Lecture 31

CS 6260 Automated Planning and Learning

FOND Planning

Department of Computer Science and Engineering

Indian Institute of Technology Madras

Recap: Classical Planning

State-transition system or *classical planning domain*:
$\Sigma = (S,A,T,\text{cost})$ or (S,A,T)

- S - finite set of *states*
- A - finite set of *actions*
- $T: S \times A \to S$

  prediction (or *state-transition*) function
  - *partial* function: $T(s,a)$ is not necessarily defined for every (s,a)
  - a is *applicable* in s iff $T(s,a)$ is defined
  - $\text{Domain}(a) = \{s \in S \mid a \text{ is applicable in } s\}$
  - $\text{Range}(a) = \{T(s,a) \mid s \in \text{Domain}(a)\}$
- $C: S \times A \to \mathbb{R}^+$ or $C: A \to \mathbb{R}^+$
  - optional; default is $\text{cost}(a) \equiv 1$
  - money, time, something else

- *plan*: a sequence of actions $\pi = \langle a_1, \ldots, a_n \rangle$
- π is *applicable* in s_0 if the actions are applicable in the order given

  $T(s_0, a_1) = s_1$

  $T(s_1, a_2) = s_2$

  $\ldots$

  $T(s_{n-1}, a_n) = s_n$
  - In this case define $T(s_0, \pi) = s_n$, *and* $\hat{T}(s_0,\pi) = \langle s_1,\ldots, s_n \rangle$

- *Classical planning problem*:
  - $P = (\Sigma, s_0, S_g)$
  - planning domain, initial state, set of goal states
- *Solution* for P:
  - a plan π such that that $T(s_0,\pi) \in S_g$

FOND Planning*

State-transition system or *classical planning domain*: $\Sigma = (S,A,T)$ or (S,A,T)

- S - finite set of *states*
- A - finite set of *actions*
- $T: S \times A \times S$

  prediction (or *state-transition*) function
  - *partial* function: $T(s,a,s')$ is not necessarily defined for every (s,a,s')
  - a is *applicable* in s iff $T(s,a,s')$ is defined
  - Domain$(a) = \{s \in S \mid a \text{ is applicable in } s\}$
  - An action is nondeterministic if $|EXEC(s,a)| > 1$.

- *FOND planning problem*:
  - $P = (\Sigma, s_0, S_g)$
  - planning domain, initial state, set of goal states
- *Solution* for P:
  - a policy π

* Cimatti et al., Weak, strong, and strong cyclic planning via symbolic model checking, Artificial Intelligence, Vol. 147, Iss. 1–2, 2003.

FOND Planning: Action Properties

- $ACT(s) = \{a : \exists s'.\ T(s,a,s')\}$
- $EXEC(s,a) = \{s' : T(s,a,s')\}$
- An action is nondeterministic if $|EXEC(s,a)| > 1$.
- An action is deterministic if $|EXEC(s,a)| = 1$.

