Interval Iteration Algorithm for MDPs and IMDPs

Serge Haddad (LSV, ENS Cachan, CNRS & Inria)
and
Benjamin Monmege

Séminaire Modélisation et Vérification
November 2015, Marseille

Mixing non-determinism and probabilities

- Acting in an *uncertain* world
  - non-determinism: *decisions* of an agent;
  - probabilities: effects of the decisions;
  - goal: *maximizing* some utility function.

Mixing non-determinism and probabilities

- Acting in an *uncertain* world
  - non-determinism: *decisions* of an agent;
  - probabilities: effects of the decisions;
  - goal: *maximizing* some utility function.
- Randomness against the *environment*
  - probabilities: distributed *randomized* algorithm;
  - non-determinism: behavior of the network;
  - goal: evaluating the *worst-case* behavior.

Mixing non-determinism and probabilities

- Acting in an *uncertain* world
  - ✦ non-determinism: *decisions* of an agent;
  - ✦ probabilities: effects of the decisions;
  - ✦ goal: *maximizing* some utility function.
- Randomness against the *environment*
  - ✦ probabilities: distributed *randomized* algorithm;
  - ✦ non-determinism: behavior of the network;
  - ✦ goal: evaluating the *worst-case* behavior.

Optimization problems

Markov Decision Processes

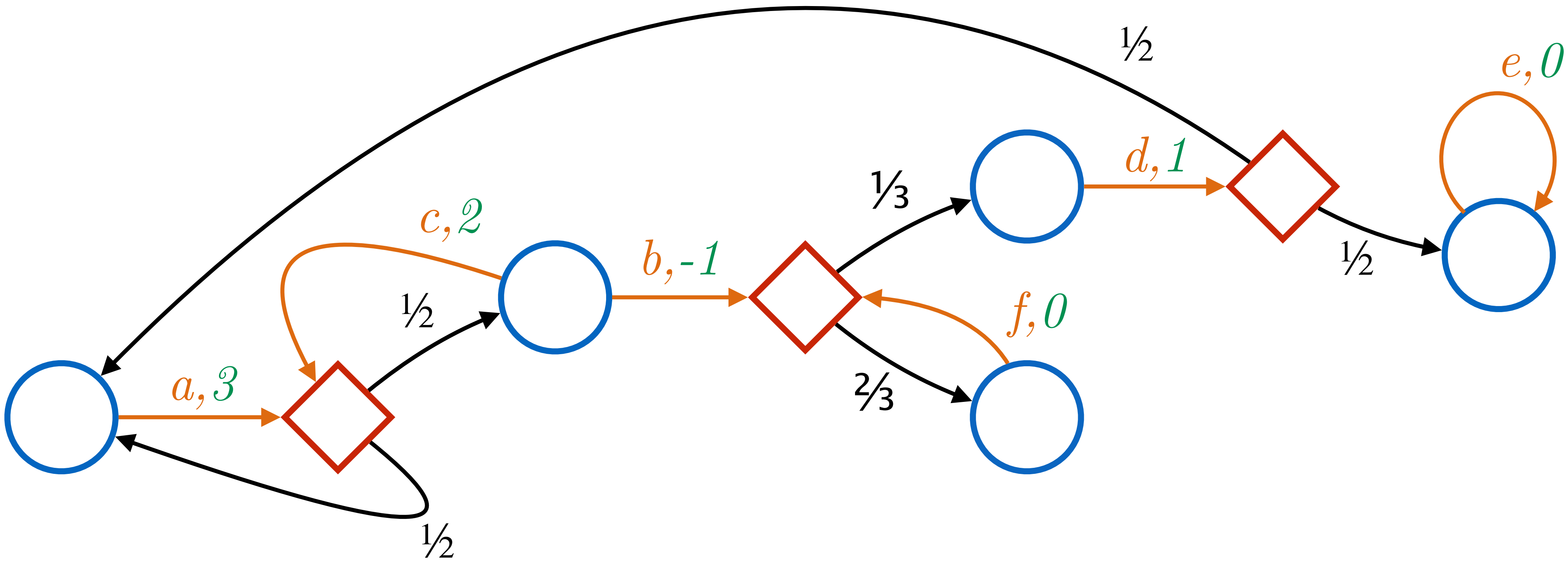
- What?
  - (Finite) *stochastic* process with *non-determinism*
  - Non-determinism solved by *policies*/strategies
  - *Rewards* based on the pair of state and action

Markov Decision Processes

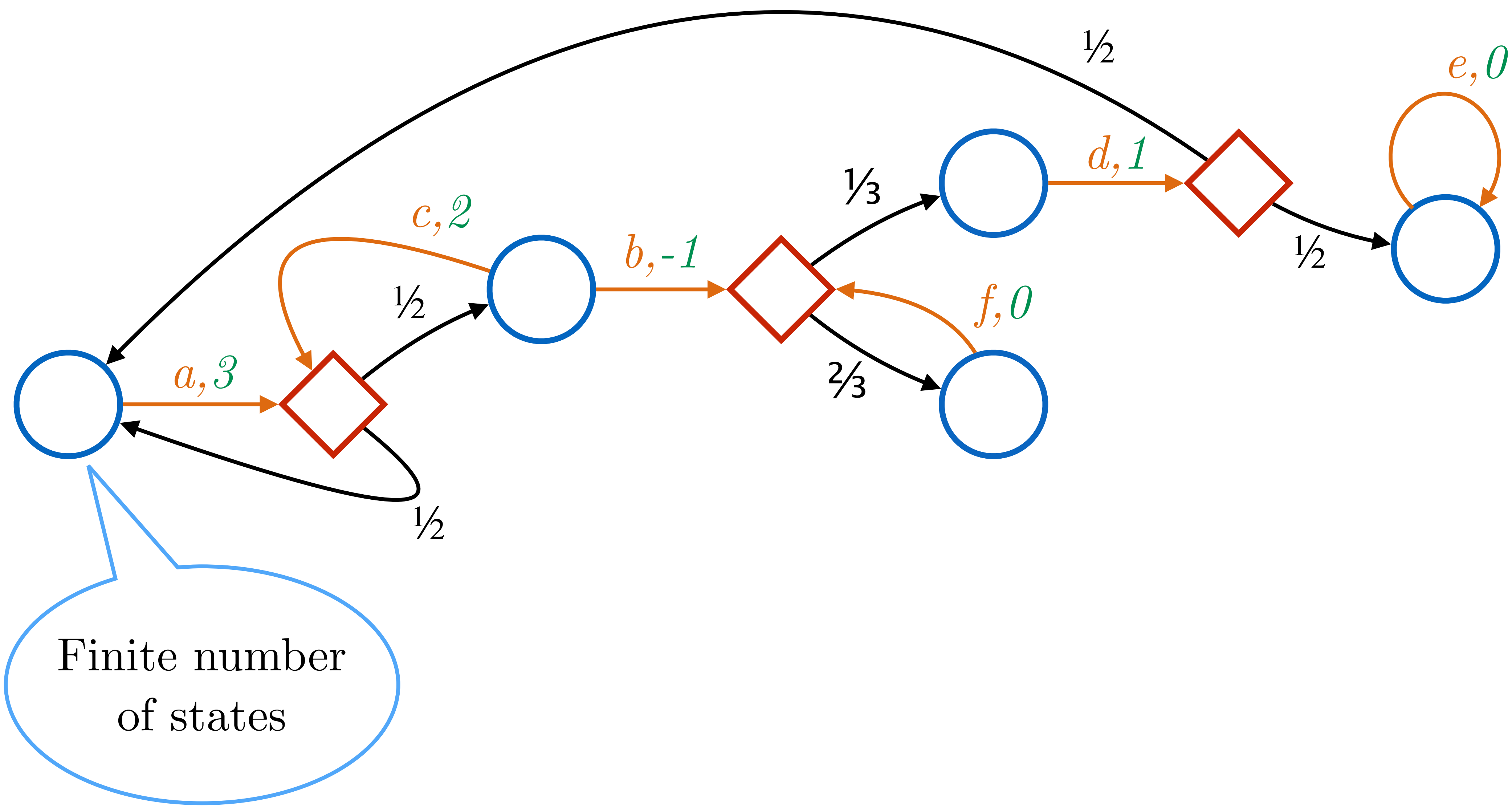
- What?
  - (Finite) *stochastic* process with *non-determinism*
  - Non-determinism solved by *policies*/strategies
  - *Rewards* based on the pair of state and action

- Where?
  - *Optimization*
  - *Program verification*: PCTL model-checking...
  - *Game theory*: 1+½ players

