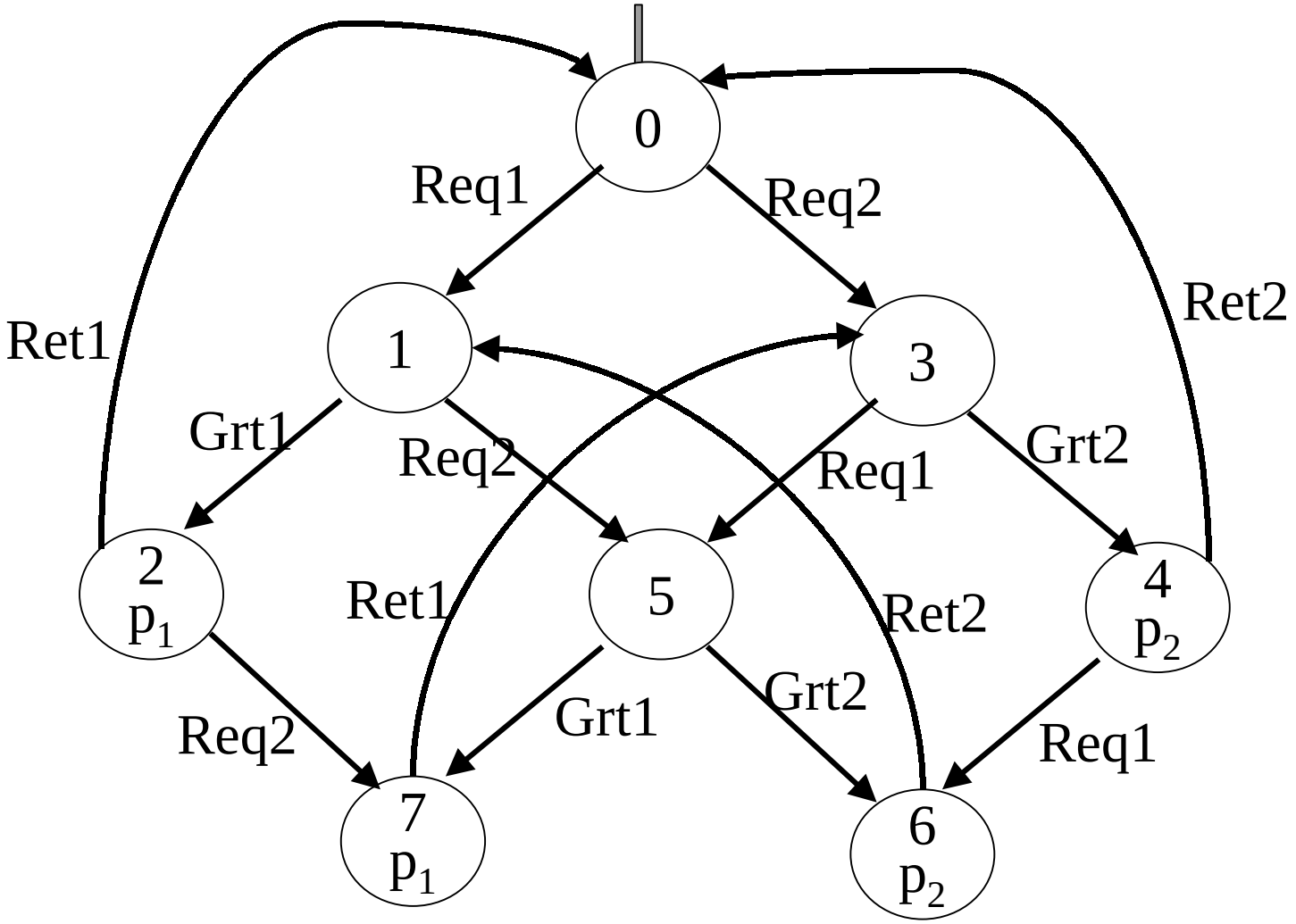
M, 0 ⊨ EU(⊤, p_1) ?


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

M, 0 ⊨ AU(⊤, p_1) ?

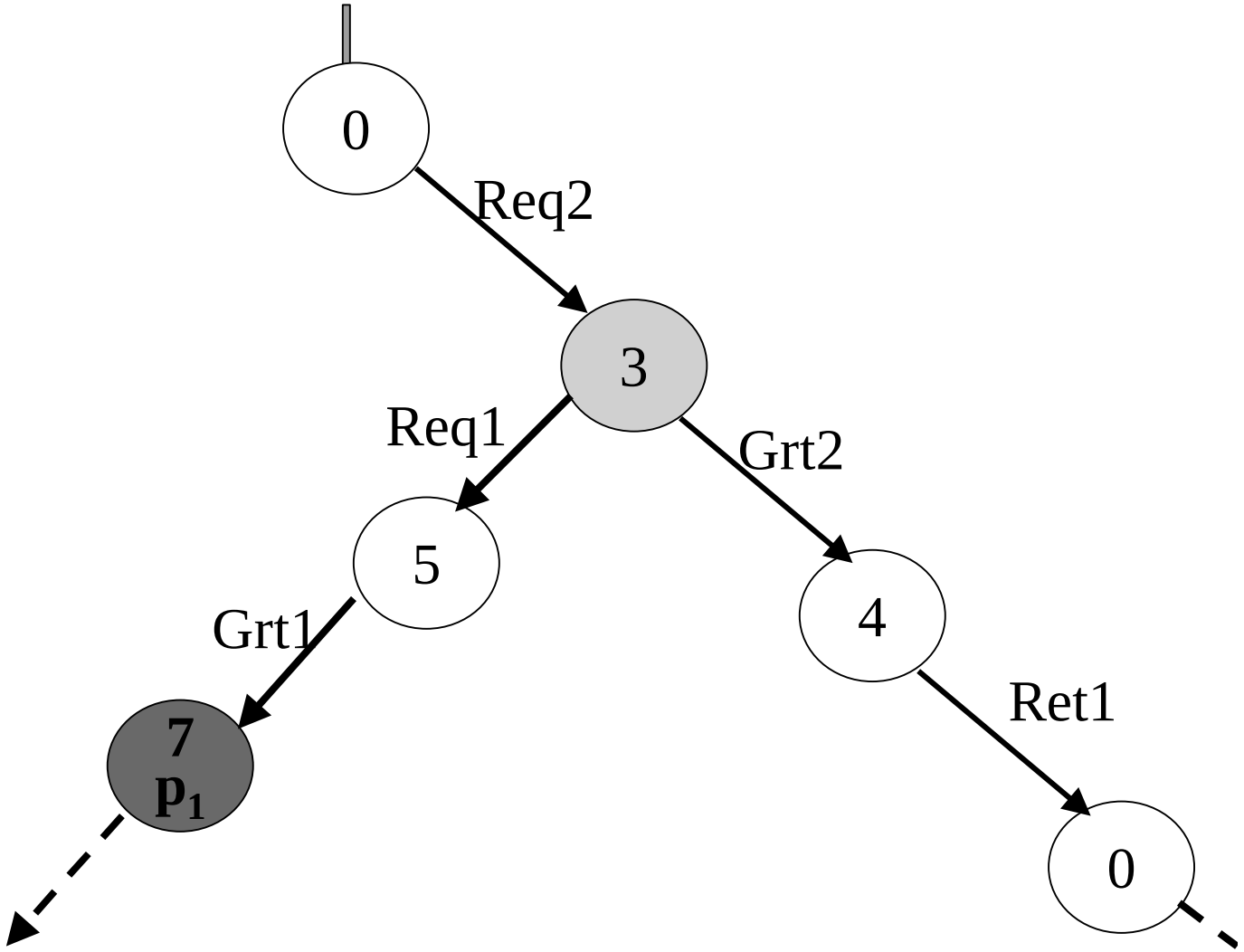


M, 0 ⊨ AU(⊤, $p_1 \vee p_2$) ?



M, 0 ⊨ AU(⊤, EU(⊤, p_1)) ?

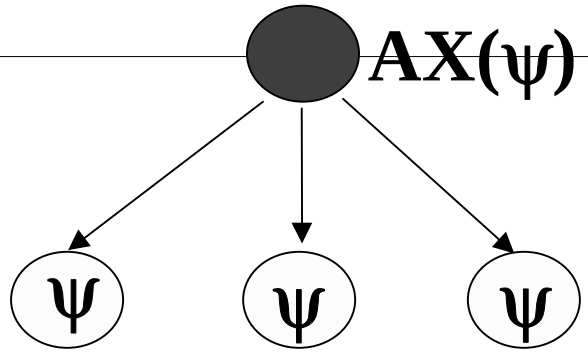
From s_0, ***all*** the computations will reach a point, where it is ***possible*** for 1 to print ***eventually.***



M, 0 ⊨ AU(⊤, EU(⊤, p_1)) ?

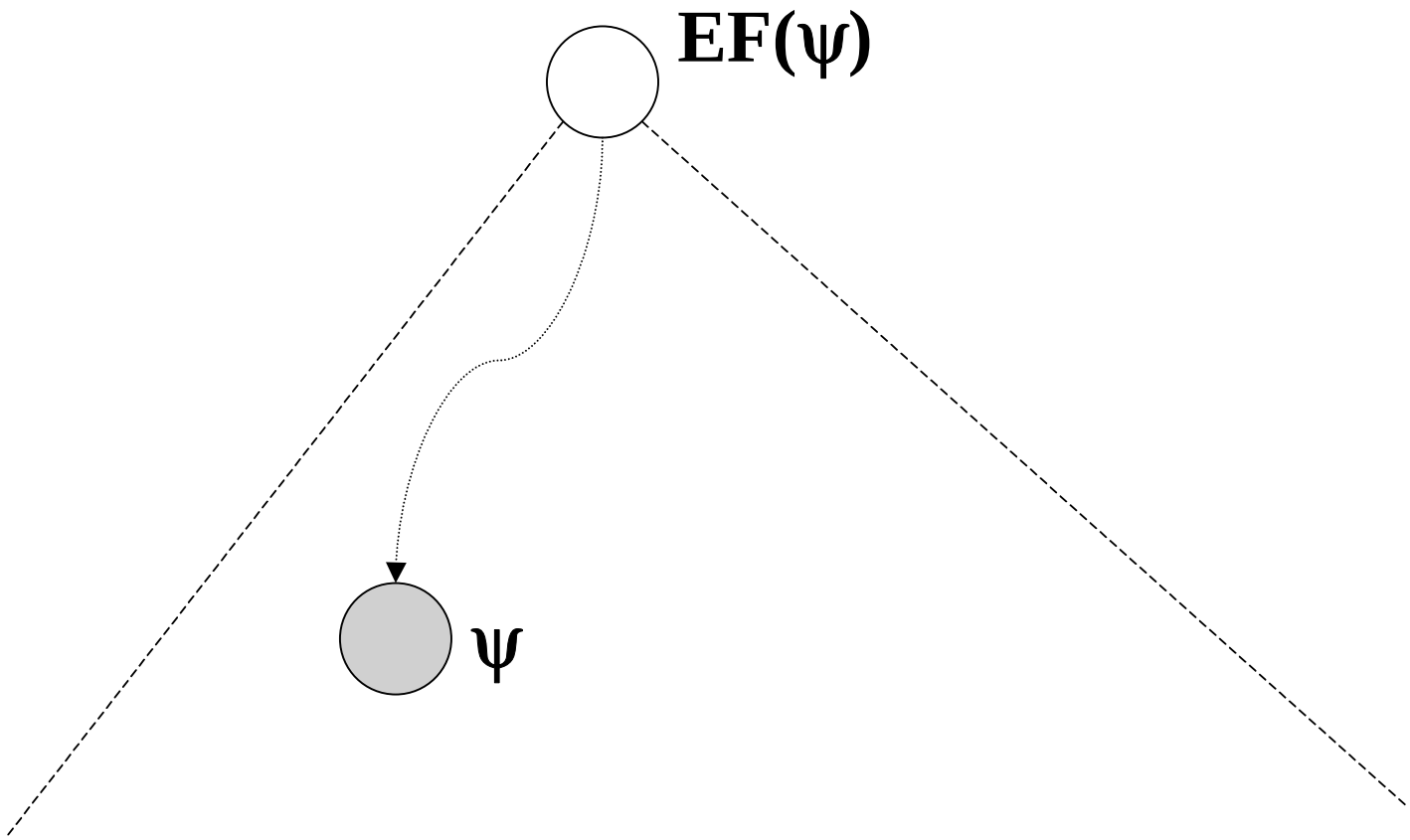
Derived Operators

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



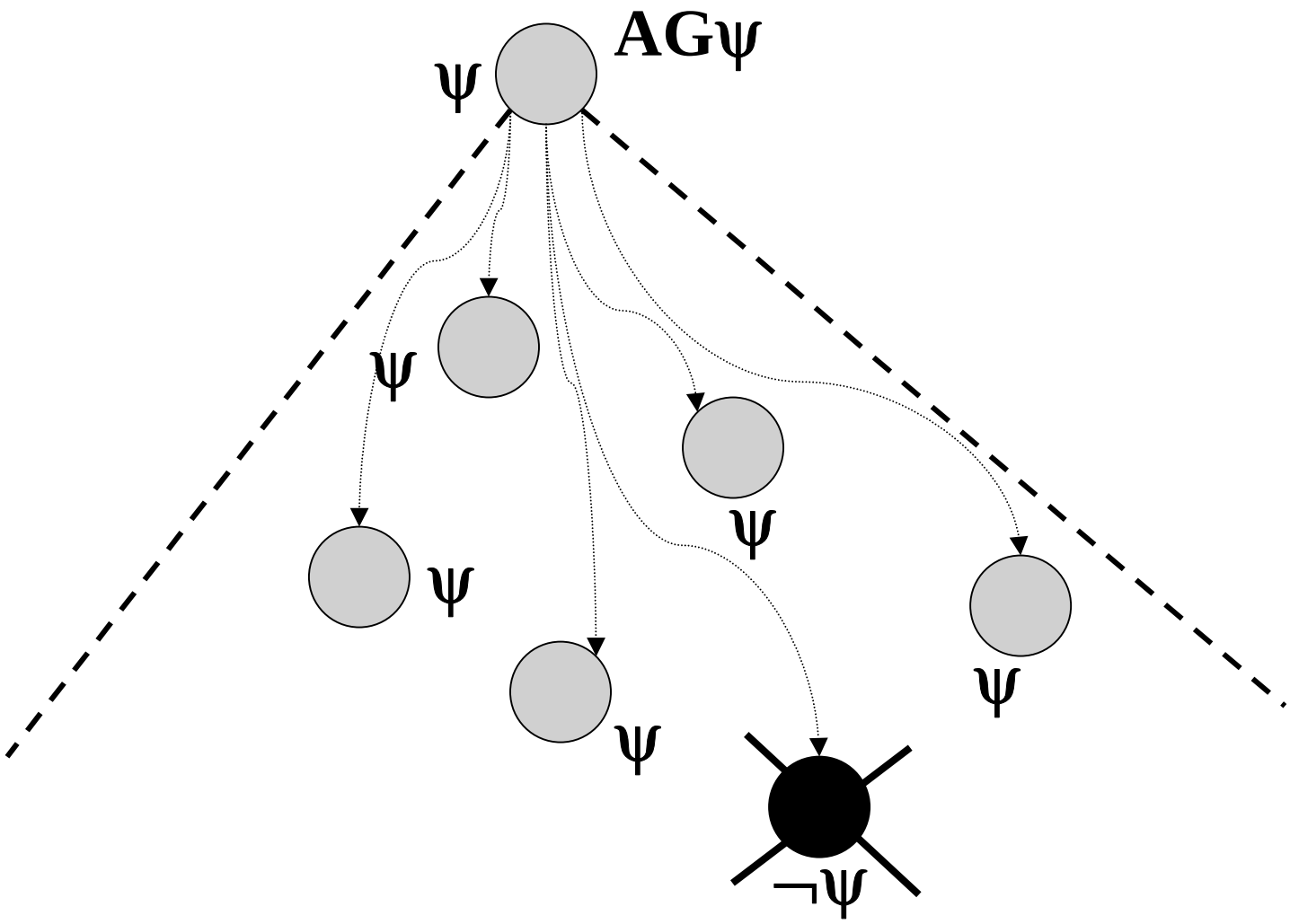
Derived Operators

- **K, s $\models$ EF(ψ)**
- **EF(ψ) = EU($\top$, ψ)**
  - There exists a path π (from **s**) and **k** $\geq$ **0** such that:
    - **K, π(k) $\models$ ψ.**