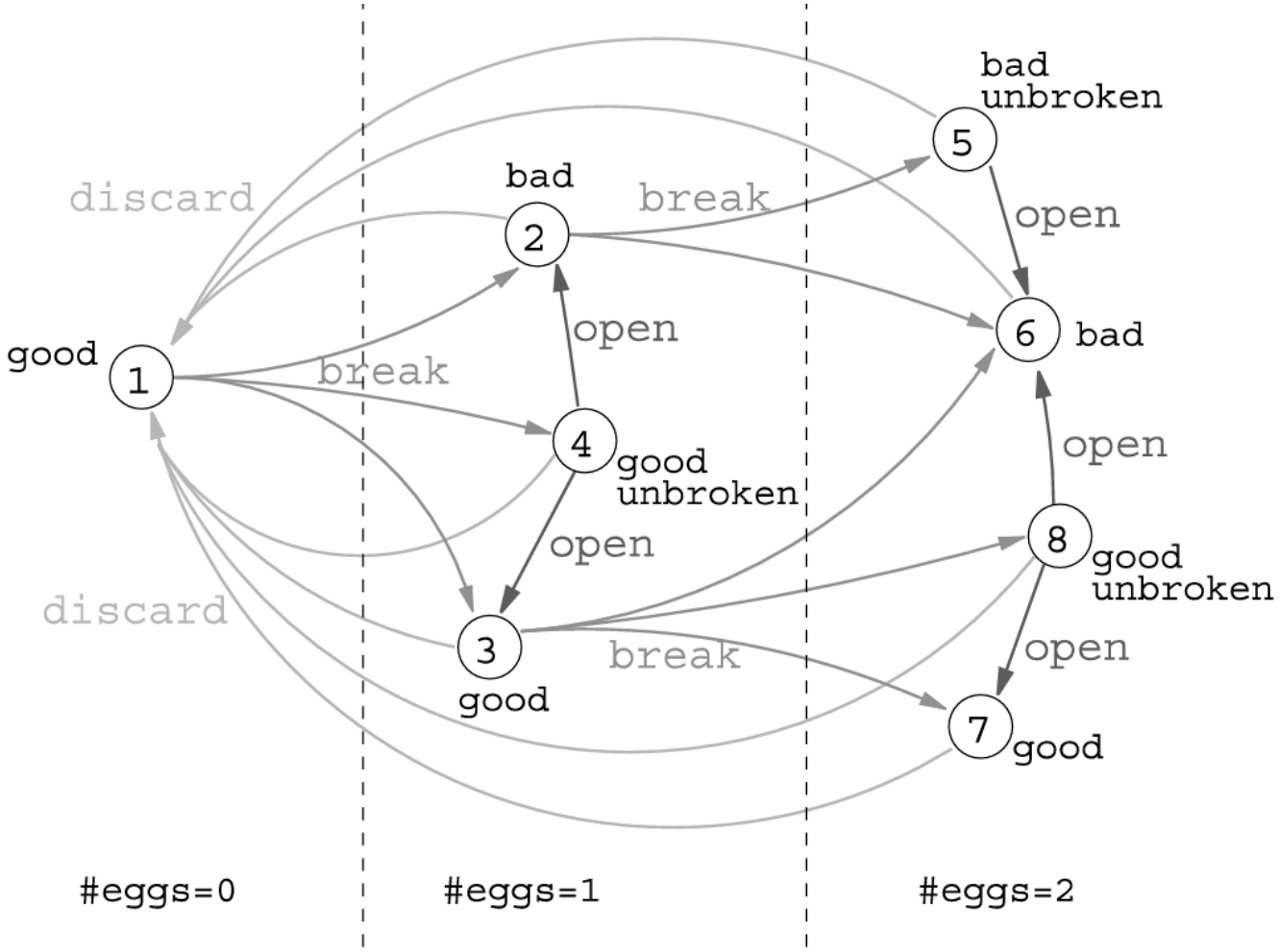
* Cimatti et al., Weak, strong, and strong cyclic planning via symbolic model checking, Artificial Intelligence, Vol. 147, Iss. 1–2, 2003.

Running Example: Omelette Preparation

- Propositions P:
  {#eggs=0,
  #eggs=1,
  #eggs=2,
  bad,
  good,
  unbroken}

- State S: 2^P

- Actions: {break,open,discard}



* Cimatti et al., Weak, strong, and strong cyclic planning via symbolic model checking, Artificial Intelligence, Vol. 147, Iss. 1–2, 2003.

Naïve Solution Algorithm

$s :=$ SENSECURRENTSTATE();

while $s \in$ STATESOF(π) **do**

    $a :=$ GETACTION(s, π);

    EXECUTE(a);

    $s :=$ SENSECURRENTSTATE();

done

FOND Actions: break-first-egg

```
(:action break-first-egg
        :precondition (and
                (eggs-0)
                (not (unbroken))
        )
        :effect (oneof
                ;; Outcome 1: egg breaks cleanly and is good
                (and (not (eggs-0)) (eggs-1) (good) (not (bad)))

                ;; Outcome 2: egg breaks cleanly and is bad
                (and (not (eggs-0)) (eggs-1) (not (good)) (bad))

                ;; Outcome 3: egg fails to break open (still unknown good/bad)
                (and (not (eggs-0)) (eggs-1) (good) (unbroken))
        )
)
```