MDPs with discounted rewards



MDPs with discounted rewards



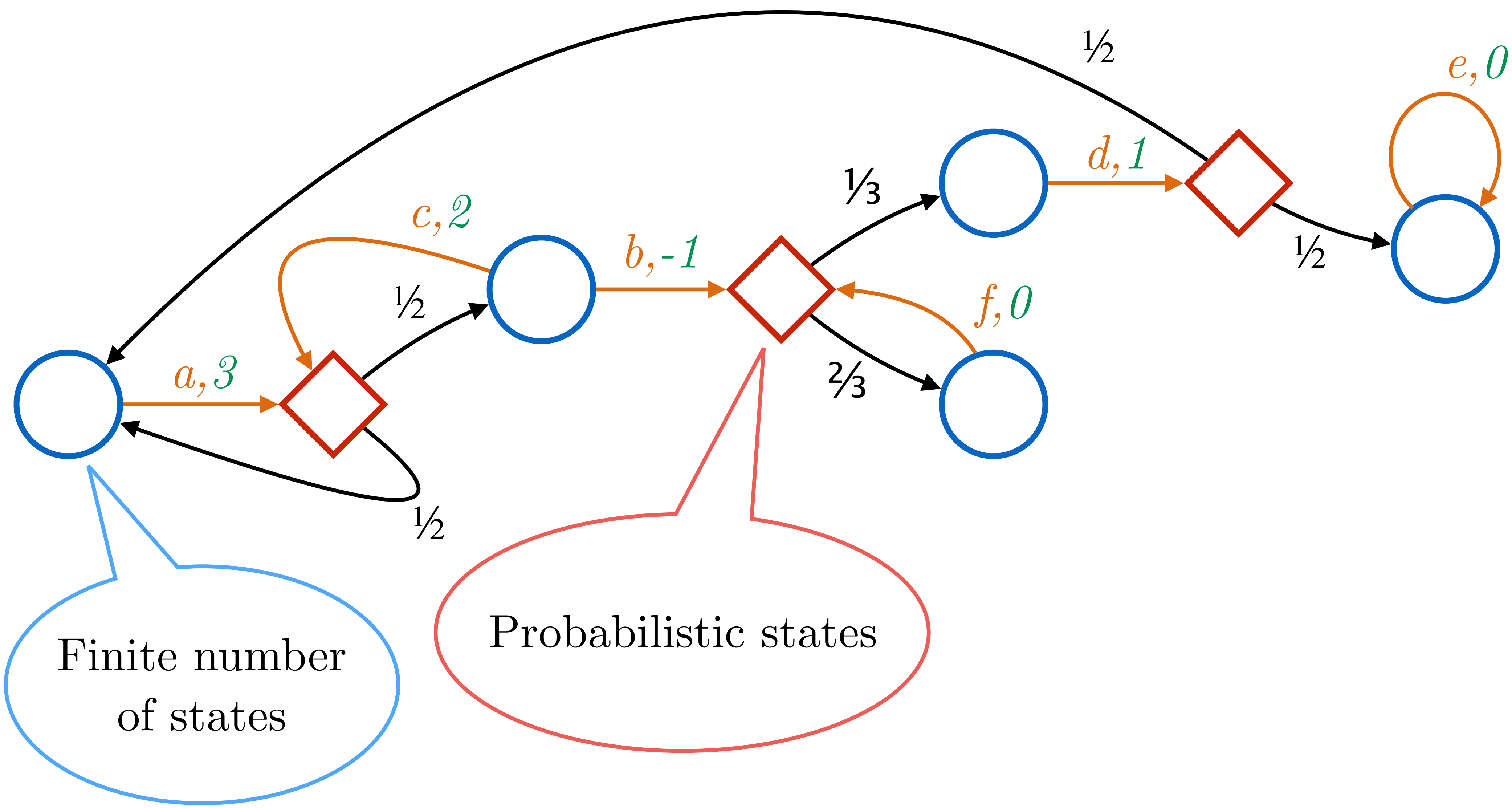
$\mathcal{M} = (S, \alpha, \delta, r) \qquad 0 < \lambda < 1$

$\delta : S \times \alpha \longrightarrow Dist(S)$

$r : S \times \alpha \longrightarrow \mathbb{R}$

Policy $\sigma : (S \cdot \alpha)^{\star} \cdot S \longrightarrow Dist(\alpha)$

MDPs with discounted rewards



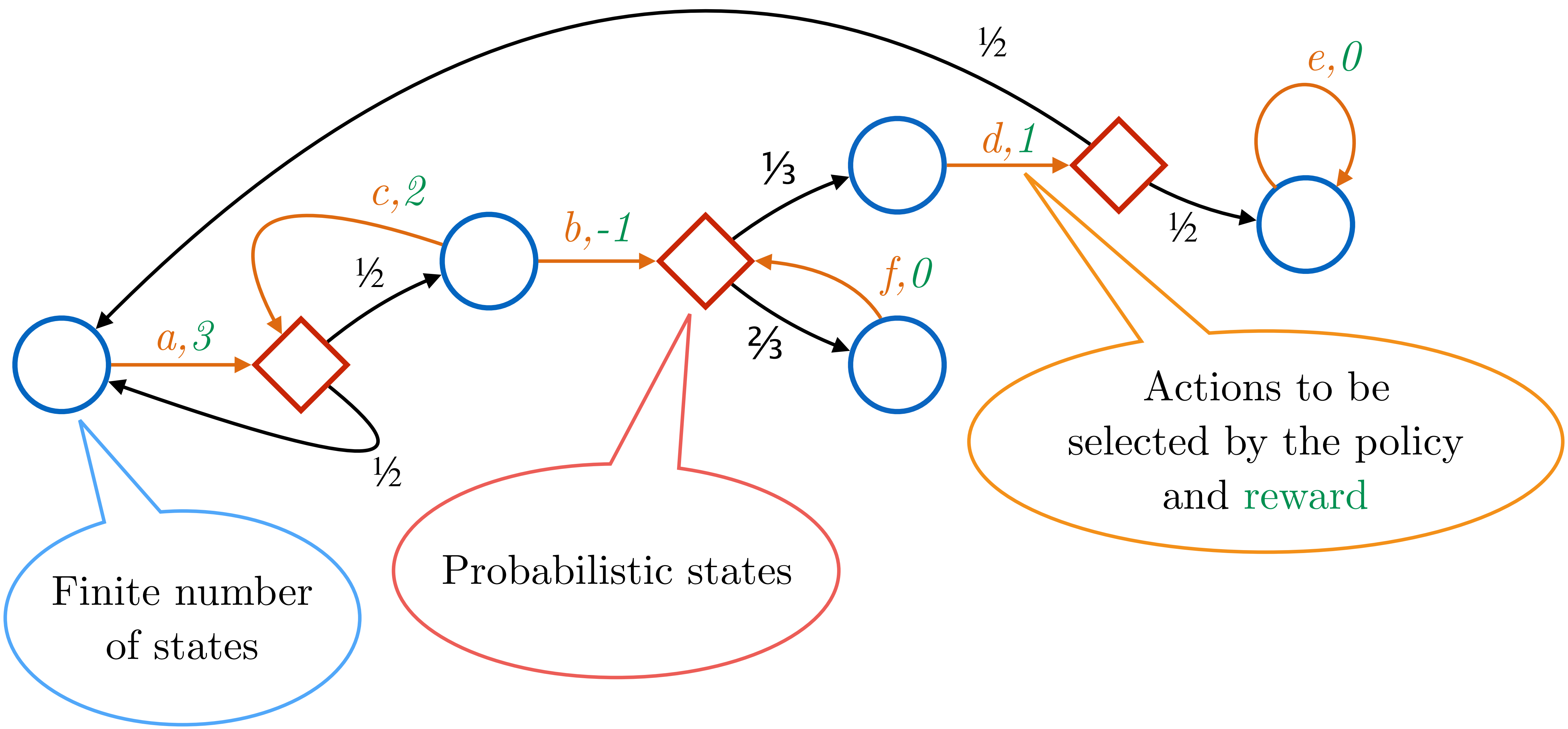
$$\mathcal{M} = (S, \alpha, \delta, r) \qquad 0 < \lambda < 1$$