EF(ψ)
ψ

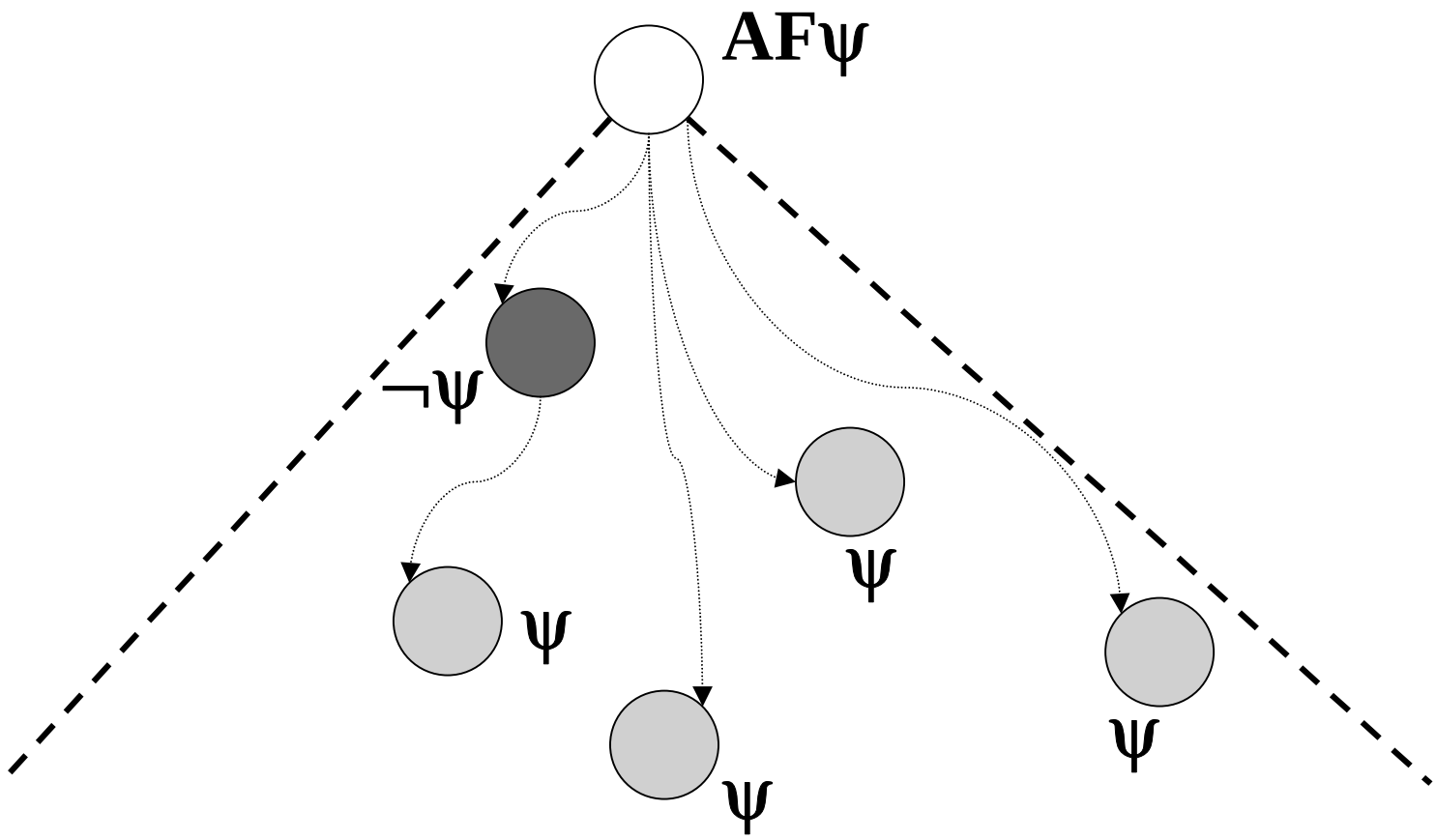
Derived Operators

- **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

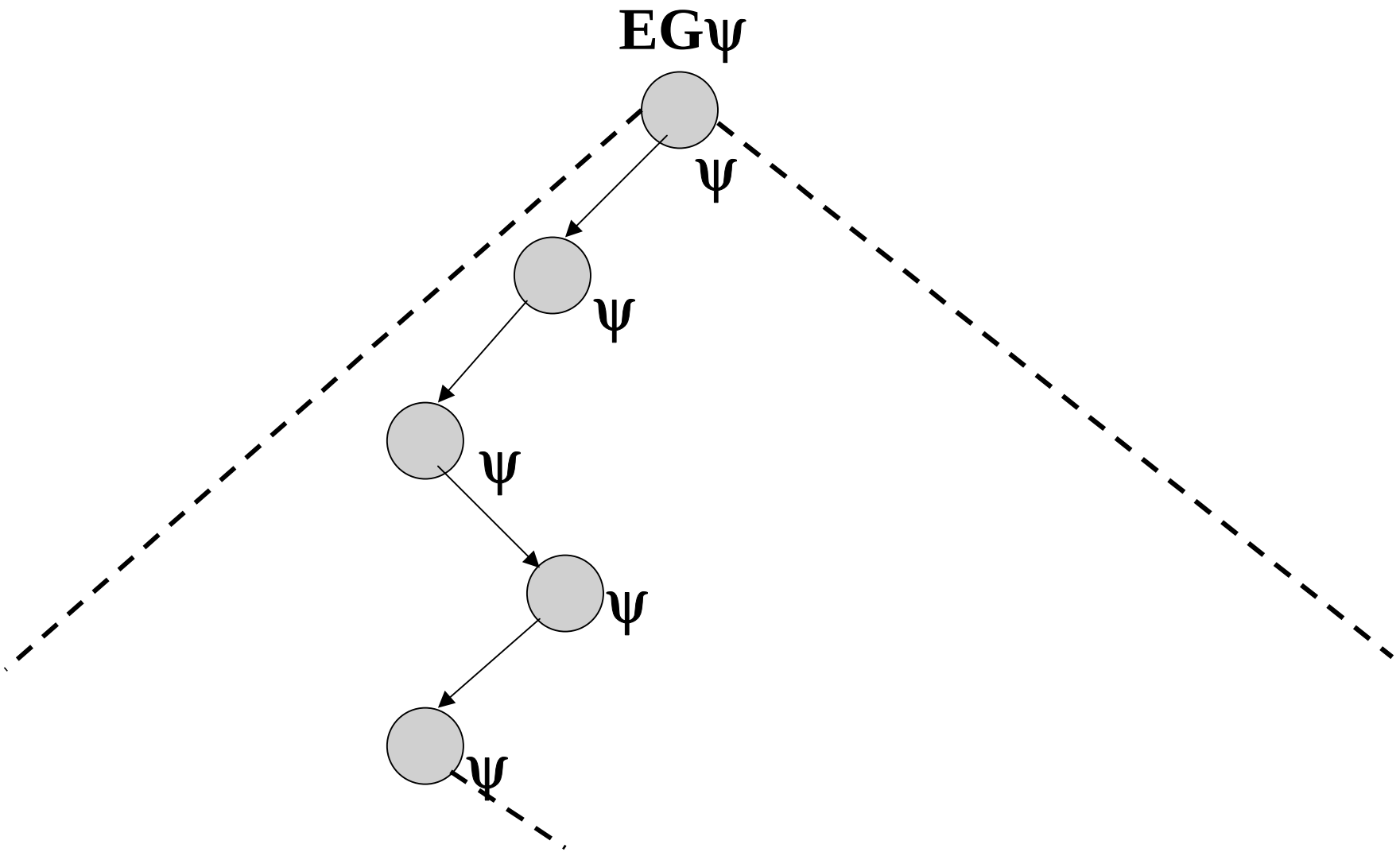
Derived Operators

- **K, s $\models$ AF(ψ)**
- **AF(ψ) = AU($\top$, ψ)**
  - ***For every path*** **π** from **s**, there ***exists*** **k $\geq$ 0** such that:
    - **K, π(k) $\models$ ψ.**


AFψ
¬ψ
ψ
ψ
ψ
ψ

Derived Oparators

- **K, s $\models$ EG(ψ)**
- **EG(ψ) = $\neg$AF($\neg\psi$)**
  - It is **not** the case that ***for every path*** π from **s** there is a **k** $\geq$ **0** such that **K,** π**(k)** $\models$ ψ.
  - ***There exists a path*** π from **s** such that, for every **k** $\geq$ **0**:
    - **K,** π**(k)** $\models$ ψ.


EGψ
ψ
ψ
ψ
ψ
ψ

A more convenient CTL

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

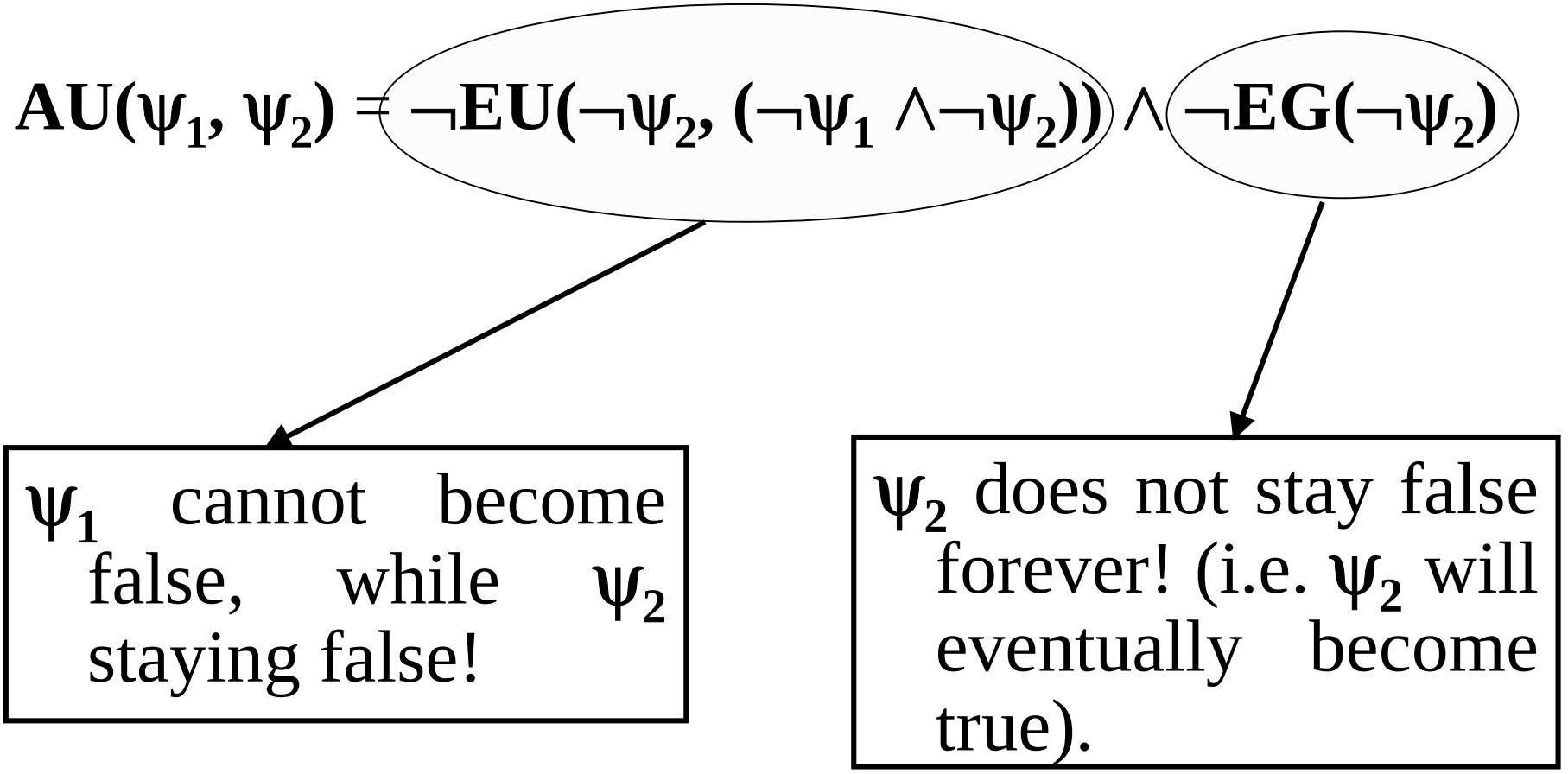
A more convenient CTL

- **NCTL ::= p** | $\neg\psi$ | $\psi_1 \vee \psi_2$ | **EX(ψ)** |
  | **EU(ψ_1, ψ_2)** | **EG(ψ)**
- **K, s** $\models$ **EG(ψ)** iff ***there exists a path*** π from **s** such that for every **k** $\geq$ **0**:
  - **K,** π**(k)** $\models$ 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)
- **AU(ψ_1, ψ_2) = $\neg$EU($\neg\psi_2$, ($\neg\psi_1 \wedge \neg\psi_2$)) $\wedge$ $\neg$EG($\neg\psi_2$)**

A more convenient CTL



A more convenient CTL

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

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

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

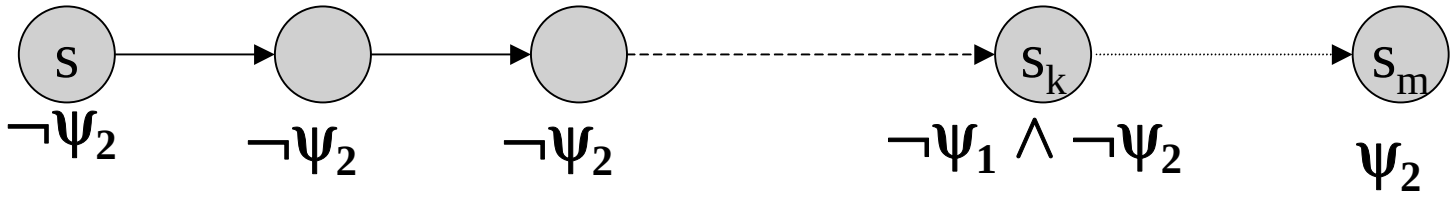
A more convenient CTL

- $\mathbf{AU}(\psi_1, \psi_2) = \mathbf{MagicAU}(\psi_1, \psi_2) =$

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

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

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



- Suppose **K, π(m) ⊨ ψ_2,** required by **K, s ⊨ AU(ψ_1,ψ_2)**
- Take **m** to be the least such number.
- Then **m > k**, since **K, s ⊨ EU($\neg\psi_2$, $\neg\psi_1 \wedge \neg\psi_2$)**
- But **0 ≤ k < m** and **K, π(k) ⊨ $\neg\psi_1$**
- Hence **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 ⊨ MagicAU(ψ_1, ψ_2)**

From CTL to NCTL

- In a similar way we can argue that:

> ***if*** **K, s $\models$ MagicAU(ψ_1, ψ_2)**
>
> ***then*** **K, s $\models$ 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) = MagicAU(ψ_1, ψ_2) =**

  $\neg$(EU($\neg\psi_2$, ($\neg\psi_1 \wedge \neg\psi_2$)) $\wedge$ AF(ψ_2)

- **Magic$_1$ = $\neg$ EU($\neg\psi_2$, ($\neg\psi_1 \wedge \neg\psi_2$)**
- **Magic$_2$ = 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:
  - $\mathbf{K, s \models AU(\psi_1, \psi_2)}$ iff

    $\mathbf{K, s \models Magic_1 \wedge Magic_2}$
- We argued last time:
  - If $\mathbf{K, s \models AU(\psi_1, \psi_2)}$ then

    $\mathbf{K, s \models Magic_1 \wedge Magic_2}$

From CTL to NCTL

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

$\Leftarrow$ We need to argue that:

– If $\mathbf{K, s} \models \mathbf{Magic}_1 \wedge \mathbf{Magic}_2$ then

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

- So assume $\mathbf{K, s} \models \mathbf{Magic}_1 \wedge \mathbf{Magic}_2$.