FOND Actions: break-second-egg

```
(:action break-second-egg
        :precondition (and
                (eggs-1) (good) (not (unbroken))
        )
        :effect (oneof
                ;; Outcome 1: second egg breaks cleanly and is good
                (and (not (eggs-1)) (eggs-2) (good))

                ;; Outcome 2: second egg breaks cleanly and is bad
                (and (not (eggs-1)) (eggs-2) (not (good)) (bad))

                ;; Outcome 3: second egg fails to break open
                (and (not (eggs-1)) (eggs-2) (good) (unbroken))
        )
)
```

FOND Actions: open-egg

```
(:action open-egg
        :precondition (and
                (unbroken)
        )
        :effect (oneof
                ;; Outcome 1forced-open egg turns out good
                (and (not (unbroken)) (good))

                ;; Outcome 2: forced-open egg turns out bad
                (and (not (unbroken)) (not (good)) (bad))

        )
)
```

```
(:action discard
        :precondition (or
                (eggs-1) (eggs-2)
        )
        :effect (and
            (eggs-0)
            (not (eggs-1)) (not (eggs-2))
            (not (good)) (not (bad)) (not (unbroken))
            (bowl-empty)
        )
)
```

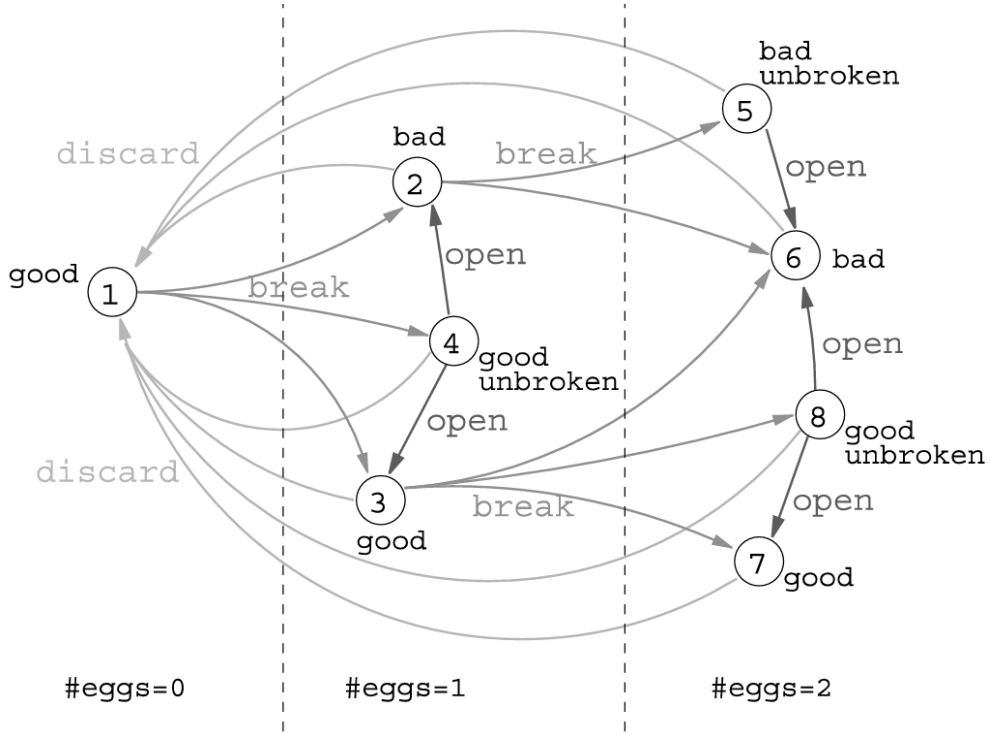
FOND Problem

```
(define (problem omelette-goal)
        (:domain omelette)
        (:init
                eggs-0)
                (good)
                (bowl-empty)
         )

        ;; Goal: two good eggs in the bowl, no bad ones
        (:goal
                 (and (eggs-2) (good) (not (bad)) (not (unbroken)))
        )
)
```