$$\delta : S \times \alpha \longrightarrow Dist(S)$$

$$r : S \times \alpha \longrightarrow \mathbb{R}$$

Policy $\sigma : (S \cdot \alpha)^{\star} \cdot S \longrightarrow Dist(\alpha)$

MDPs with discounted rewards



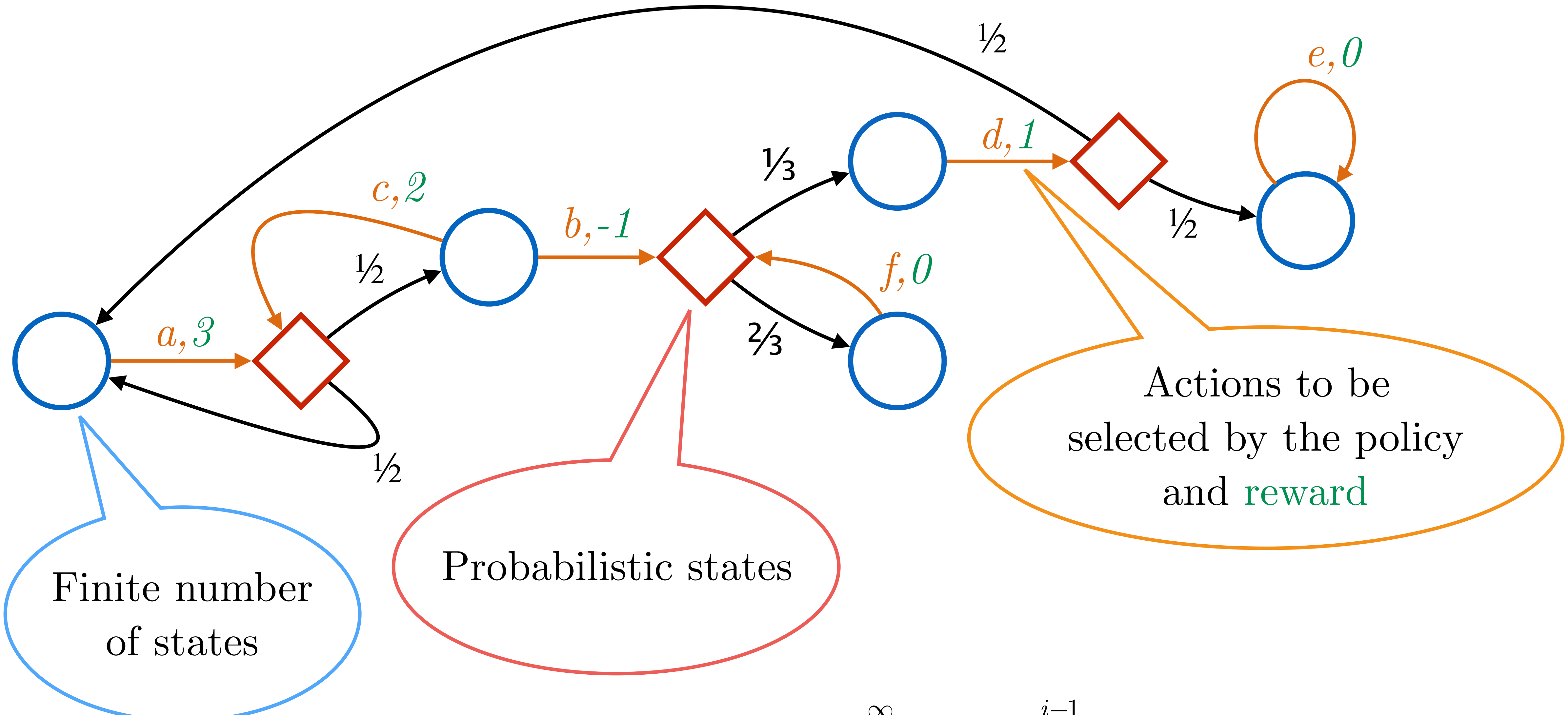
$\mathcal{M} = (S, \alpha, \delta, r) \qquad 0 < \lambda < 1$

$\delta : S \times \alpha \longrightarrow Dist(S)$

$r : S \times \alpha \longrightarrow \mathbb{R}$

Policy $\sigma : (S \cdot \alpha)^{\star} \cdot S \longrightarrow Dist(\alpha)$

MDPs with discounted rewards



$$\mathcal{M} = (S, \alpha, \delta, r) \qquad 0 < \lambda < 1$$

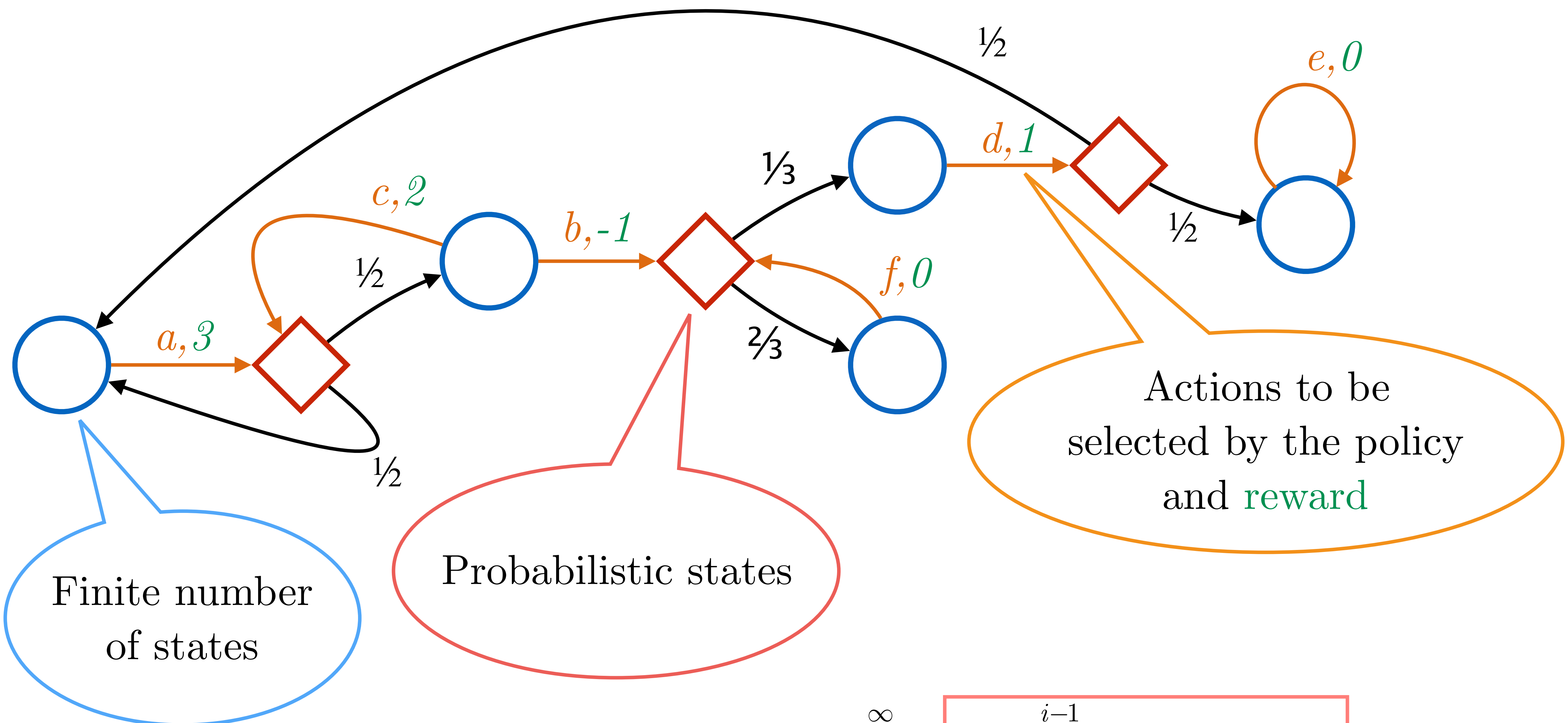
$$\delta : S \times \alpha \to Dist(S)$$

$$r : S \times \alpha \to \mathbb{R}$$

Policy $\sigma : (S \cdot \alpha)^\star \cdot S \to Dist(\alpha)$

$$v^\sigma(s_0) = \sum_{i=0}^{\infty} \lambda^i \sum_{s_1,\ldots,s_i} \prod_{j=0}^{i-1} \delta\big(s_j, \sigma(\ldots s_j)\big)\, r\big(s_i, \sigma(\ldots s_i)\big)$$