- $\mathbf{Magic}_1 = \neg\mathbf{EU}(\neg\psi_2, (\neg\psi_1 \wedge \neg\psi_2))$.
- $\mathbf{Magic}_2 = \neg\mathbf{EG}\neg\psi_2 = \mathbf{AF}\psi_2$

From CTL to NCTL

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

From CTL to NCTL

Now consider **m** such that $\mathbf{0 \leq m < k}$.

CLAIM: $\mathbf{K, \sigma(m) \models \psi_1}$

- If the **CLAIM** is true then we are done.
- Suppose $\mathbf{K, \sigma(m) \models \neg\psi_1}$.
  - Then $\mathbf{K, \sigma(m) \models \neg\psi_1 \wedge \neg\psi_2}$ $(\mathbf{m < k})$ ***WHY*?**
  - Further $\mathbf{K, \sigma(j) \models \neg\psi_2}$ if $\mathbf{0 \leq j < m}$ since $\mathbf{j < m < k}$.
  - Hence $\mathbf{K, \sigma(0) \models EU(\neg\psi_2, \neg\psi_1 \wedge \neg\psi_2)}$
  - Hence $\mathbf{K, s \not\models Magic_1}$ and ***Contradiction***!

CTL Model Checking

- The ***CTL model checking problem***.
  - **K = (S, S_0, R, AP, L)** (system model)
  - ψ a ***CTL*** formula (specification of the property)
- **K** $\models \psi$ ***iff***
  - **K, s_0** $\models \psi$ for every $s_0 \in S_0$.
- 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:
    - **K, s** $\models \psi$

The Sub-formulas of ψ

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

The Labeling Procedure.

- **K = (S, S_0, R, AP, L)**
  - **s** $\in$ **S**
  - ψ a ***NCTL*** formula (built out of **AP**).

- **Strategy**:
  - Construct **Labels**: **S** $\rightarrow$ $\mathbf{2^{SF(\psi)}}$
  - $\mathbf{2^{SF(\psi)}}$, 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** $\vDash \psi$ ***iff*** $\psi \in$ **Labels(s)**.

The Labels function

- For every **s** $\in$ **S**:
- **Stage 1**:
  - **Labels(s) = L(s)** (**K =(S, S_0, R, AP, L)**)
- ….
- Assume we have done up to stage **i**.
- **Stage i +1**:
  - If $\alpha = \neg\beta$ then

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

The Labels function

- For every **s** $\in$ **S**:
- **Stage i +1**:
  - If $\alpha = \beta_1 \vee \beta_2$ then

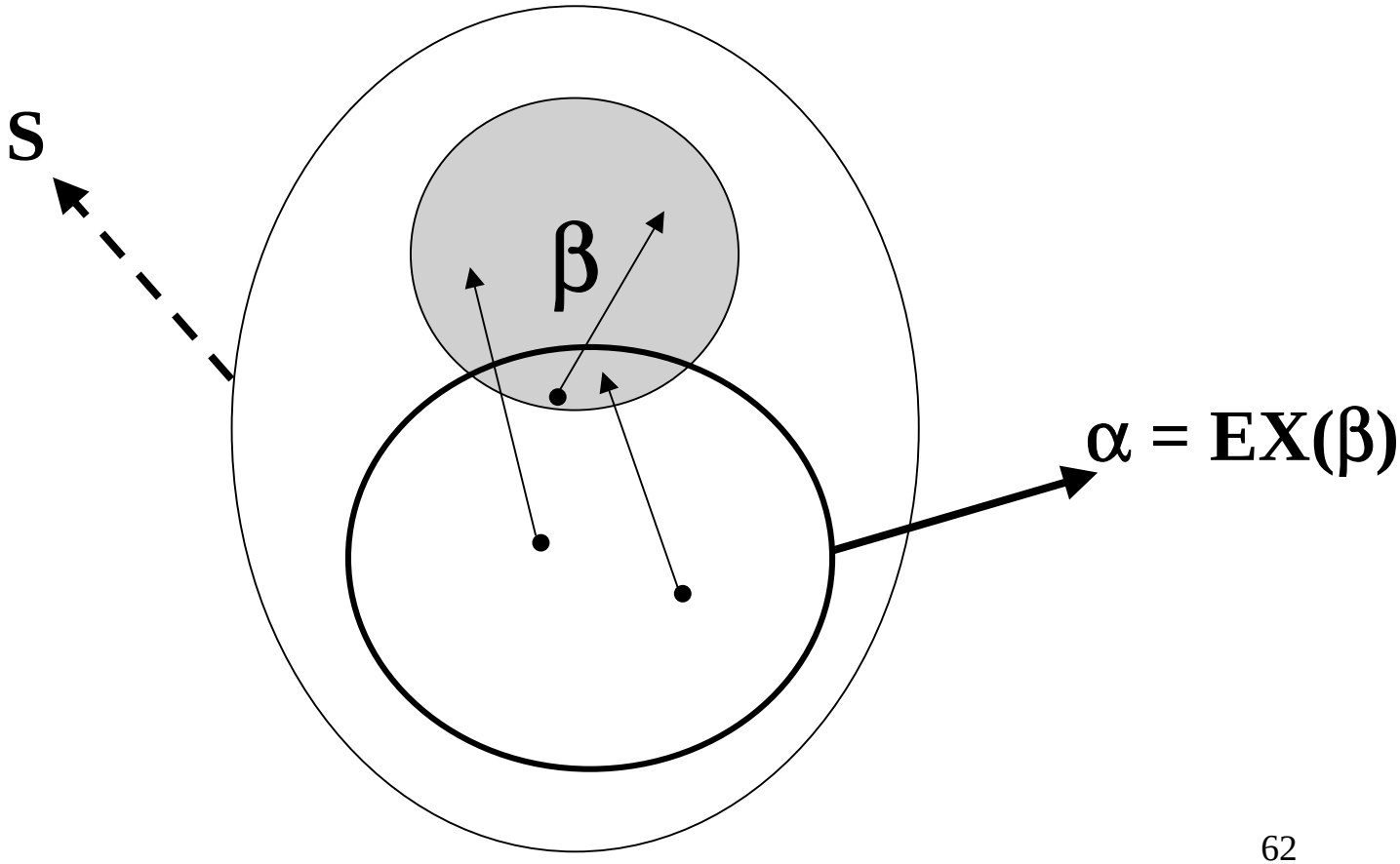
    $\alpha \in$ **Labels(s)** ***iff*** $\beta_1 \in$ **Labels(s) or** $\beta_2 \in$ **Labels(s)**

  - If $\alpha =$ **EX**β then

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

    $\beta \in$ **Labels(t)** and **R(s, t)**

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**};

forall **t** ∈ **S** such that (**t**,**s**) ∈ **R** do

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

The Labels Function

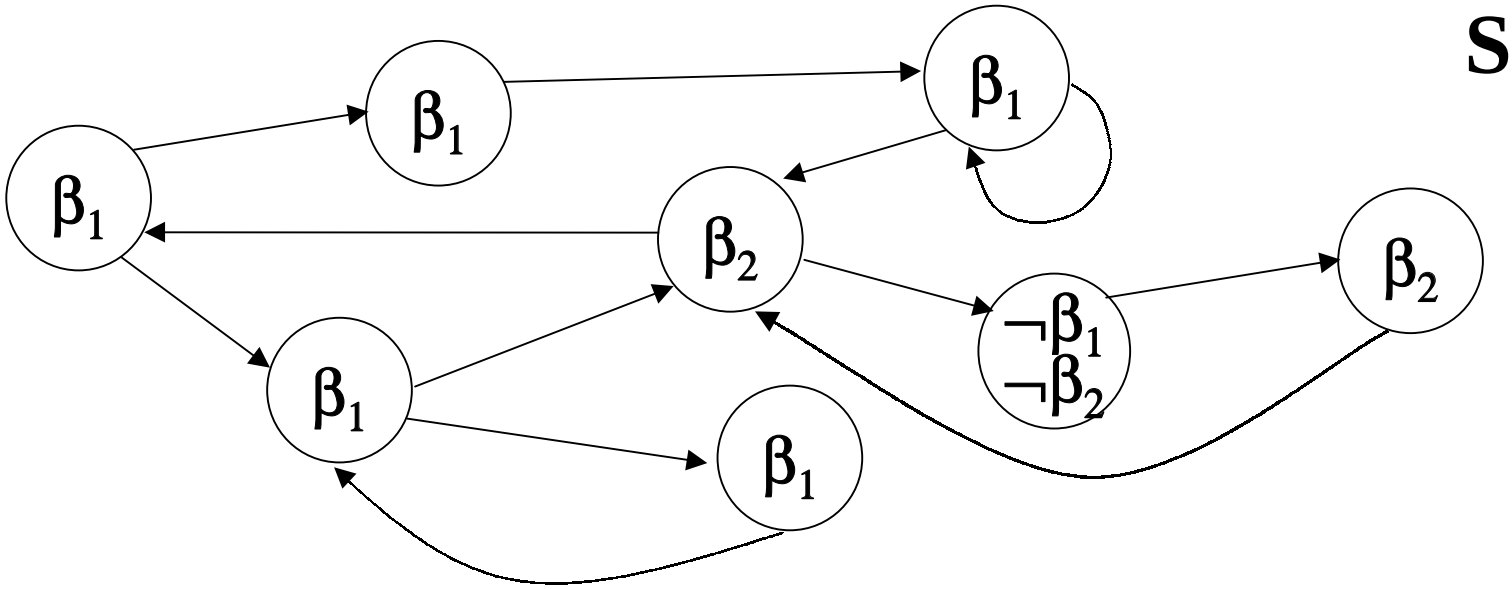
- For every **s** $\in$ **S**:
- **Stage i +1**:
  - If α = **EU**(β_1,β_2) then

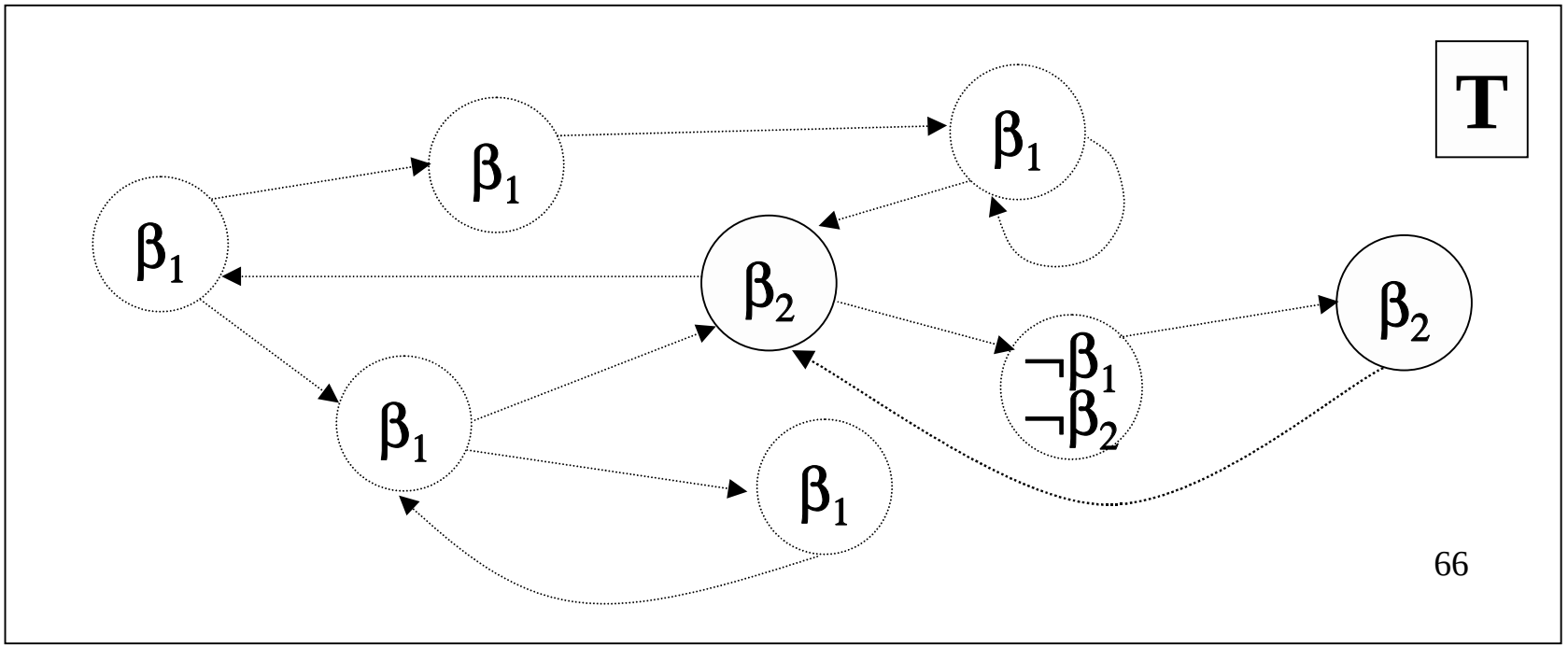
    $\alpha \in$ **Labels(s)** ***iff***
    - $\beta_2 \in$ **Labels(s) or**
    - $\beta_1 \in$ **Labels(s) and EU**(β_1,β_2) $\in$ **Labels(t) for some t with R(s,t).**