State Action Table

- A state-action table π is a set of pairs ⟨s, a⟩ where s ∈ S and a ∈ ACT(s). It is deterministic if at most one action is associated with each state. I serves as a reactive policy — the agent senses the world and picks an action at each step.



$$\pi_a \doteq \{\langle 1, \text{break}\rangle, \langle 3, \text{break}\rangle\},$$

$$\pi_b \doteq \{\langle 1, \text{break}\rangle, \langle 2, \text{break}\rangle, \langle 3, \text{break}\rangle, \langle 4, \text{open}\rangle, \langle 5, \text{open}\rangle, \langle 8, \text{open}\rangle\}.$$

$$\pi_c \doteq \{\langle 1, \text{break}\rangle, \langle 2, \text{discard}\rangle, \langle 3, \text{break}\rangle, \langle 4, \text{open}\rangle, \langle 6, \text{discard}\rangle, \langle 8, \text{open}\rangle\}.$$

* Cimatti et al., Weak, strong, and strong cyclic planning via symbolic model checking, Artificial Intelligence, Vol. 147, Iss. 1–2, 2003.

Execution Structure

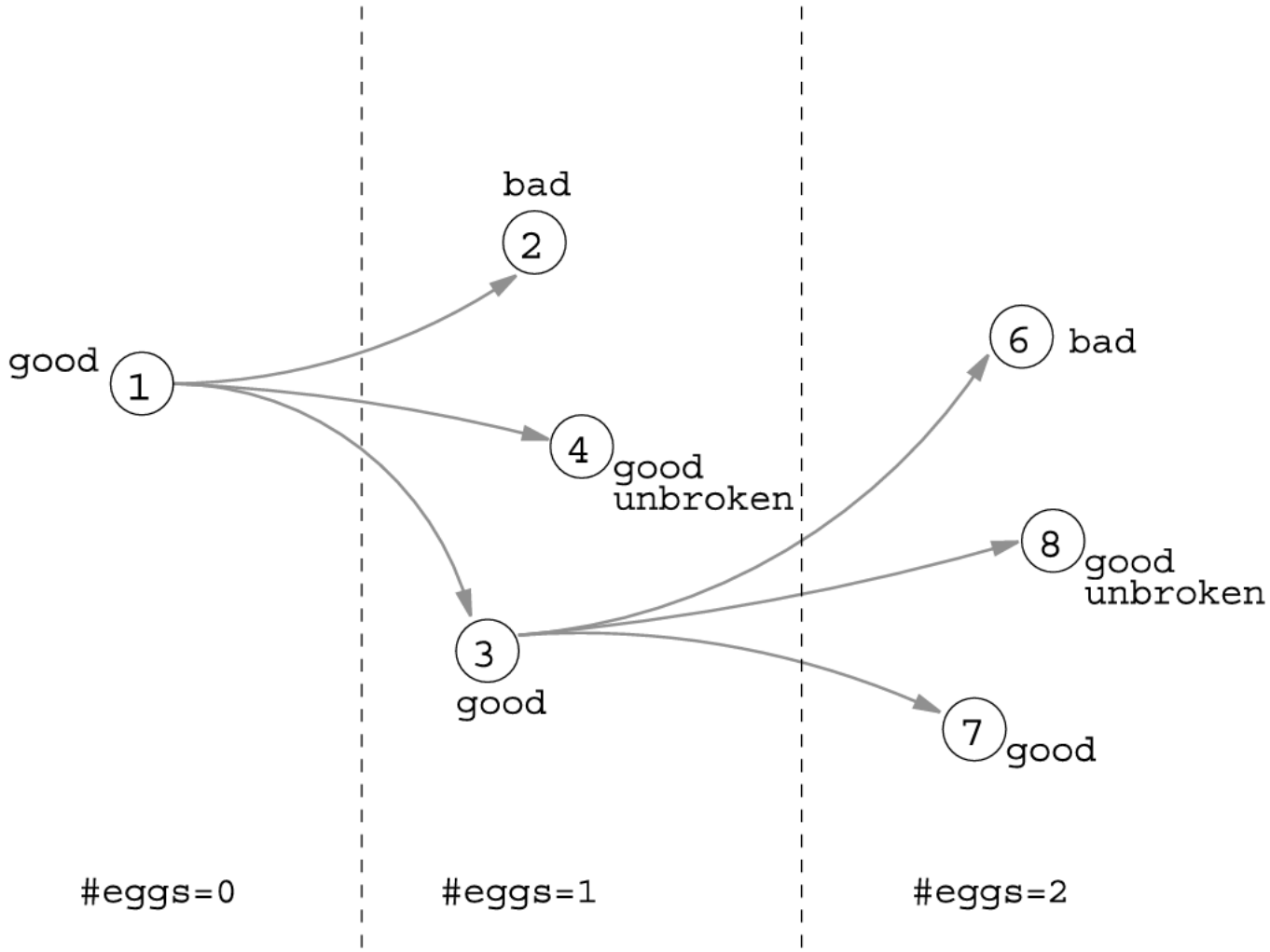
- Let π be a state-action table of a planning domain D
  The execution structure induced by π from the set of initial states I ⊆ S is a tuple K =<Q, L>, where Q⊆S and L ⊆ S ×S are the minimal sets satisfying the following rules:

1. if s ∈ I,then s ∈Q, and
2. if s ∈ Q and there exists a state-action pair s, a ∈ π such that T(s,a,s'), then s' ∈Q and L(s,s').

- A state s ∈Q is a terminal state of K if there is no s' ∈Q such that T(s,s').

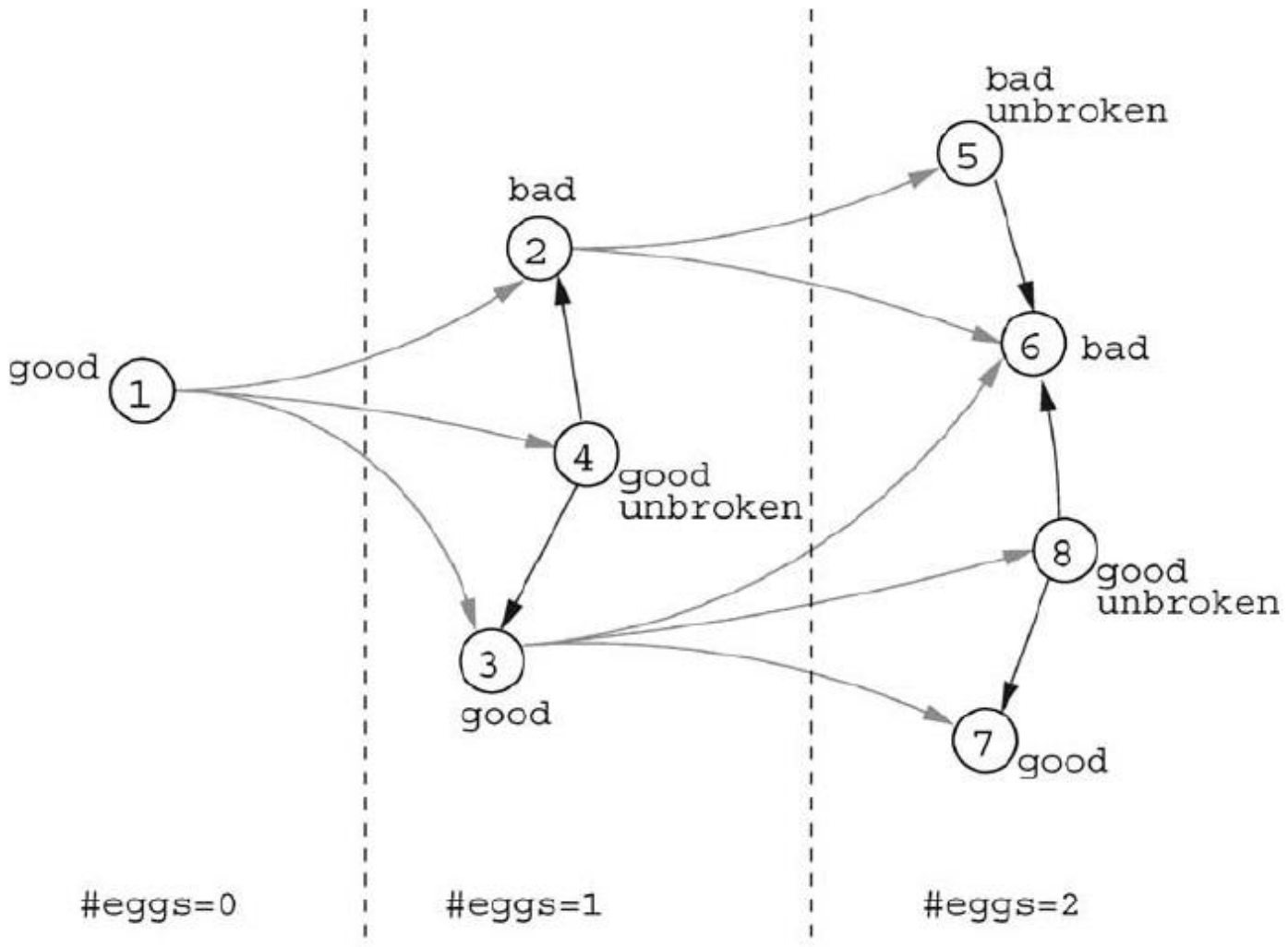
* Cimatti et al., Weak, strong, and strong cyclic planning via symbolic model checking, Artificial Intelligence, Vol. 147, Iss. 1–2, 2003.