MDPs with discounted rewards



$\mathcal{M} = (S, \alpha, \delta, r) \qquad 0 < \lambda < 1$

$\delta : S \times \alpha \to \mathit{Dist}(S)$

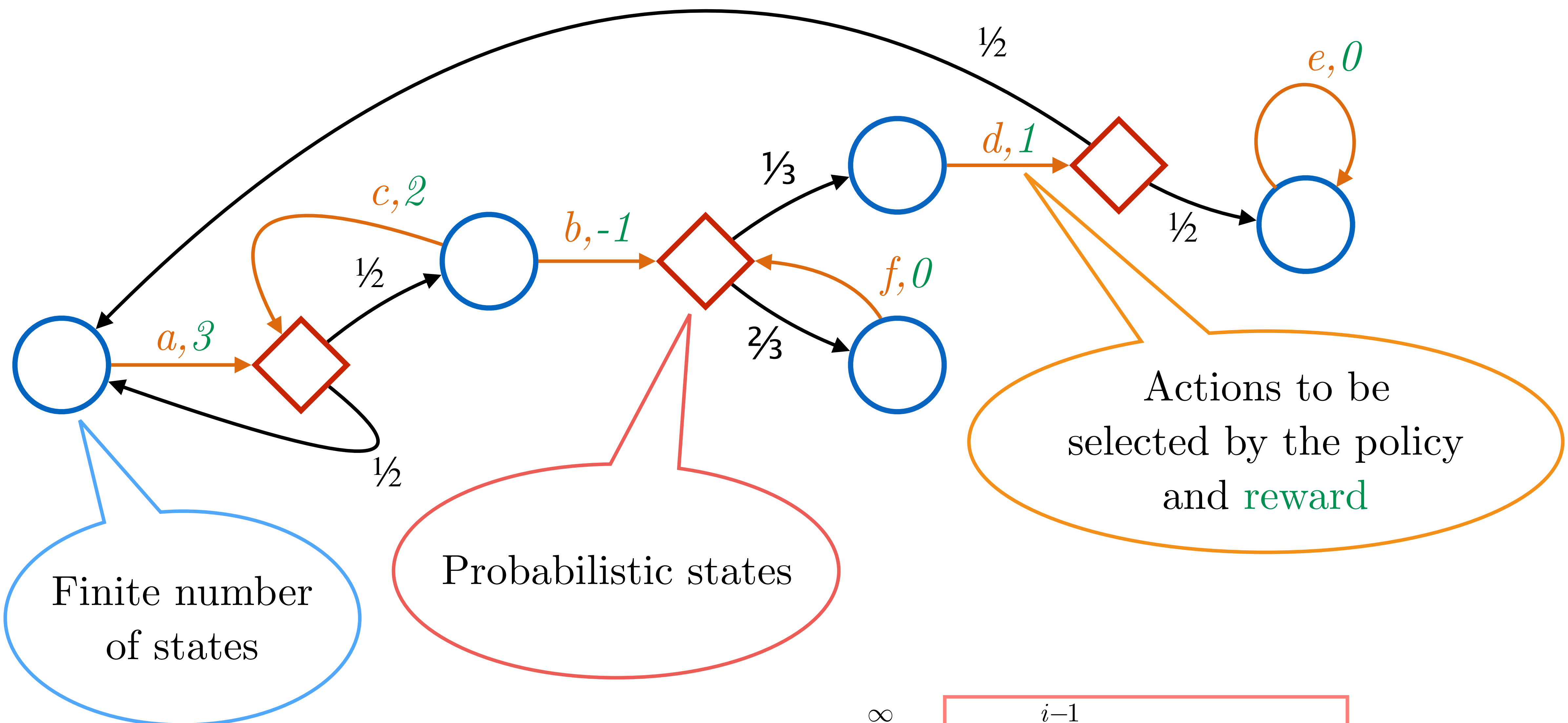
$r : S \times \alpha \to \mathbb{R}$

Policy $\sigma : (S \cdot \alpha)^\star \cdot S \to \mathit{Dist}(\alpha)$

$$v^\sigma(s_0) = \sum_{i=0}^{\infty} \lambda^i \sum_{s_1,\ldots,s_i} \prod_{j=0}^{i-1} \delta\big(s_j, \sigma(\ldots s_j)\big)\, r\big(s_i, \sigma(\ldots s_i)\big)$$

probability to arrive in s_i after i steps

MDPs with discounted rewards



$\mathcal{M} = (S, \alpha, \delta, r) \qquad 0 < \lambda < 1$

$\delta : S \times \alpha \to Dist(S)$

$r : S \times \alpha \to \mathbb{R}$

Policy $\sigma : (S \cdot \alpha)^\star \cdot S \to Dist(\alpha)$

$$v^\sigma(s_0) = \sum_{i=0}^{\infty} \lambda^i \sum_{s_1,\ldots,s_i} \prod_{j=0}^{i-1} \delta\big(s_j, \sigma(\ldots s_j)\big)\, r\big(s_i, \sigma(\ldots s_i)\big)$$

probability to arrive in s_i after i steps

Objective: compute $\sup_\sigma v^\sigma(s_0)$ and good policies

Resolution of MDPs with discounted rewards

$$v^{\sigma}(s_0) = \sum_{i=0}^{\infty} \lambda^i \sum_{s_1,\ldots,s_i} \prod_{j=0}^{i-1} \delta\Big(s_j, \sigma(\ldots s_j)\Big)\, r\Big(s_i, \sigma(\ldots s_i)\Big)$$

Resolution of MDPs with discounted rewards

$$v^\sigma(s_0) = \sum_{i=0}^{\infty} \lambda^i \sum_{s_1,\ldots,s_i} \prod_{j=0}^{i-1} \delta\Big(s_j, \sigma(\ldots s_j)\Big)\, r\Big(s_i, \sigma(\ldots s_i)\Big)$$

memoryless optimal strategies exist: $\quad v^\sigma = \sum_{i=0}^{\infty} (\lambda\Delta^\sigma)^i r^\sigma = (I - \lambda\Delta^\sigma)^{-1} r^\sigma$

Resolution of MDPs with discounted rewards

$$v^\sigma(s_0) = \sum_{i=0}^{\infty} \lambda^i \sum_{s_1,\ldots,s_i} \prod_{j=0}^{i-1} \delta\Big(s_j, \sigma(\ldots s_j)\Big)\, r\Big(s_i, \sigma(\ldots s_i)\Big)$$

memoryless optimal strategies exist: $v^\sigma = \sum_{i=0}^{\infty} (\lambda \Delta^\sigma)^i r^\sigma = (I - \lambda \Delta^\sigma)^{-1} r^\sigma$

$$\boxed{v^\sigma = r^\sigma + \lambda \Delta^\sigma v^\sigma}$$

time horizon 1
with « terminal rewards » λv^σ.

Resolution of MDPs with discounted rewards

$$v^\sigma(s_0) = \sum_{i=0}^{\infty} \lambda^i \sum_{s_1,\dots,s_i} \prod_{j=0}^{i-1} \delta\Big(s_j, \sigma(\dots s_j)\Big)\, r\Big(s_i, \sigma(\dots s_i)\Big)$$

memoryless optimal strategies exist: $v^\sigma = \sum_{i=0}^{\infty} (\lambda \Delta^\sigma)^i r^\sigma = (I - \lambda\Delta^\sigma)^{-1} r^\sigma$

$$\boxed{v^\sigma = r^\sigma + \lambda \Delta^\sigma v^\sigma}$$

time horizon 1
with « terminal rewards » λv^σ.

Function $L : \mathbb{R}^S \to \mathbb{R}^S$ defined by

$$L(v)_s = \max_{a \in \alpha} r(s,a) + \lambda \sum_{s' \in S} \delta(s,a)(s') v_{s'}$$

Resolution of MDPs with discounted rewards

$$v^\sigma(s_0) = \sum_{i=0}^{\infty} \lambda^i \sum_{s_1,\ldots,s_i} \prod_{j=0}^{i-1} \delta\big(s_j, \sigma(\ldots s_j)\big)\, r\big(s_i, \sigma(\ldots s_i)\big)$$

memoryless optimal strategies exist: $v^\sigma = \sum_{i=0}^{\infty} (\lambda \Delta^\sigma)^i r^\sigma = (I - \lambda \Delta^\sigma)^{-1} r^\sigma$

$$\boxed{v^\sigma = r^\sigma + \lambda \Delta^\sigma v^\sigma}$$

time horizon 1
with « terminal rewards » λv^σ.

Function $L : \mathbb{R}^S \to \mathbb{R}^S$ defined by

$$L(v)_s = \max_{a \in \alpha} r(s,a) + \lambda \sum_{s' \in S} \delta(s,a)(s') v_{s'}$$

verifies $\| L(v) - L(v') \|_\infty \leq \lambda \| v - v' \|_\infty$

Resolution of MDPs with discounted rewards

$$v^\sigma(s_0) = \sum_{i=0}^{\infty} \lambda^i \sum_{s_1,\ldots,s_i} \prod_{j=0}^{i-1} \delta\big(s_j, \sigma(\ldots s_j)\big)\, r\big(s_i, \sigma(\ldots s_i)\big)$$

memoryless optimal strategies exist: $v^\sigma = \sum_{i=0}^{\infty} (\lambda \Delta^\sigma)^i r^\sigma = (I - \lambda \Delta^\sigma)^{-1} r^\sigma$

$$\boxed{v^\sigma = r^\sigma + \lambda \Delta^\sigma v^\sigma}$$

time horizon 1
with « terminal rewards » λv^σ.