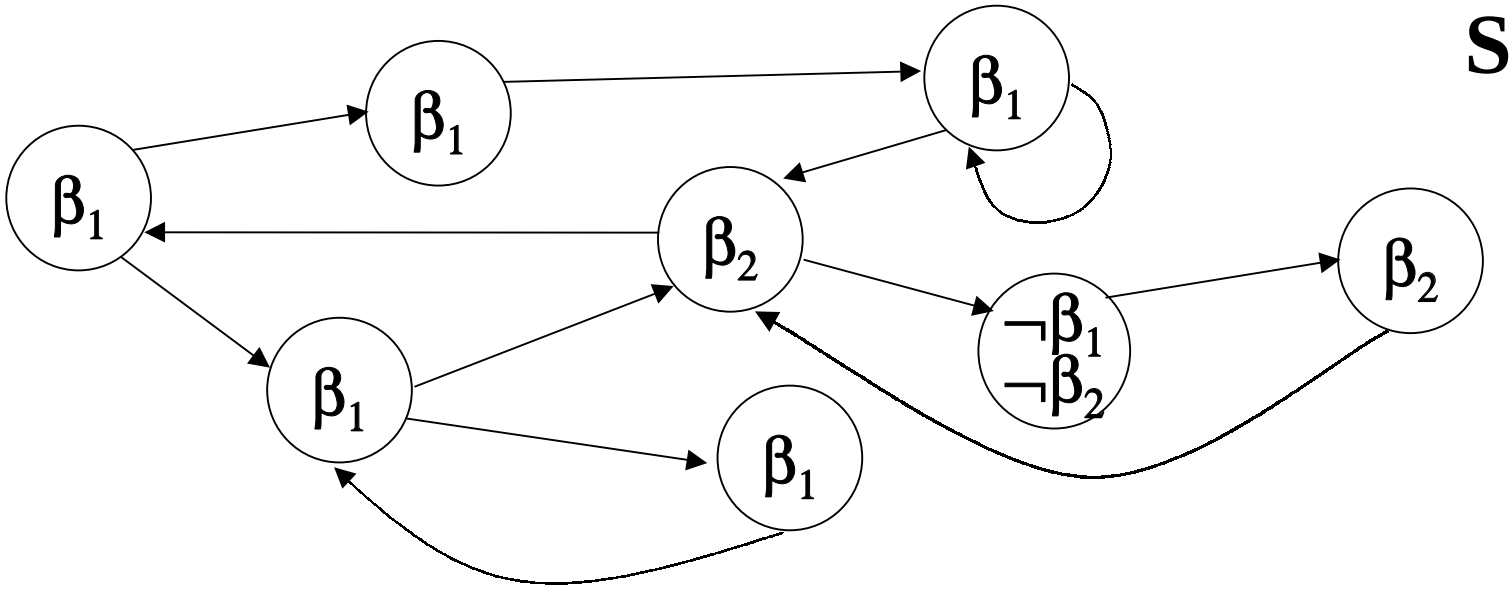
The Labels Function

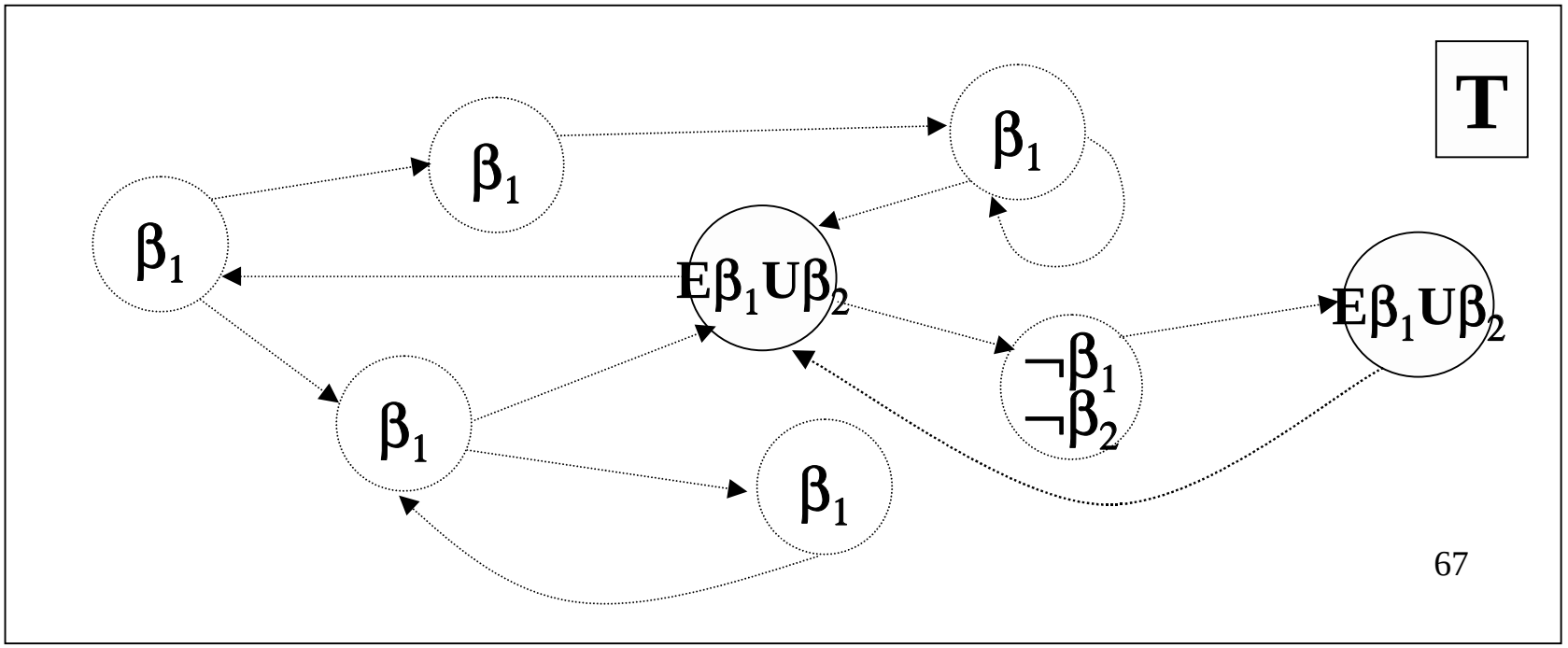
- Collect in **T** all the states satisfying β_2
  - **all these states do also satisfy EU(β_1,β_2)**.
- Traversing backward **R** from states in **T** and label with **EU(β_1,β_2)** those states satisfying β_1 and reaching at least a state **s** labeled with **EU(β_1,β_2)**.

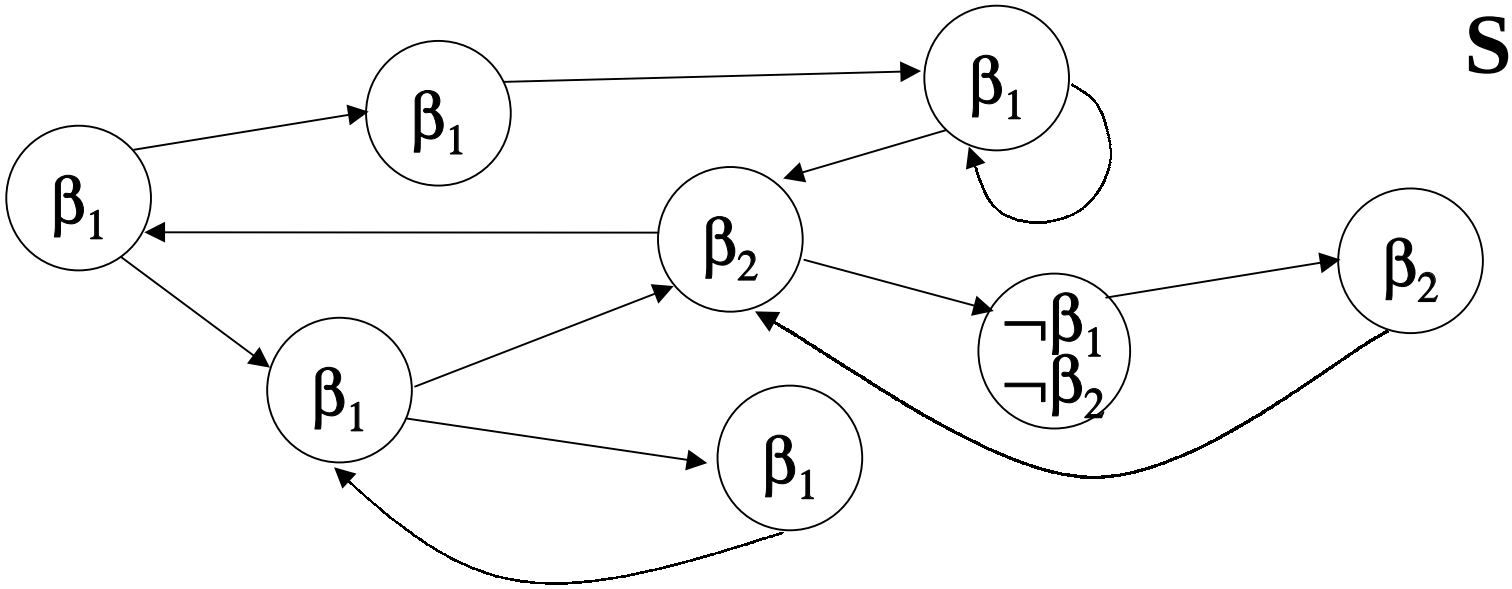
If s $\in$ T and t with R(t,s) and $\beta_1 \in$ Labels(t)
then EU(β_1,β_2) $\in$ Labels(t)