Execution Structure: π_a



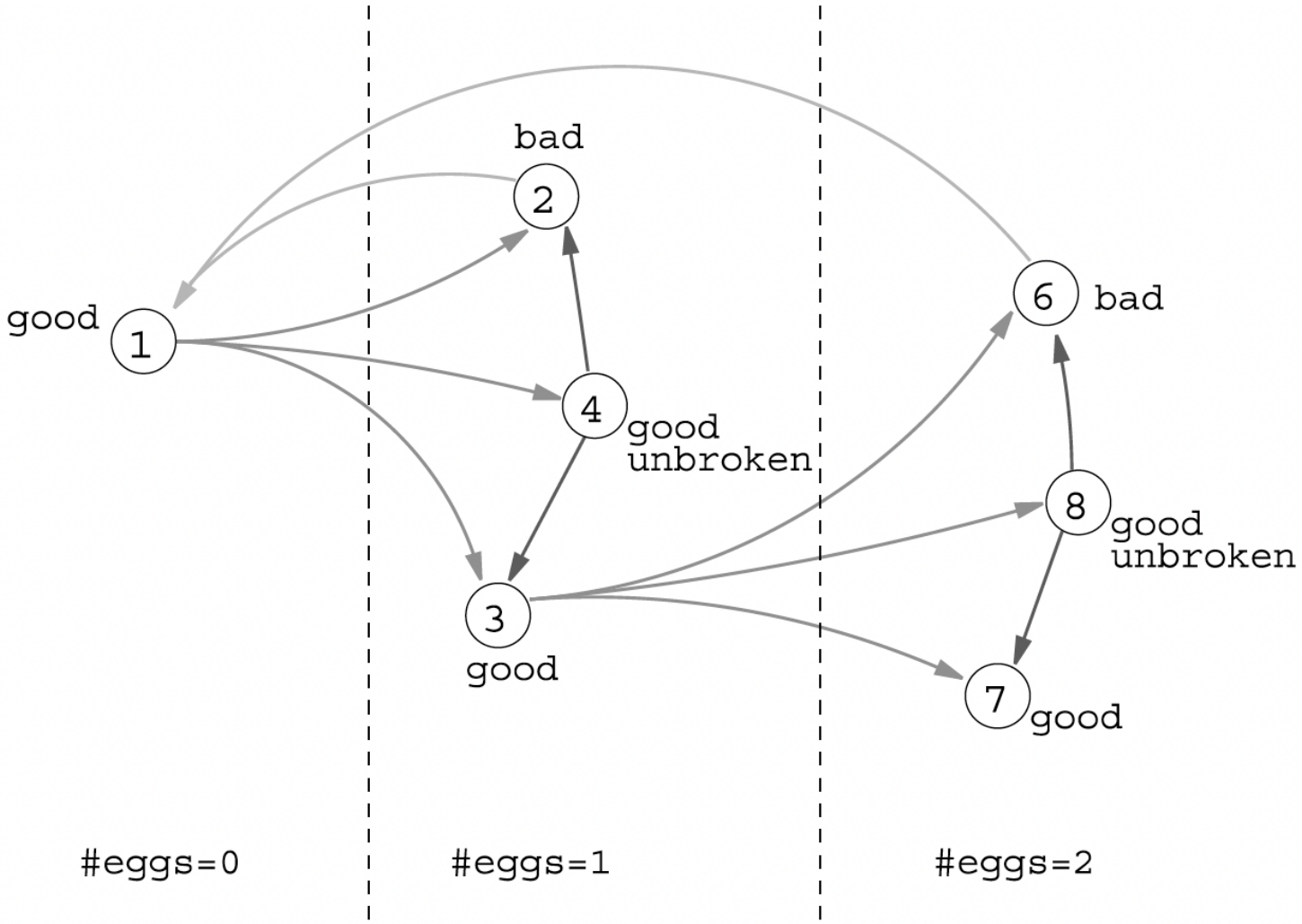
* Cimatti et al., Weak, strong, and strong cyclic planning via symbolic model checking, Artificial Intelligence, Vol. 147, Iss. 1–2, 2003.