Function $L : \mathbb{R}^S \to \mathbb{R}^S$ defined by

$$L(v)_s = \max_{a \in \alpha} r(s,a) + \lambda \sum_{s' \in S} \delta(s,a)(s') v_{s'}$$

verifies $\| L(v) - L(v') \|_\infty \leq \lambda \| v - v' \|_\infty$

$v^\star = \sup_\sigma v^\sigma$ is the unique fixed point of L

$\lim_{n \to \infty} L^n(v_0) = v^\star$ $\quad\quad \| v^\star - L^n(v_0) \|_\infty \leq \frac{\lambda^n}{1 - \lambda} \| L(v_0) - v_0 \|_\infty$

Resolution of MDPs with discounted rewards

$$v^\sigma(s_0) = \sum_{i=0}^{\infty} \lambda^i \sum_{s_1,\ldots,s_i} \prod_{j=0}^{i-1} \delta\big(s_j, \sigma(\ldots s_j)\big)\, r\big(s_i, \sigma(\ldots s_i)\big)$$

memoryless optimal strategies exist: $v^\sigma = \sum_{i=0}^{\infty} (\lambda \Delta^\sigma)^i r^\sigma = (I - \lambda \Delta^\sigma)^{-1} r^\sigma$

$$\boxed{v^\sigma = r^\sigma + \lambda \Delta^\sigma v^\sigma}$$

time horizon 1
with « terminal rewards » λv^σ.

Function $L : \mathbb{R}^S \to \mathbb{R}^S$ defined by

$$L(v)_s = \max_{a \in \alpha} r(s,a) + \lambda \sum_{s' \in S} \delta(s,a)(s') v_{s'}$$

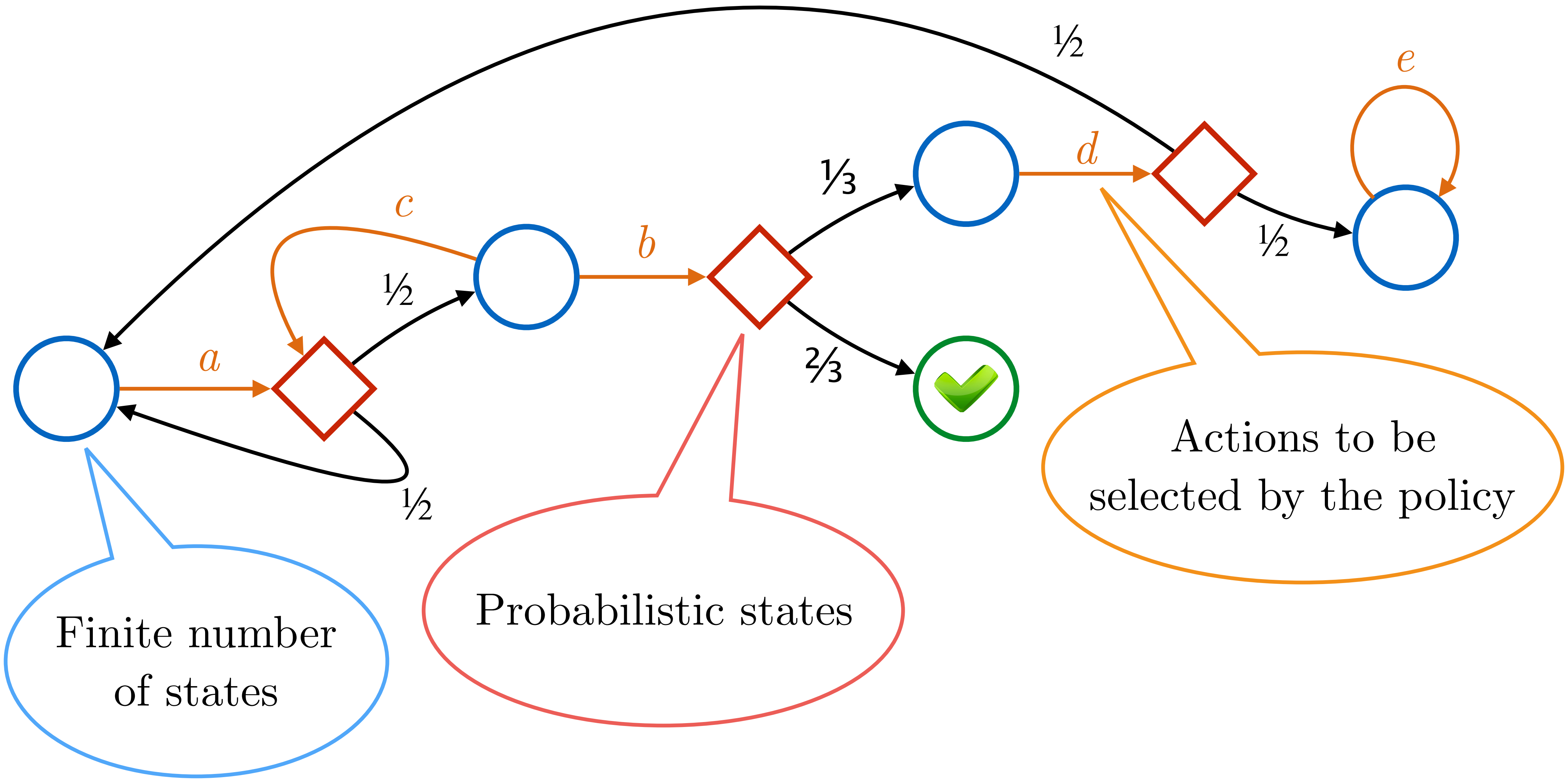
verifies $\| L(v) - L(v') \|_\infty \leq \lambda \| v - v' \|_\infty$

speed of convergence +
stopping criterion for algorithm

$v^\star = \sup_\sigma v^\sigma$ is the unique fixed point of L

$$\lim_{n\to\infty} L^n(v_0) = v^\star \qquad \| v^\star - L^n(v_0) \|_\infty \leq \frac{\lambda^n}{1-\lambda} \| L(v_0) - v_0 \|_\infty$$