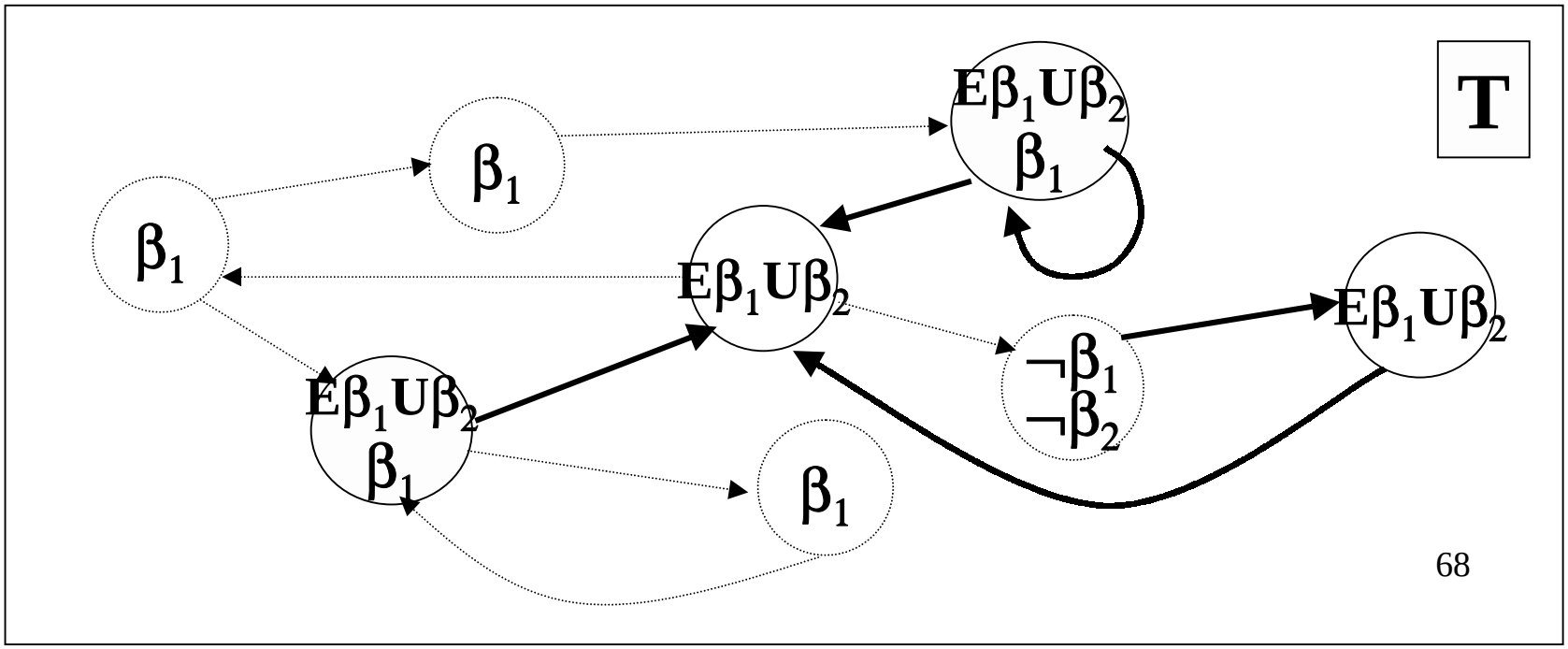

S
β1
β1
β1
β2
β2
¬β1
¬β2
β1
β1

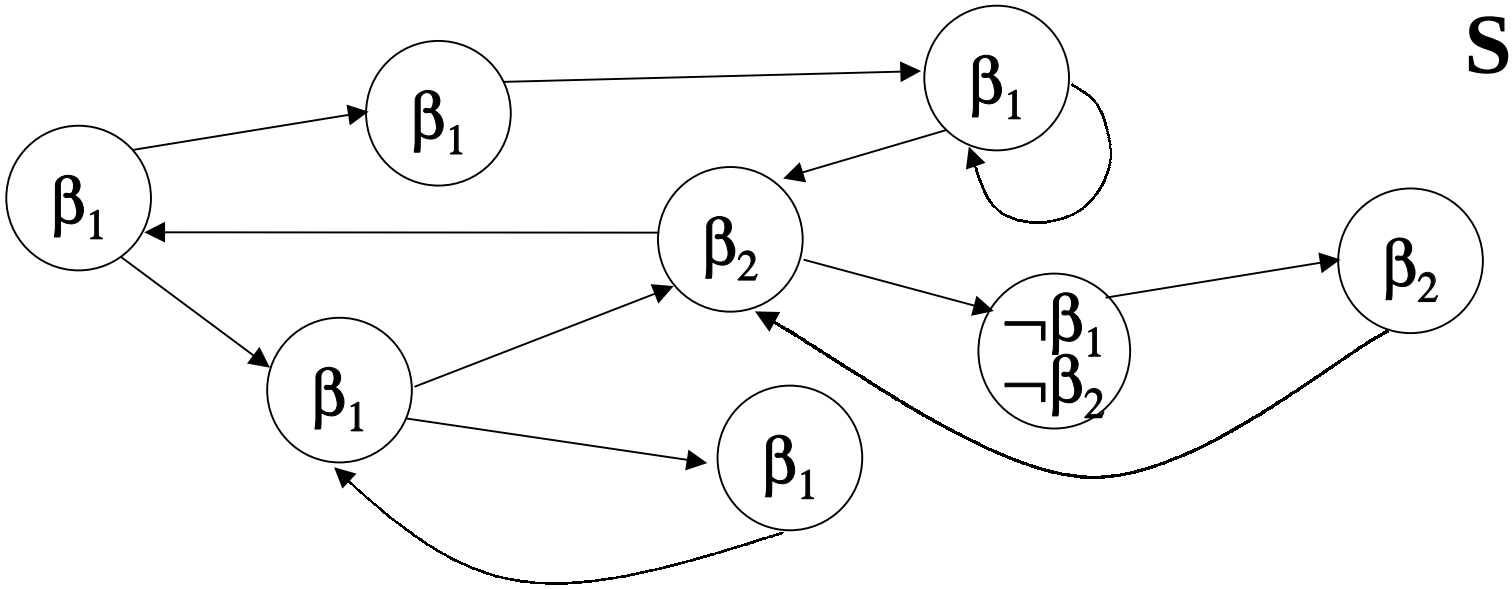

T
β1
β1
β1
β2
β2
¬β1
¬β2
β1
β1

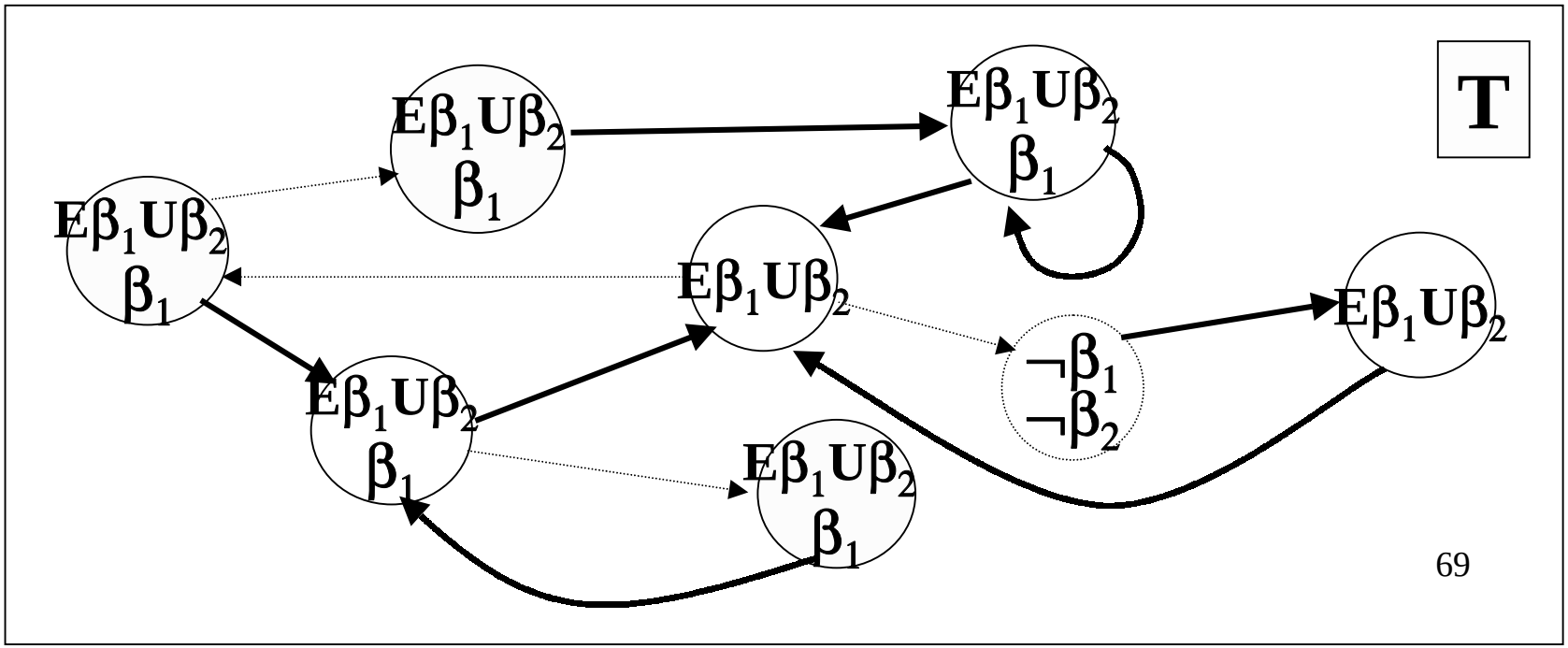

S
β1
β1
β1
β2
β2
¬β1
¬β2
β1
β1

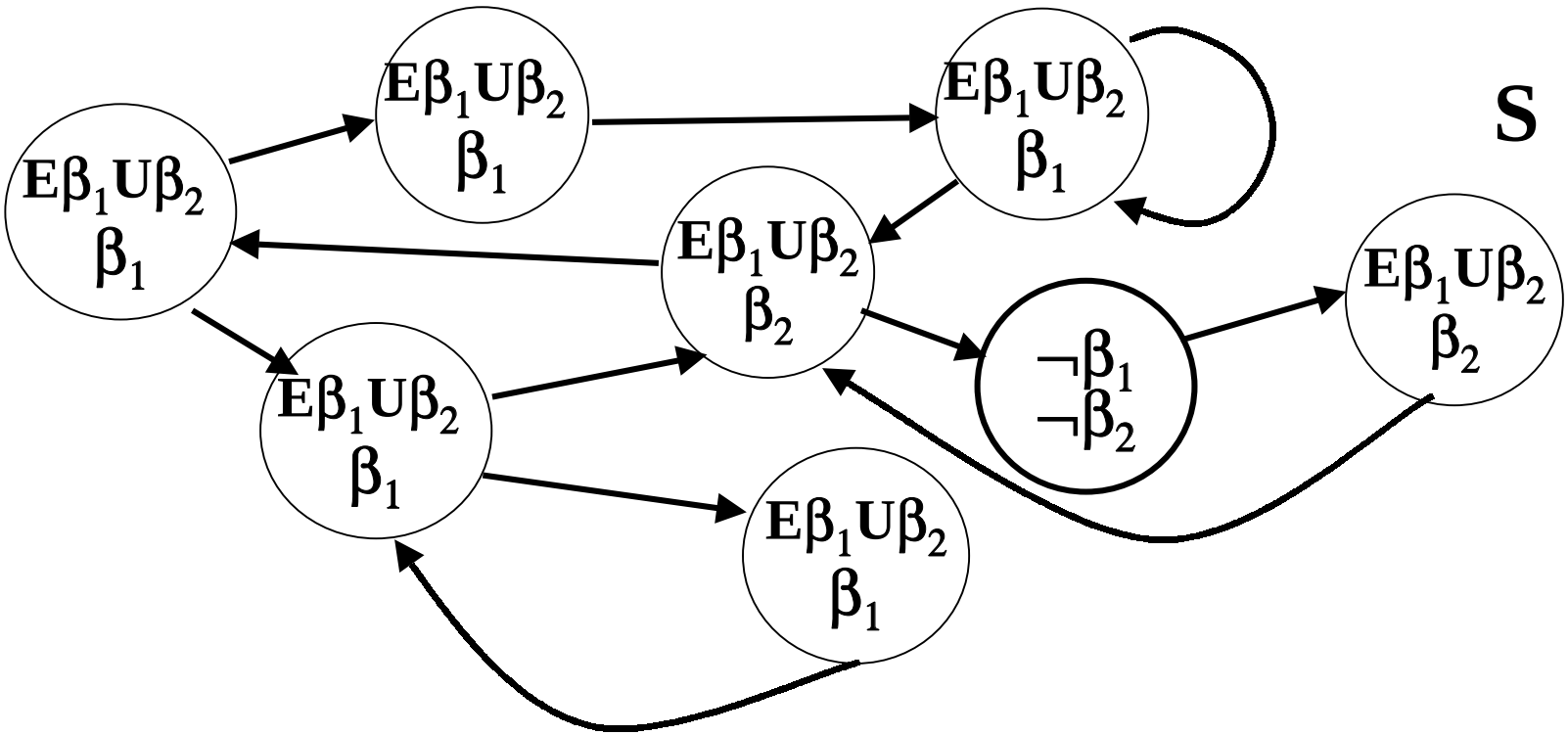

T
β1
β1
β1
Eβ1Uβ2
Eβ1Uβ2
¬β1
¬β2
β1
β1


S
β1
β1
β1
β2
β2
¬β1
¬β2
β1
β1


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


S
β1
β1
β1
β2
β2
β1
β1
¬β1
¬β2


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


Eβ1Uβ2
β1
Eβ1Uβ2
β1
Eβ1Uβ2
β1
Eβ1Uβ2
β2
Eβ1Uβ2
β2
¬β1
¬β2
Eβ1Uβ2
β1
Eβ1Uβ2
β1
S

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

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

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

forall **s** $\in$ **T** do

Labels(**s**) := Labels(**s**) $\cup$ {**EU**(β_1,β_2)};

while **T** $\neq \emptyset$ do

chose **s** $\in$ **T**;

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

forall **t** $\in$ **T** with **(t,s)** $\in$ **R** do

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

Labels(**t**) := Labels(**t**) $\cup$ {**EU**(β_1,β_2)};

T := **T** $\cup$ {**t**};

Complexity: O(|M|)

The Labels Function

- For every **t** $\in$ **S**:
- **Stage i +1**:
  - If α = **EG(**β **)** then

    $\alpha \in$ **Labels(t)** ***iff***

    - $\beta \in$ **Labels(t) and EG(**β**)** $\in$ **Labels(s) for some s with R(t,s).**

Property of EG(β)

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

- **S' = { s | M ,s ⊨ β }**
- **R' = $R|_{S' \times S'}$** and **L' = $L|_{S'}$**

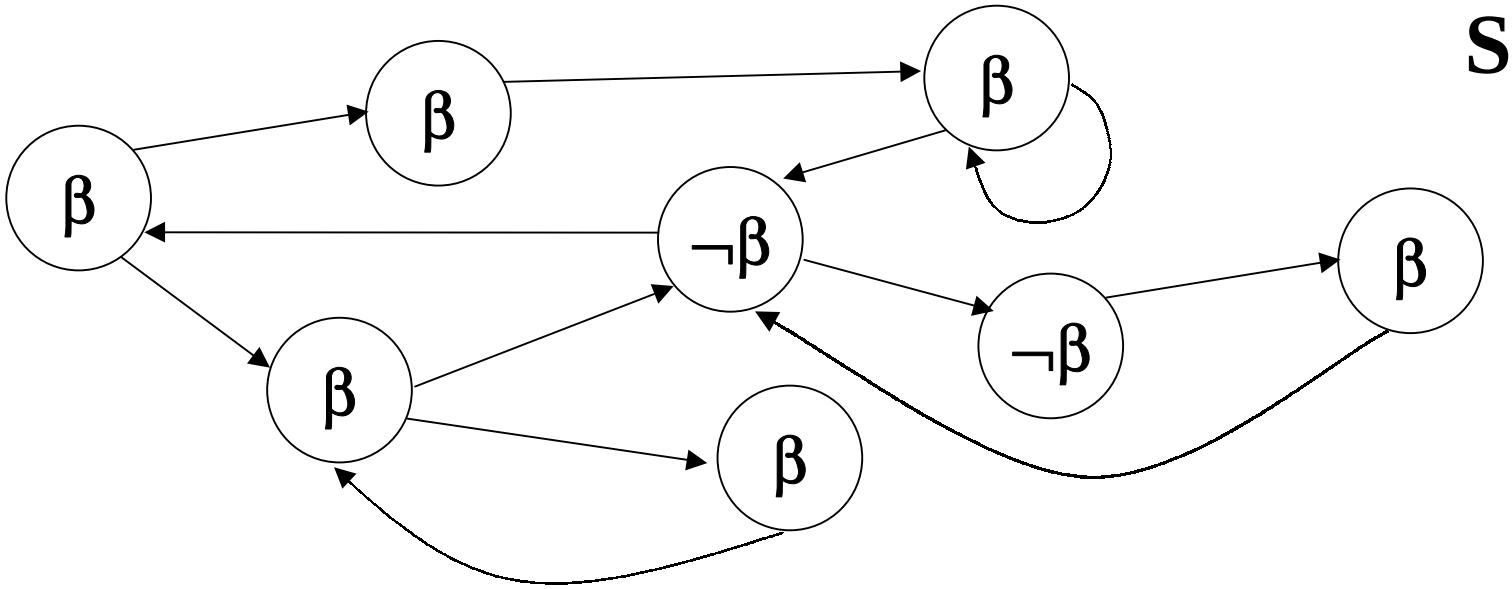
Lemma: M,s ⊨ EG(β) ***iff*** the following two conditions hold:

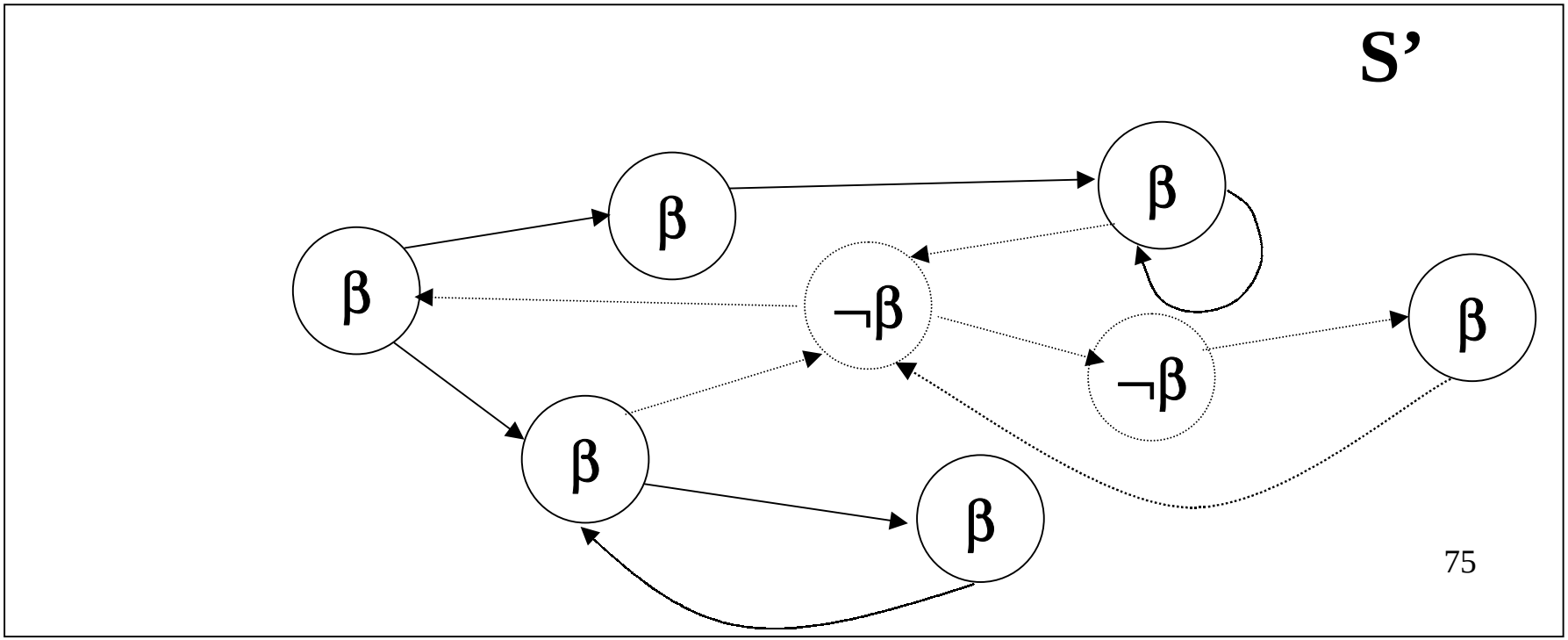
- **s ∈ S' and**
- ***there exists a path*** in **M'** leading from **s** to a ***non-trivial strongly connected component*** **C** of the graph **(S',R').**