Execution Structure : π_b



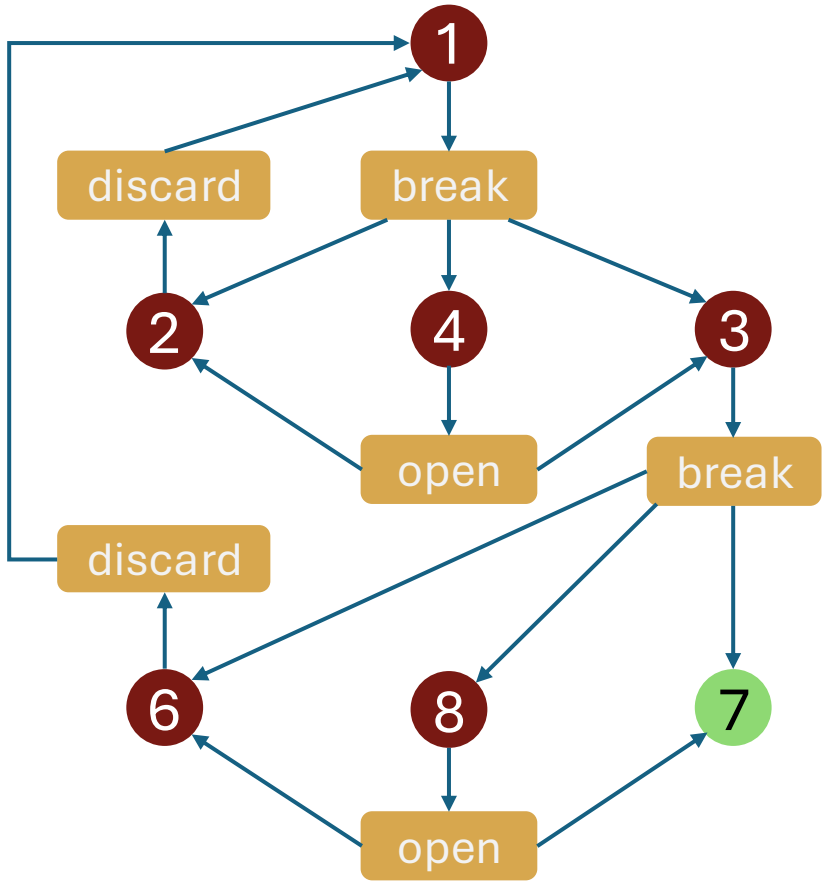
* Cimatti et al., Weak, strong, and strong cyclic planning via symbolic model checking, Artificial Intelligence, Vol. 147, Iss. 1–2, 2003.