MDPs with reachability objectives

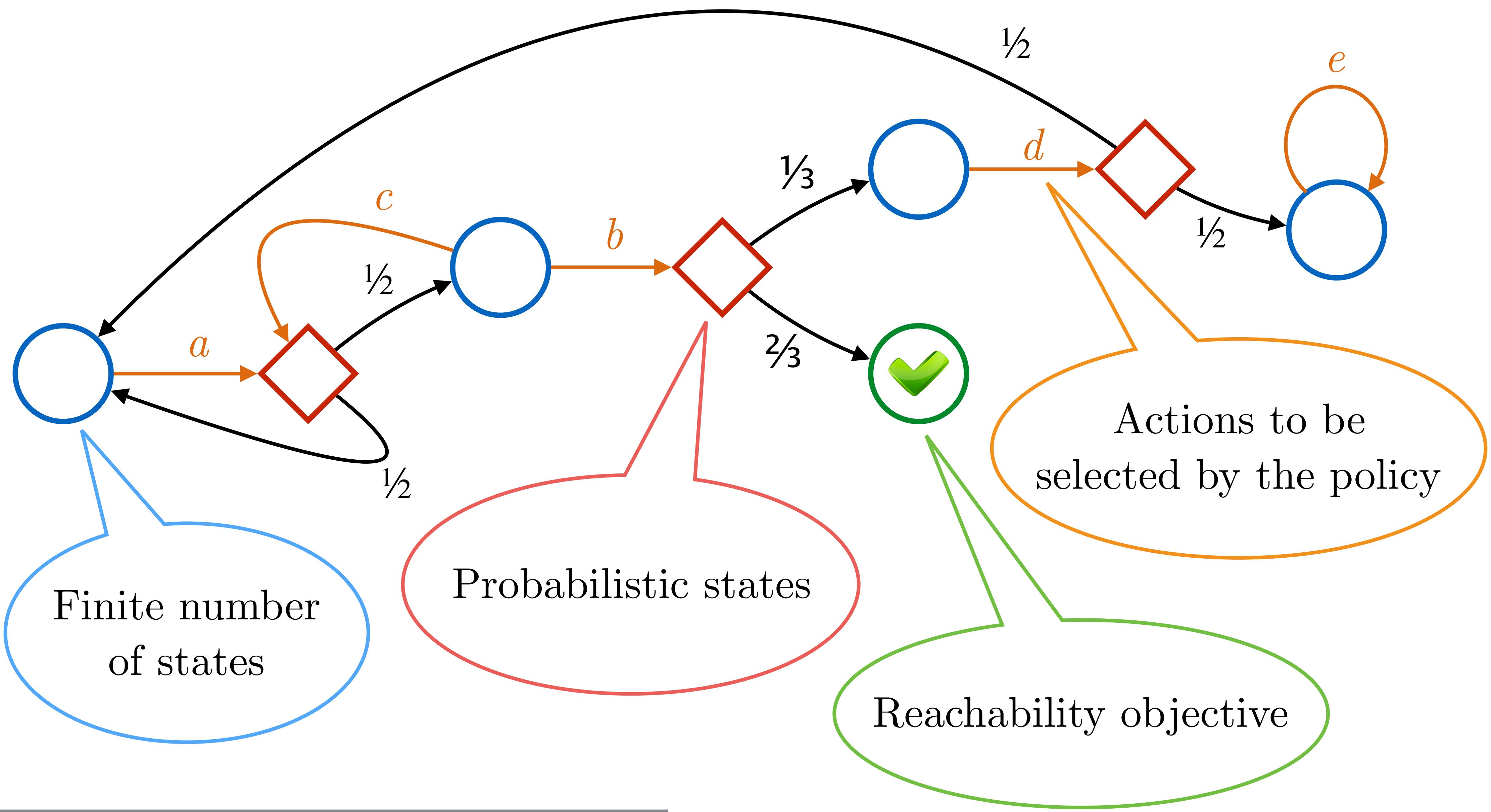


$$\mathcal{M} = (S, \alpha, \delta)$$

$$\delta : S \times \alpha \longrightarrow Dist(S)$$

Policy $\sigma : (S \cdot \alpha)^{\star} \cdot S \longrightarrow Dist(\alpha)$

MDPs with reachability objectives

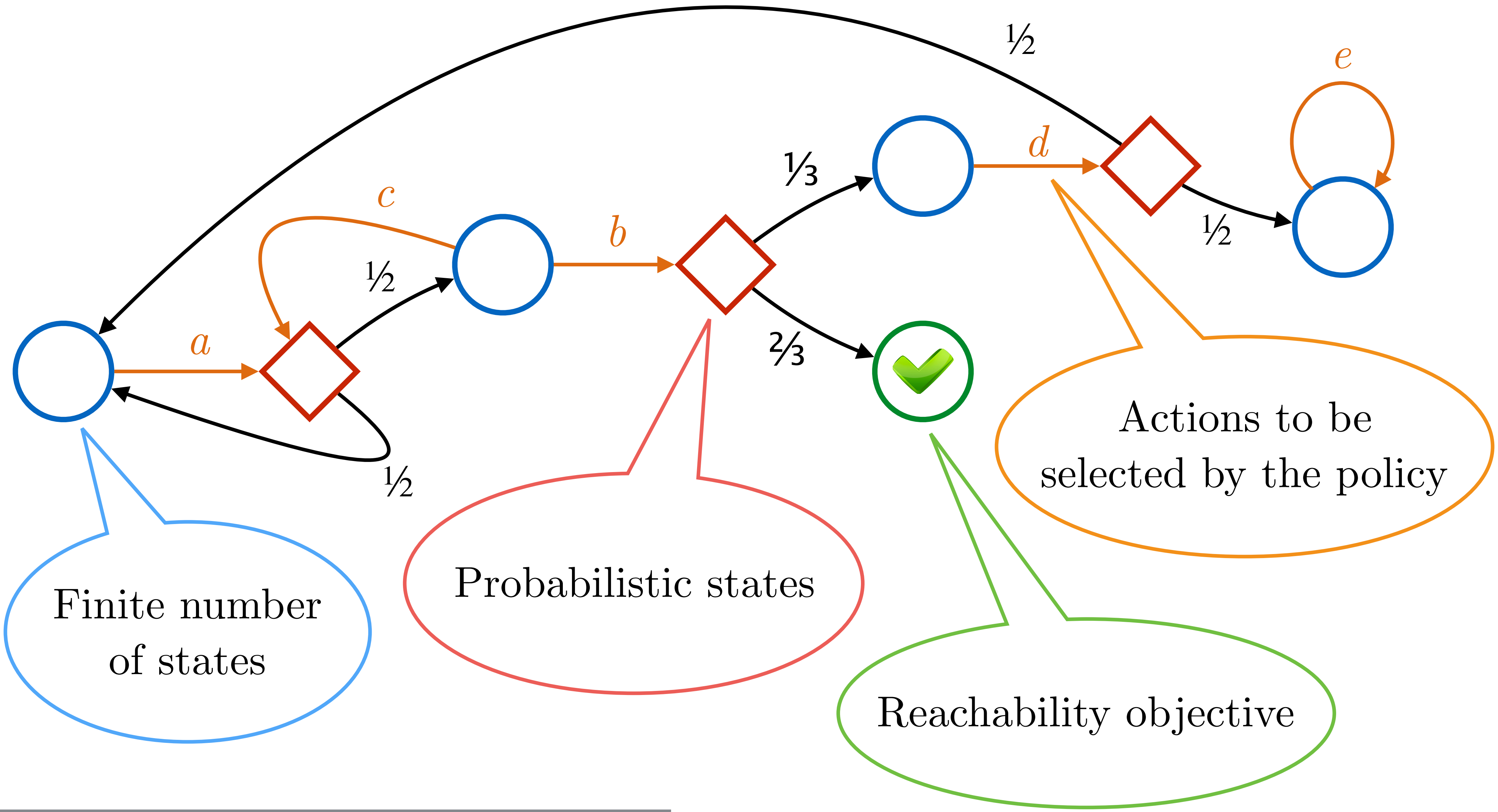


$\mathcal{M} = (S, \alpha, \delta)$

$\delta : S \times \alpha \longrightarrow Dist(S)$

Policy $\sigma : (S \cdot \alpha)^{\star} \cdot S \longrightarrow Dist(\alpha)$

MDPs with reachability objectives



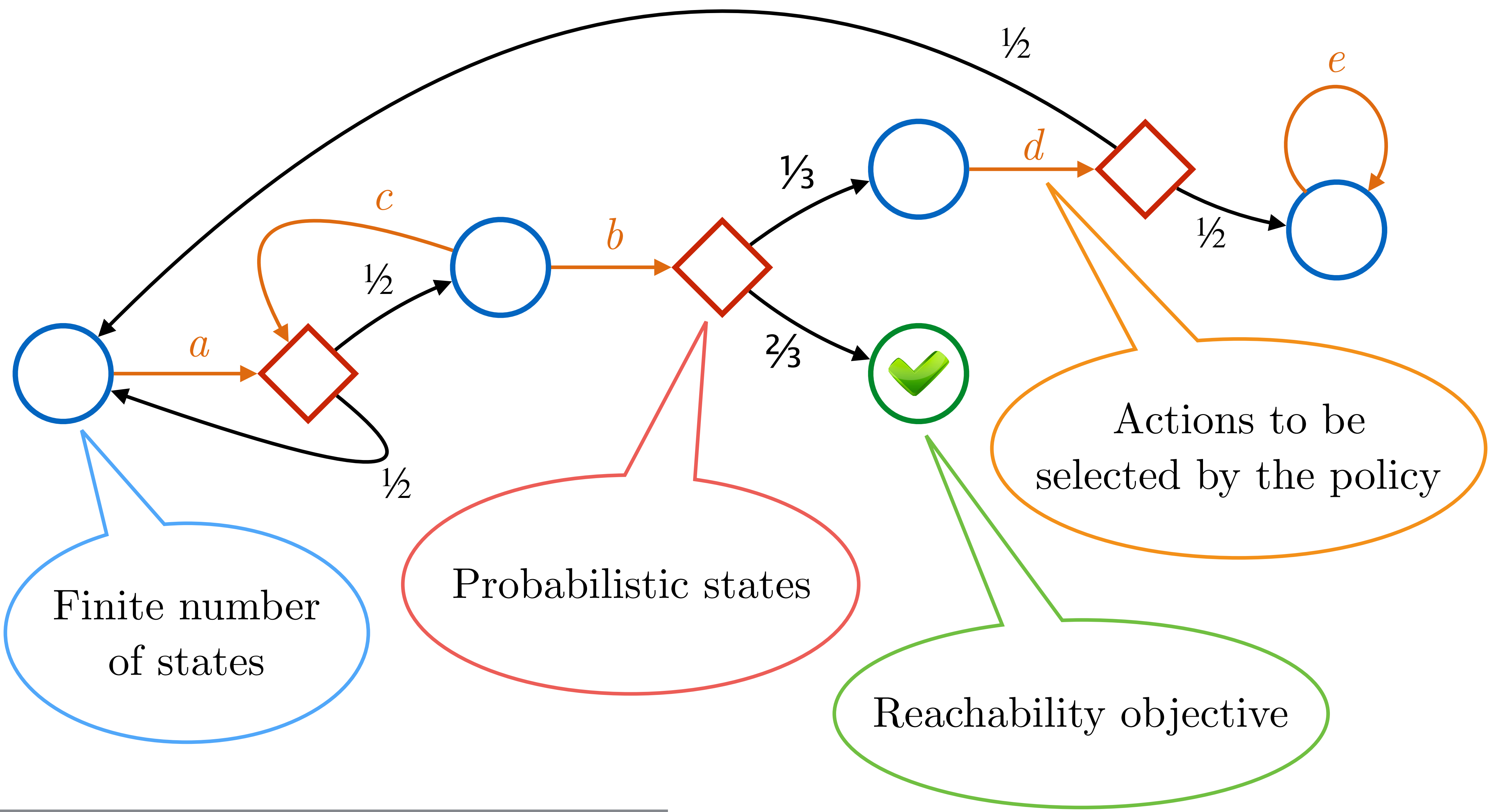
$\mathcal{M} = (S, \alpha, \delta)$

$\delta : S \times \alpha \to Dist(S)$

Policy $\sigma : (S \cdot \alpha)^{\star} \cdot S \to Dist(\alpha)$

Probability to reach: $\Pr_s^{\sigma}(\mathsf{F}$ ✔ $)$

MDPs with reachability objectives