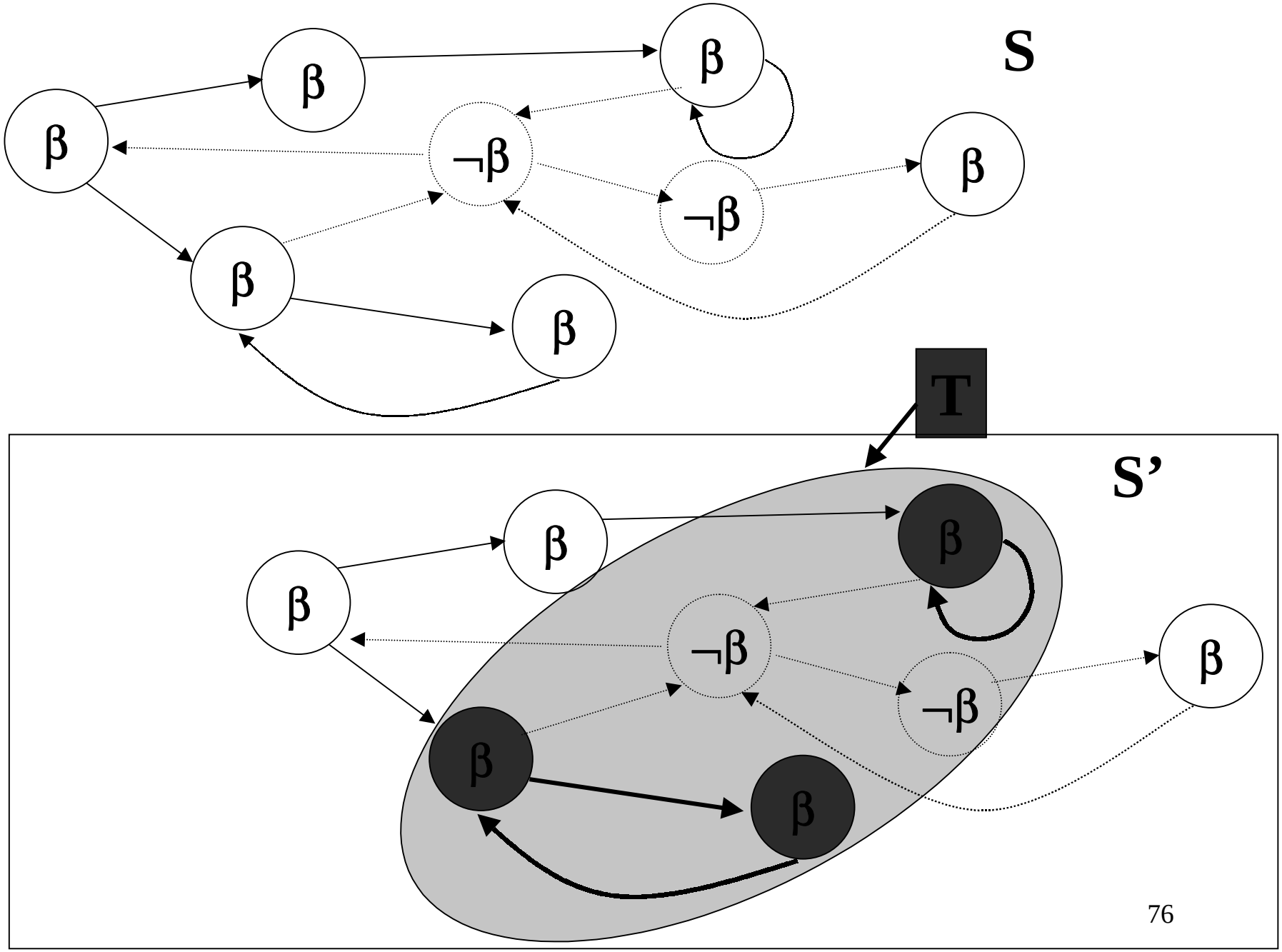
The Labels Function

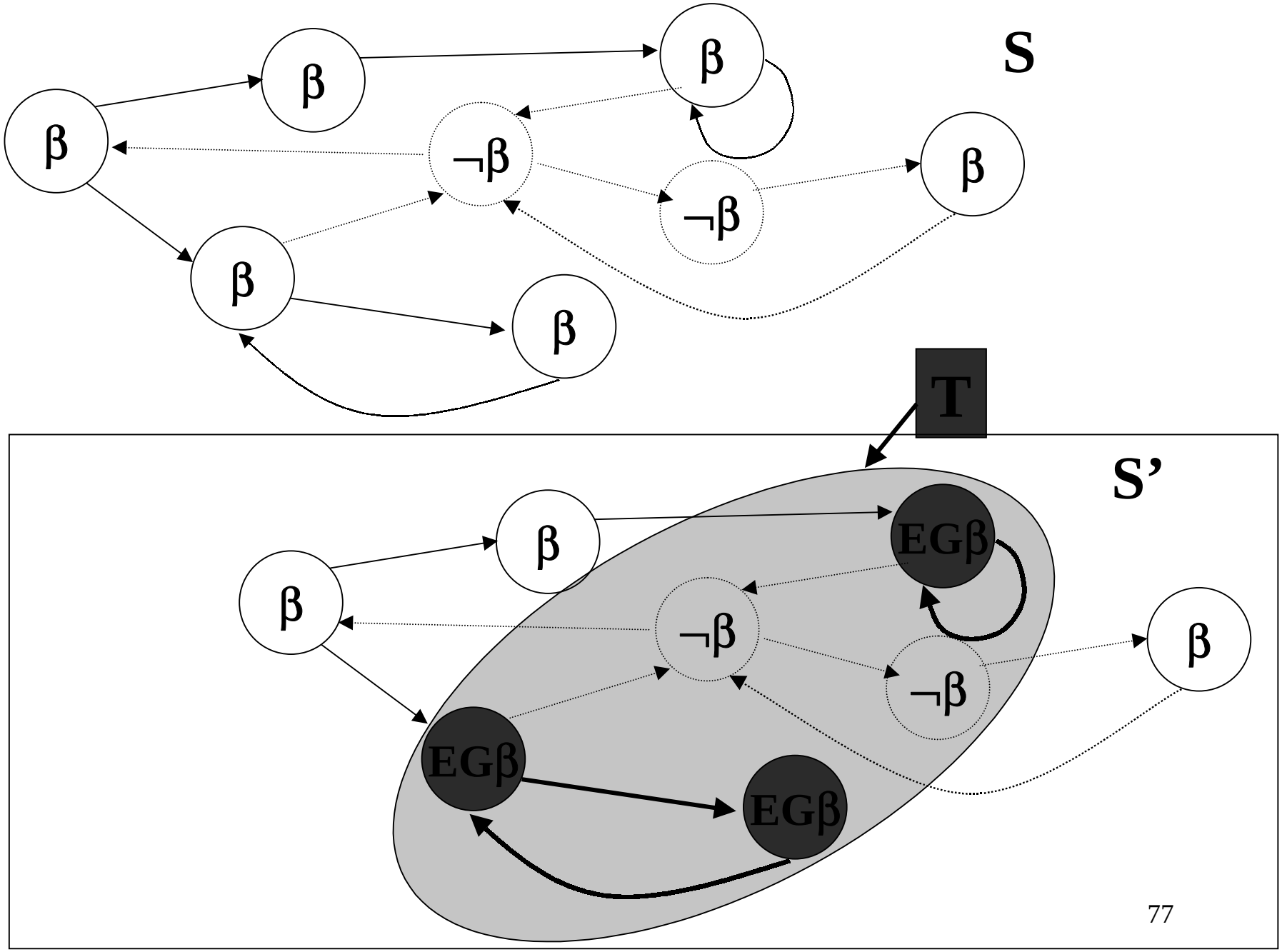
- Compute the ***non-trivial strongly connected components*** of the subgraph **S'** whose states all satisfy β
  - **all the states in these components do satisfy EG(β).**
- Traverse backward **R** in **M'** and label with **EG(β)** the ***states reaching at least a state*** **s** labeled with **EG(β)**.

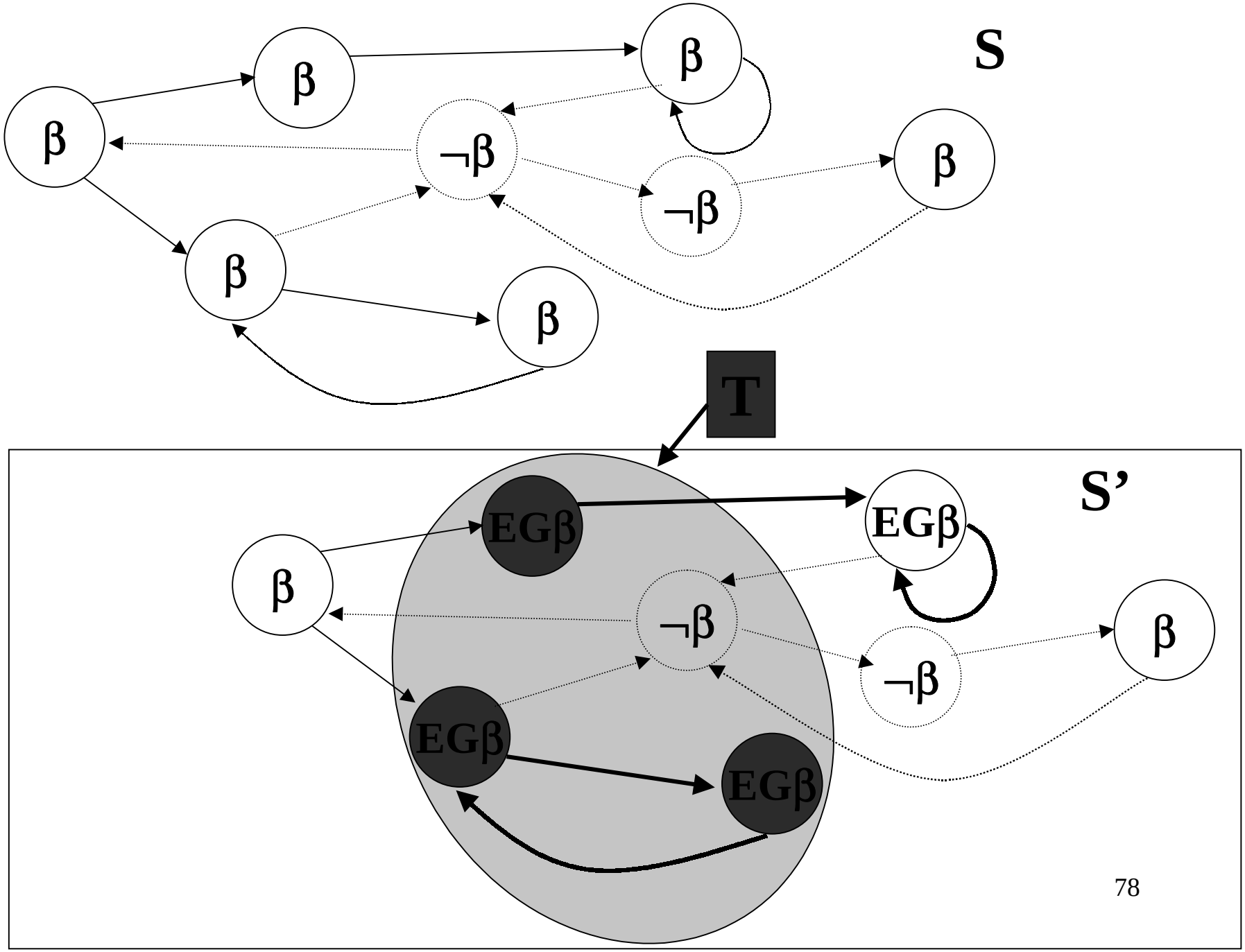
If t ∈ S' and R(t,s) then EG(β) ∈ Labels(t)