Execution Structure : π_c

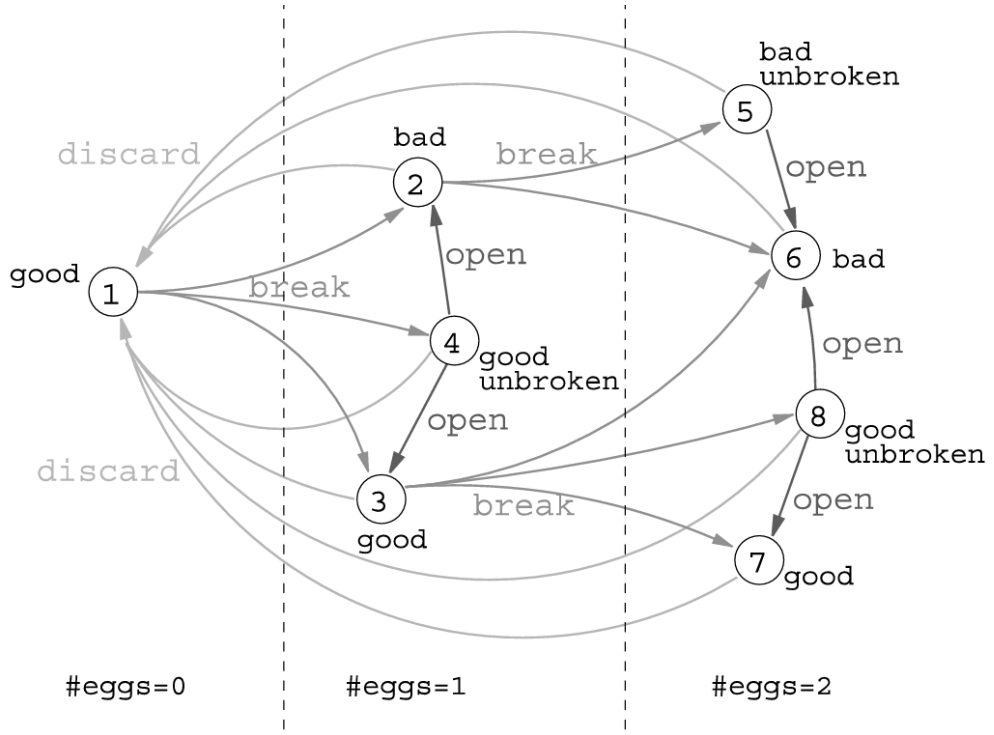


* Cimatti et al., Weak, strong, and strong cyclic planning via symbolic model checking, Artificial Intelligence, Vol. 147, Iss. 1–2, 2003.

Solution Policy



5



action node

state node

Solution Policy Properties

Weak Solution

π is weak if for any s ∈ I, some terminal state of K reachable from s is in G.

Strong Solution

π is strong if K is acyclic and ALL terminal states of K are in G.

Strong Cyclic Solution

π is strong cyclic if from every s ∈ Q some terminal state of K is reachable and all terminal states are in G.

* Cimatti et al., Weak, strong, and strong cyclic planning via symbolic model checking, Artificial Intelligence, Vol. 147, Iss. 1–2, 2003.

FOND Planner and Additional Material

- PRP Planner: https://cdn.aaai.org/ojs/13520/13520-40-17038-1-2-20201228.pdf

- FOND Domains: https://github.com/AI-Planning/fond-domains