$\mathcal{M} = (S, \alpha, \delta)$

$\delta : S \times \alpha \rightarrow Dist(S)$

Policy $\sigma : (S \cdot \alpha)^{\star} \cdot S \rightarrow Dist(\alpha)$

Probability to reach: $\Pr_s^{\sigma}(\mathsf{F}\ ✓)$

Maximal probability to reach: $\Pr_s^{\max}(\mathsf{F}\ ✓) = \sup_{\sigma} \Pr_s^{\sigma}(\mathsf{F}\ ✓)$

Optimal reachability probabilities of MDPs

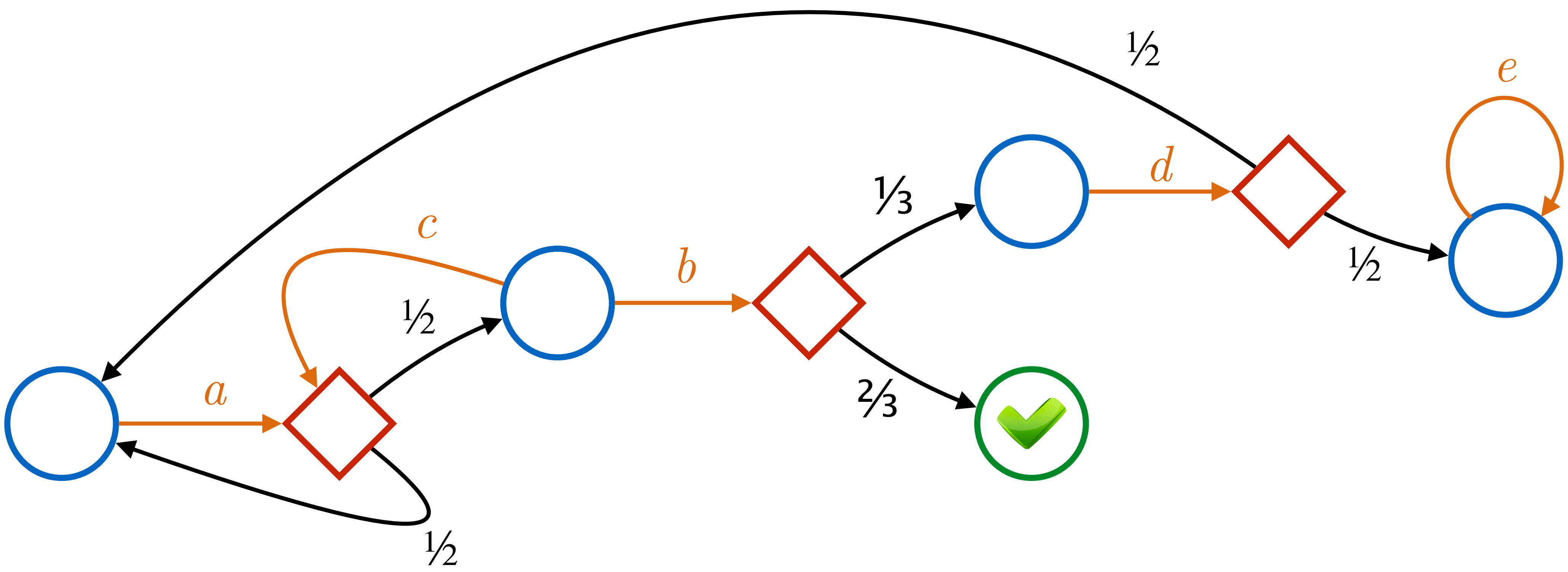
- How?
  - *Linear programming*
  - *Policy iteration*
  - *Value iteration*: numerical scheme that scales well and works in practice

Optimal reachability probabilities of MDPs

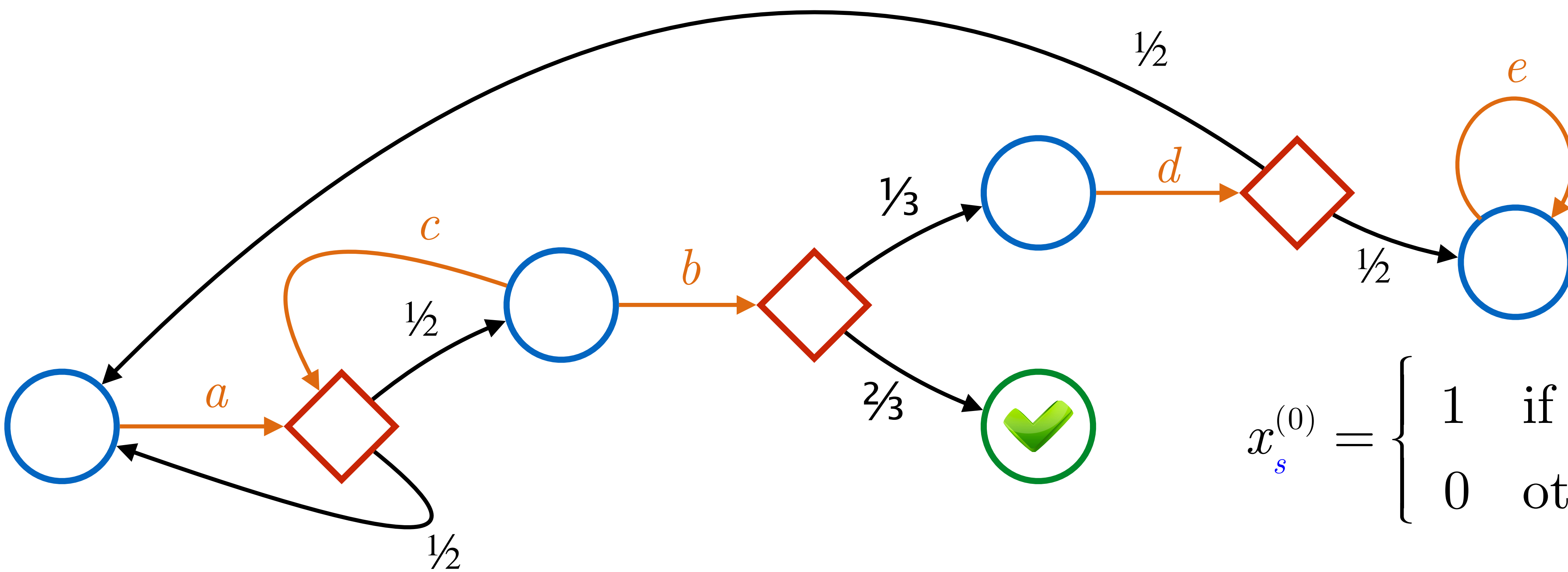
- How?
  - ✦ *Linear programming*
  - ✦ *Policy iteration*
  - ✦ *Value iteration*: numerical scheme that scales well and works in practice

used in the numerical PRISM model checker
[Kwiatkowska, Norman, Parker, 2011]