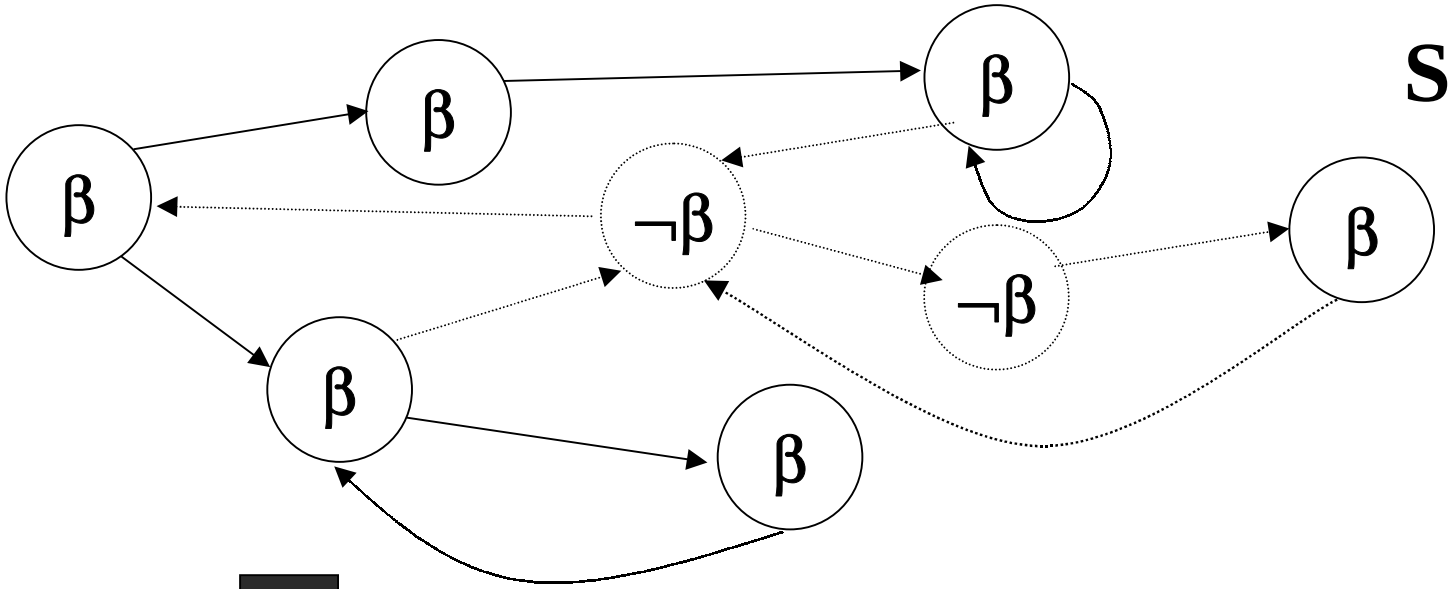

S
β
β
β
¬β
¬β
β
β
β

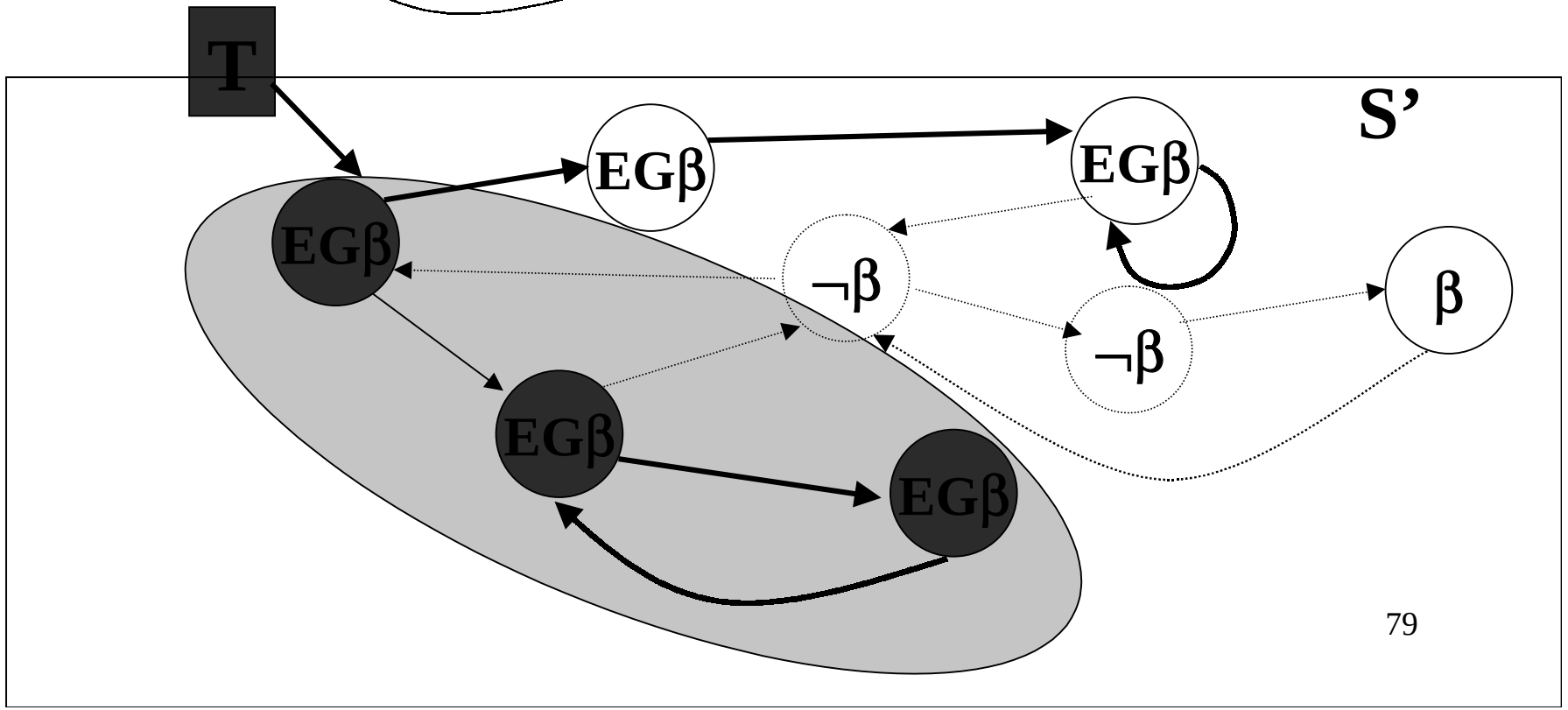

S’
β
β
β
¬β
¬β
β
β
β

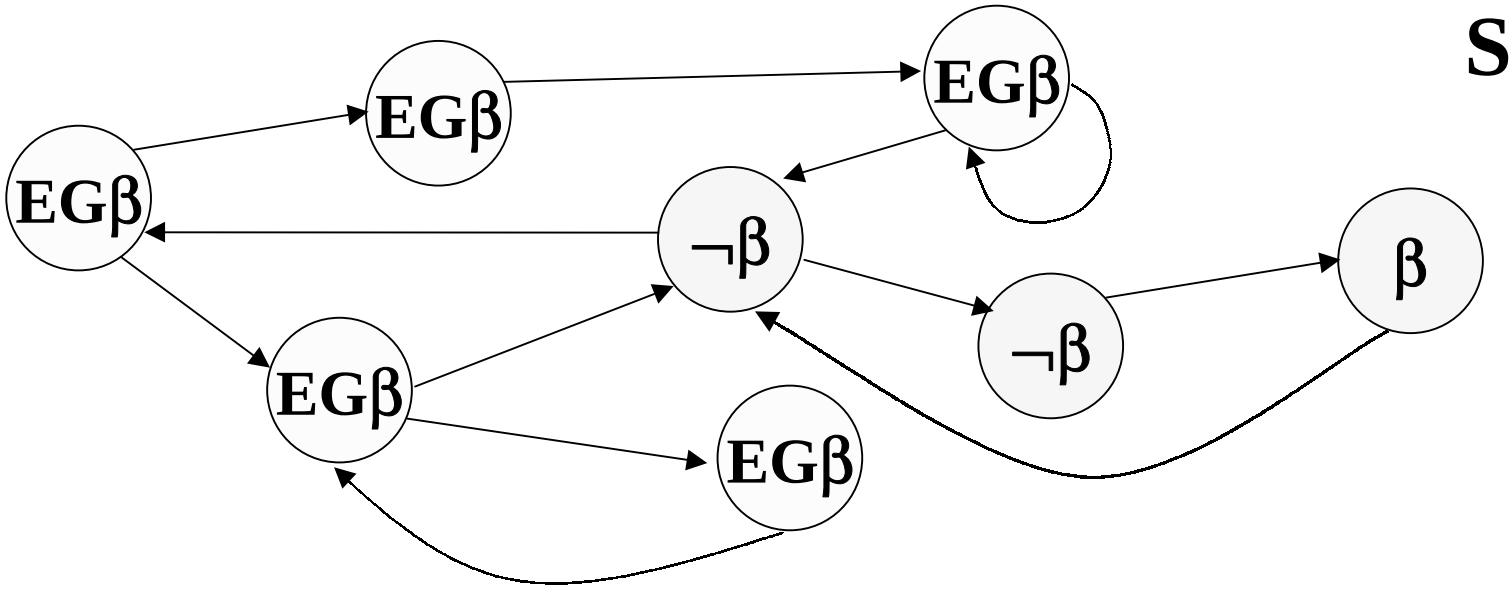

S
β
β
β
¬β
¬β
β
β
β
T
S’
β
β
β
¬β
¬β
β
β
β

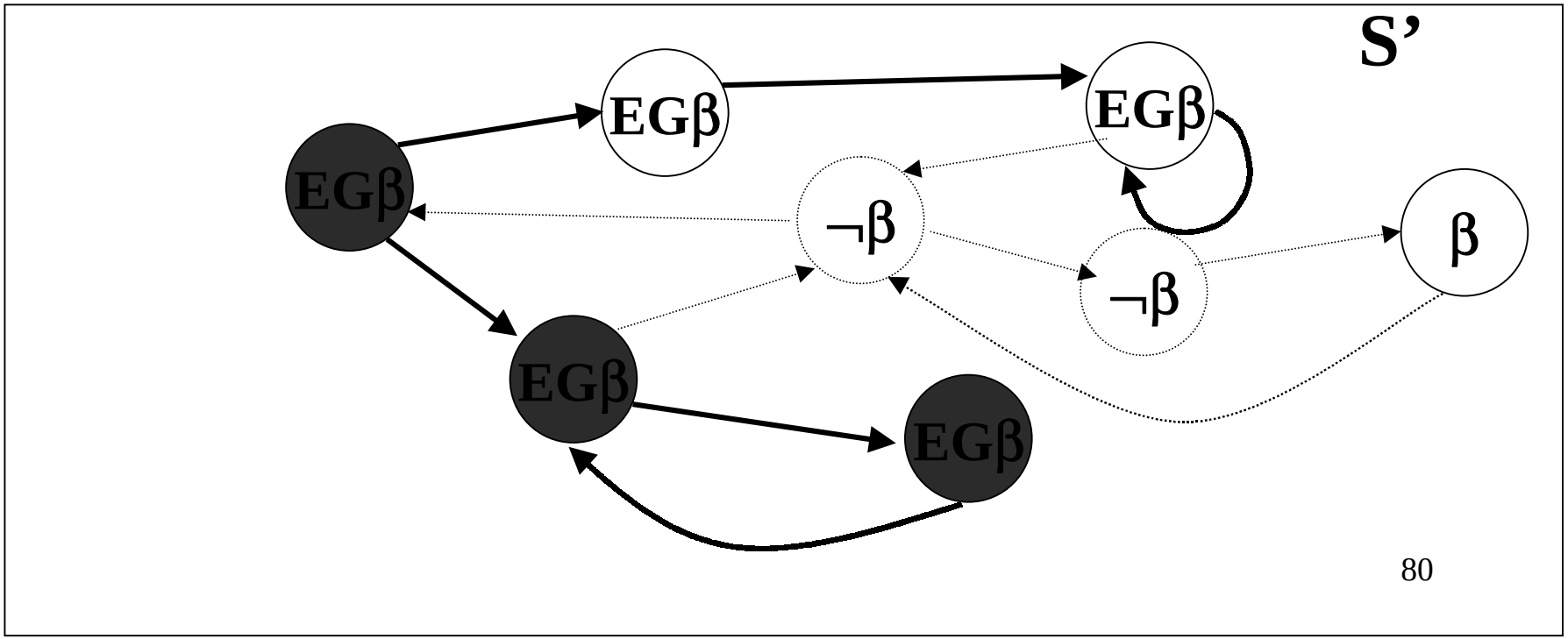

S
β
β
β
¬β
¬β
β
β
β
T
S'
β
β
EGβ
¬β
¬β
β
EGβ
EGβ


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
¬β
¬β
β
EGβ
EGβ


S
EGβ
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**};

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

while T ≠ ∅ do

chose **s** ∈ T;

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

forall **t** ∈ **S'** with **(t,s)**∈ **R** 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 **O(|M|·|ϕ|)**

where **|M|** is the size of the graph underlying **M** and |ϕ| 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 $\langle \mathbf{A}, \sqsubseteq \rangle$ is called a ***partially ordered set*** (or ***poset***).
- **Example:** If **S** is any set and $\subseteq$ is the ordinary subset relation, then $\langle \mathbf{2^S}, \subseteq \rangle$ is a *partially ordered set.*

Upper Bounds

Given <**A, ⊑**> and **A’ ⊆ A**

- **a∈ A** is an ***upper bound*** of **A’** iff ∀**a’∈ A’**, **a’ ⊑ a**
- **a∈ A** is a ***least upper bound*** (***lub***) of **A’**, written **⊔A’**, iff
  - **a** is an ***upper bound*** of **A’** and
  - ∀**a’∈ A**, if **a’** is an ***upper bound*** of **A’**, then **a ⊑ a’**

Lower Bounds

Given <**A, ⊑**> and **A’ ⊆ A**

- **a∈ A** is a ***lower bound*** of **A’** iff ∀**a’∈ A’**, **a ⊑ a’**
- **a∈ A** is a ***greatest lower bound*** (***glb***) of **A’**, written **⊓A’**, iff
  - **a** is a ***lower bound*** of **A’** and
  - ∀**a’∈ A**, if **a’** is a ***lower bound*** of **A’**, then **a’ ⊑ 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 ***least element*** **⊔A=⊥** and a unique ***greatest element*** **⊓A =⊤.**

Complete Lattice

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

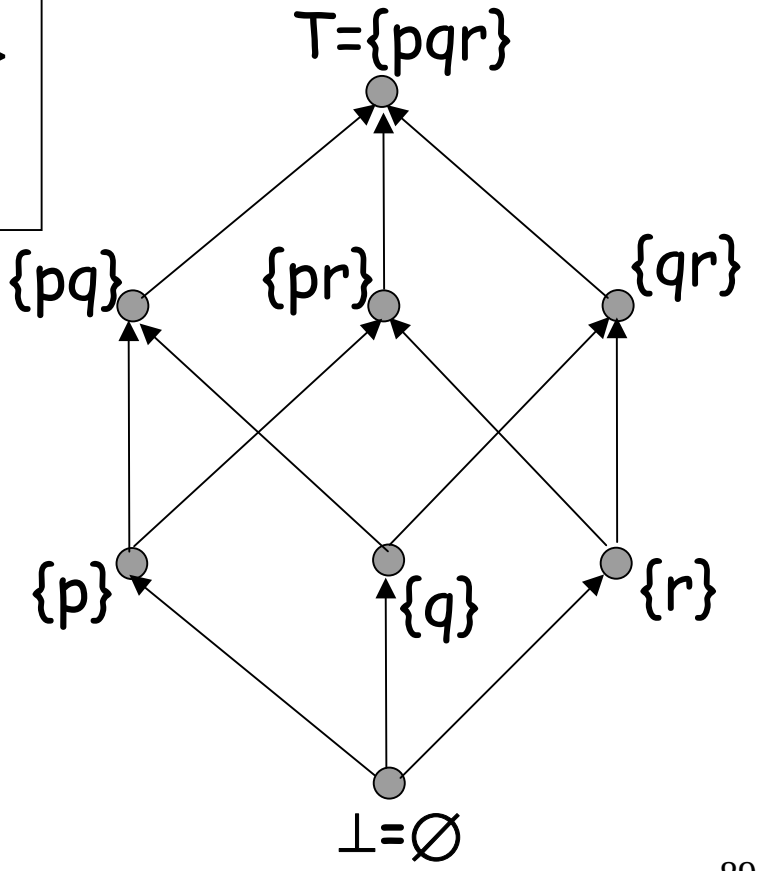
Any two subset of **S** have a ***least upper*** and a ***greatest lower bound***.

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

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

Example of a complete lattice

The complete lattice $<2^S, \subseteq>$ given the set S={p,q,r}



Monotonic functions

- A function **F: A $\rightarrow$ A** is ***monotonic*** if for each **a,b $\in$ A**, **a $\sqsubseteq$ b** implies **F(a) $\sqsubseteq$ F(b)**.
- In other words, a function **F** is monotonic if it ***preserves the ordering*** $\sqsubseteq$.

Fixed points

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

- **$\mu x.F(x) = \sqcup\{x \in A \mid x \sqsubseteq F(x)\}$**
- **$\nu x.F(x) = \sqcap\{x \in A \mid F(x) \sqsubseteq x\}$**

Fixed point in finite lattices

Let $\langle \mathbf{A}, \sqsubseteq \rangle$ be a ***finite complete lattice***, and $\mathbf{F}: \mathbf{A} \rightarrow \mathbf{A}$ 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:

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

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

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

Denotation of a CTL formula

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

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

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

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

- 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:

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

- **…**
- $|[EX\phi]| = \{ s \mid \exists t. (t \in |[\phi]| \cap R(s)) \}$
- 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:

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

in other words:

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

and

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

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

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

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

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

j=1;

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

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

while $\mathbf{T} \neq \emptyset$ do

chose $\mathbf{s} \in \mathbf{T}$;

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

forall $\mathbf{t} \in \mathbf{S}$ do

if $\mathbf{s} \in \mathbf{R(t)}$ then $\mathbf{X} := \mathbf{X} \cup \{\mathbf{t}\}$;

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

This computes

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

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

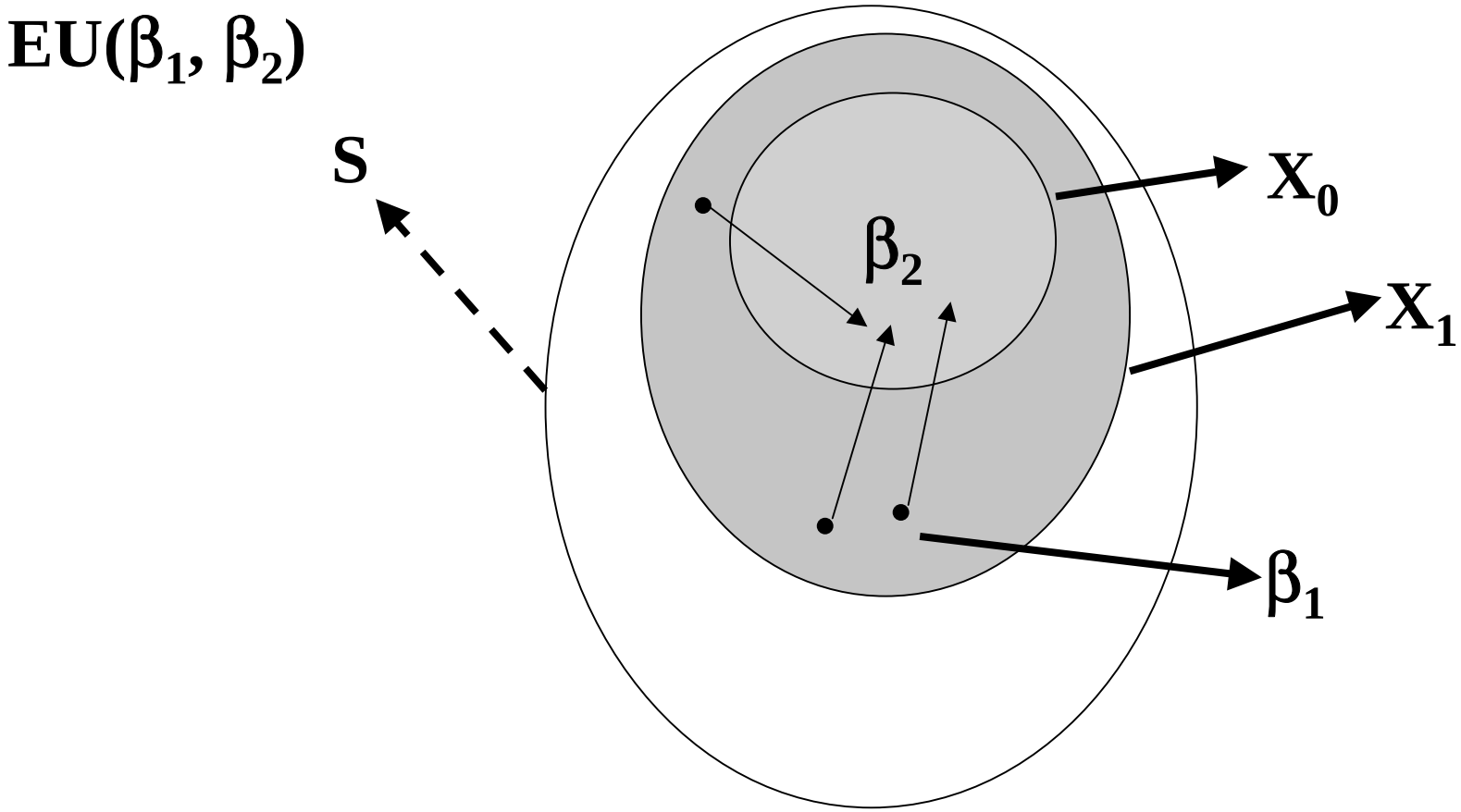
To compute $\mathbf{|[EU(\beta_1,\beta_2)]|}$ we can ***construct inductively the set*** of states $\mathbf{X_j}$ as follows:

- $\mathbf{X_1 = \{s \mid s \in |[\beta_2]|\}}$.
- $\mathbf{X_{j+1} = X_j \cup \{ s \mid s \in |[\beta_1]|}$ 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 $\mathbf{X_{n+1} = X_n}$.

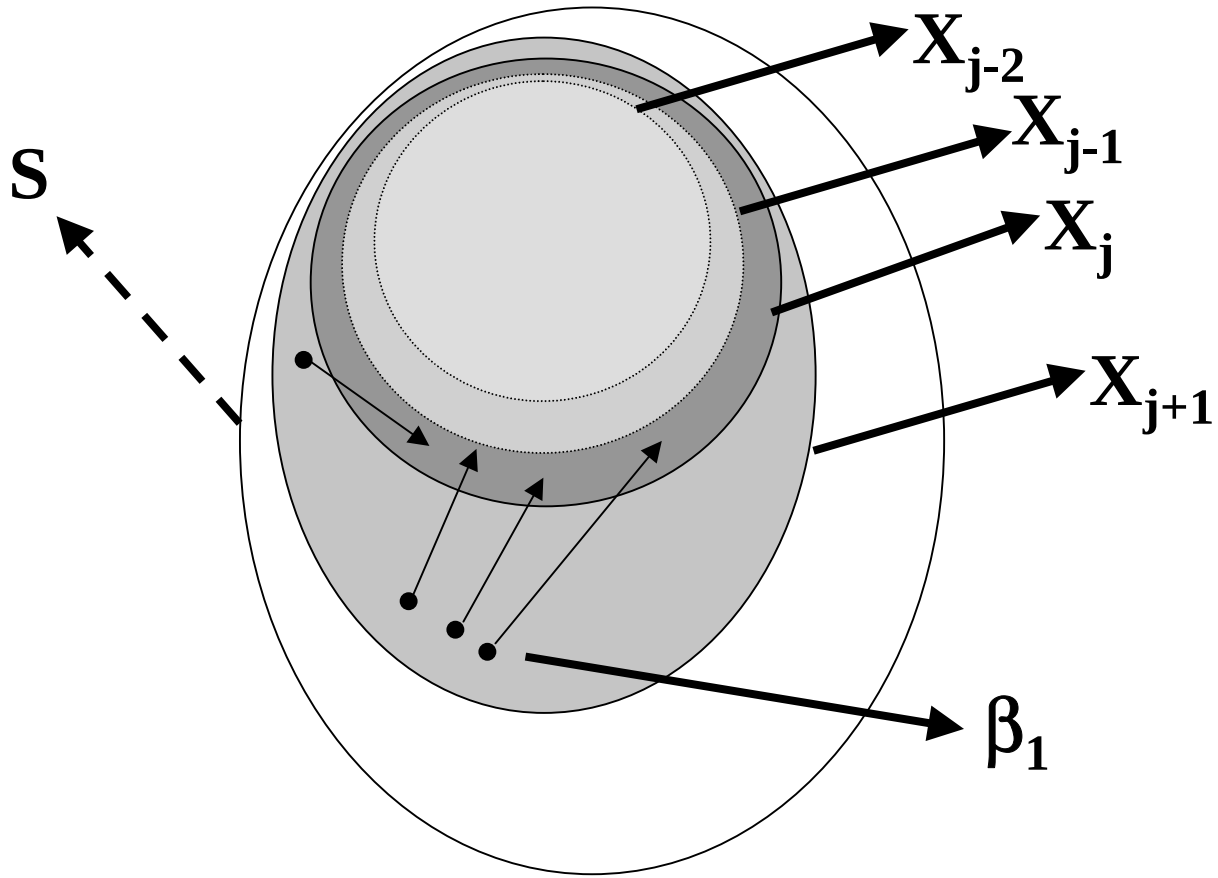
Notice that $\mathbf{n}$ ***must exist*** by ***Tarski's Theorem*** since $\mathbf{X_j \subseteq X_{j+1} \subseteq S}$.

From X_0 to X_1



From X_j to X_{j+1}

EU(β_1, β_2)



Computing fixed point for EU(β_1,β_2)

Algorithm Compute_EU(β_1,β_2)

$\mathbf{X_1} := |[\beta_2]|$;

j=1;

repeat

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

while **T** $\neq \emptyset$ do

chose **s** $\in$ **T**;

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

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

if **t** $\in |[\beta_1]|$ then **X** := **X** $\cup$ {**t**};

$\mathbf{X_j}$:= **X**

until $\mathbf{X_{j-1}} = \mathbf{X_j}$

Fixed point characterization of EG(β)

- **EG(β) ≡ β ∧ EX EG(β)**
- **|[EG(β)]| = νZ.(|[β]| ∩ |[EX Z]|))**
- **|[EG(β)]| =**

  νZ.(|[β]| ∩ { s | ∃t ∈ Z ∩ R(s) })

Computing the fixed point for EG(β)

Algorithm Compute_EG(β)

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

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

j=1;

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

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

while $\mathbf{T} \neq \emptyset$ do

chose $\mathbf{s} \in \mathbf{T}$;

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

forall $\mathbf{t} \in \mathbf{S}$ do

if $\mathbf{s} \in \mathbf{R(t)}$ then $\mathbf{X_j} := \mathbf{X_j} \cup \{\mathbf{t}\}$;

$\mathbf{X_j} := \mathbf{|[\beta]|} \cap \mathbf{X_j}$

The Labels function

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

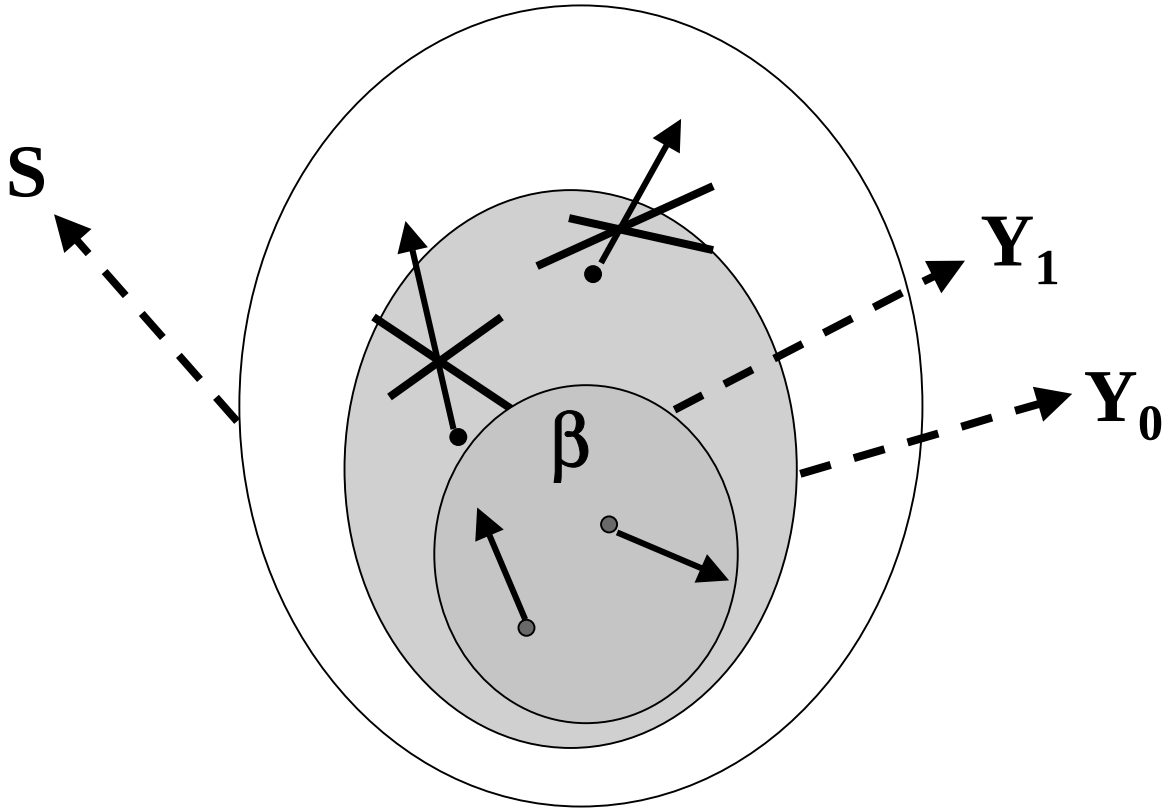
  - $\mathbf{X_1} = |[\beta]|$.
  - $\mathbf{X_{j+1}} = \mathbf{X_j} - \{ \mathbf{s} \mid \mathbf{s} \in \mathbf{X_j}$ and
    there does not exist $\mathbf{t} \in \mathbf{X_j}$
    such that $\mathbf{R(s, t)}\}$

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

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

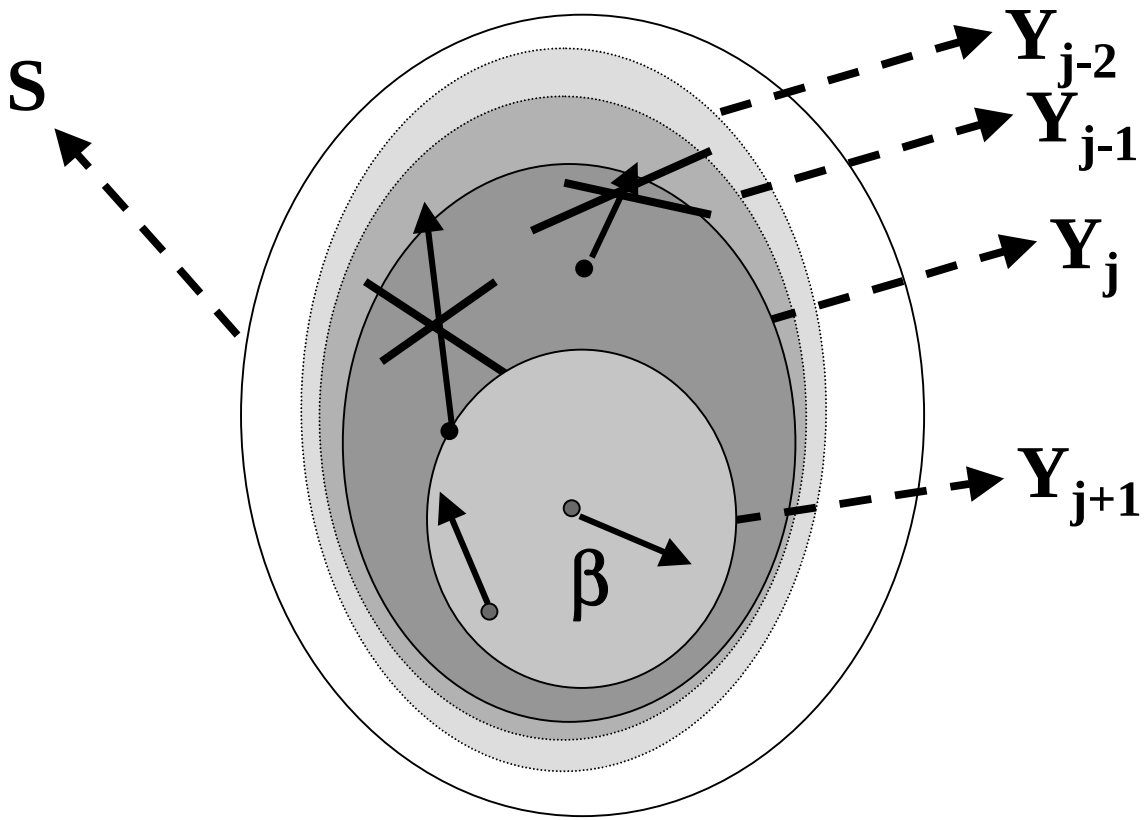
From Y_0 to Y_1

EGβ



From Y_j to Y_{j+1}

EGβ



Computing the fixed point for EG(β)

Algorithm Compute_EG(β)

$\mathbf{X_1}$:= **|[β]|**;

j=1;

repeat

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

while T ≠ ∅ do

chose **s** ∈ T;

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

if for no **t** ∈ **R(s)**, **t** ∈ $\mathbf{X_{j-1}}$ then

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

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