Value iteration



Value iteration

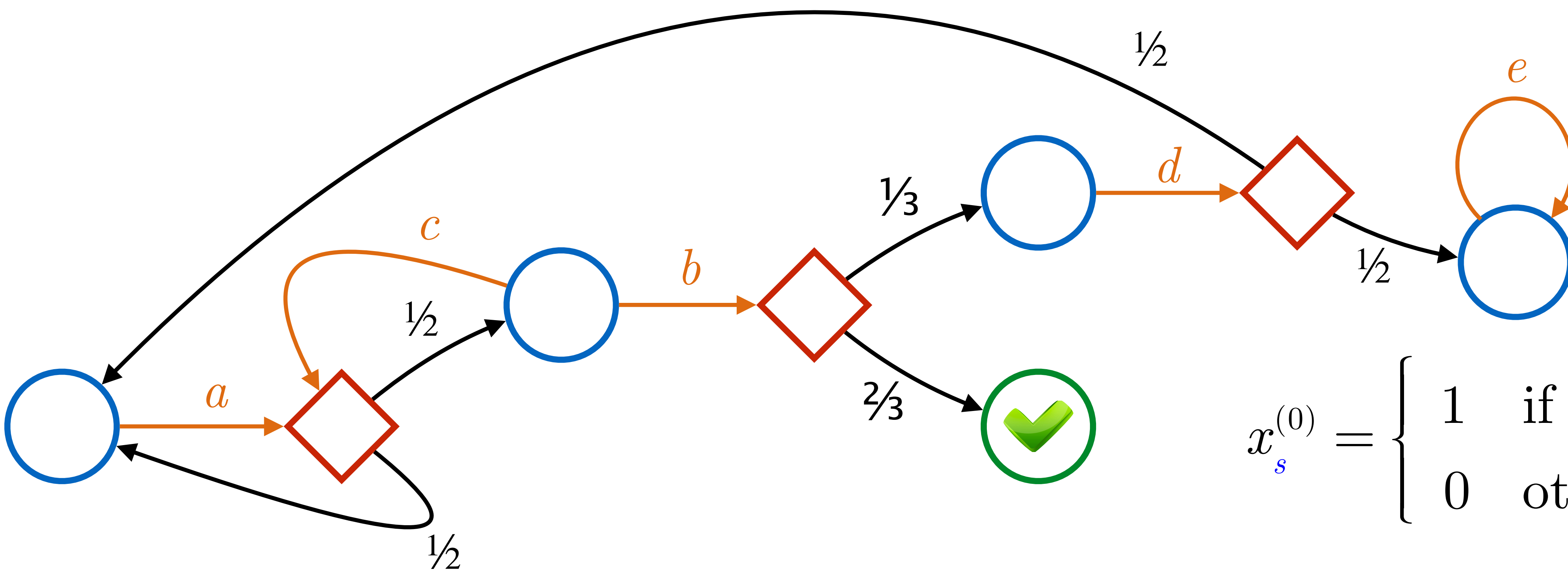


$$x_s^{(0)} = \begin{cases} 1 & \text{if } s = ✔ \\ 0 & \text{otherwise} \end{cases}$$

$$x_s^{(n+1)} = \max_{a \in \alpha} \sum_{s' \in S} \delta(s, a)(s') \times x_{s'}^{(n)}$$

Value iteration

0	0	0	0

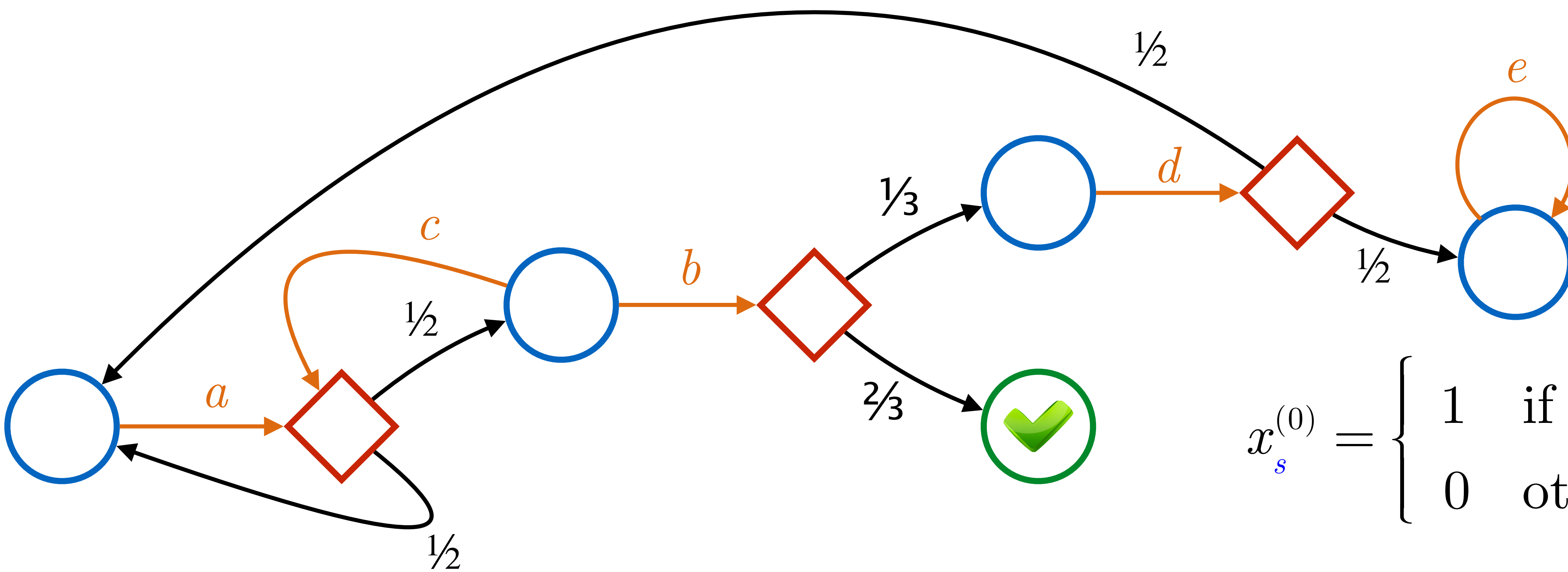


$$x_s^{(0)} = \begin{cases} 1 & \text{if } s = \checkmark \\ 0 & \text{otherwise} \end{cases}$$

$$x_s^{(n+1)} = \max_{a \in \alpha} \sum_{s' \in S} \delta(s, a)(s') \times x_{s'}^{(n)}$$

Value iteration

0	0	0	0
0	$2/3\ (b)$	0	0

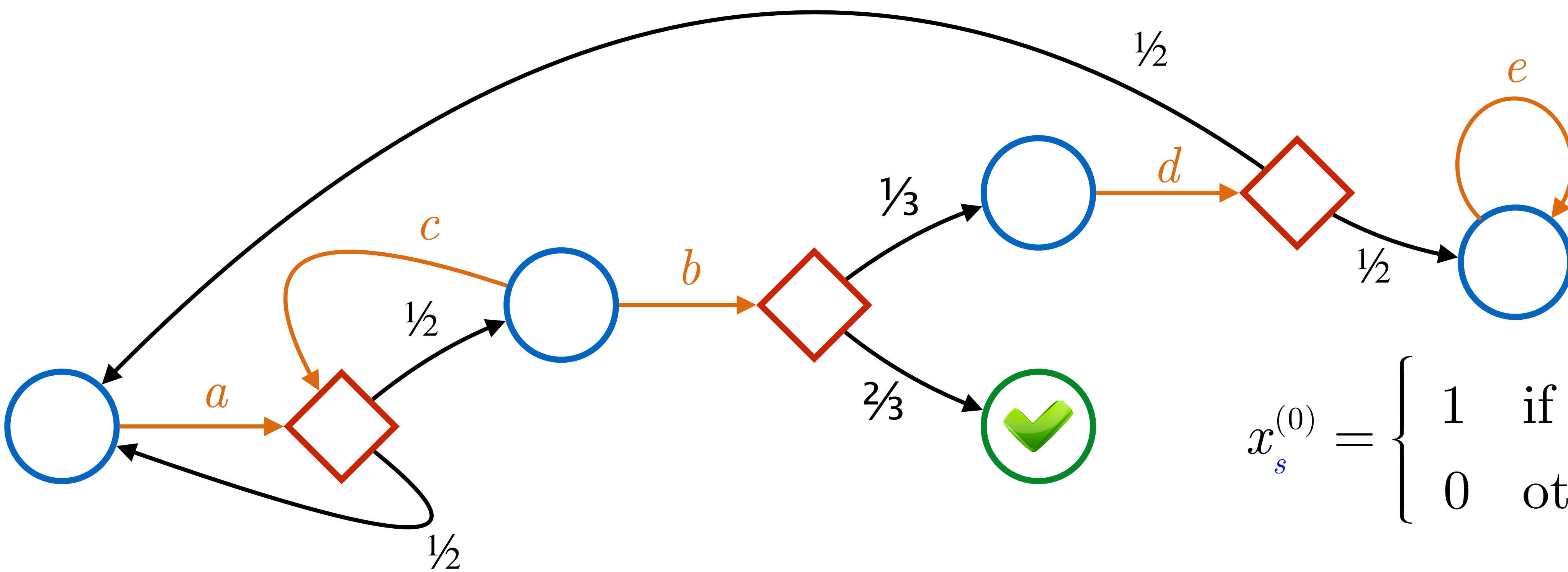


$$x_s^{(0)} = \begin{cases} 1 & \text{if } s = ✔ \\ 0 & \text{otherwise} \end{cases}$$

$$x_s^{(n+1)} = \max_{a \in \alpha} \sum_{s' \in S} \delta(s,a)(s') \times x_{s'}^{(n)}$$

Value iteration

0	0	0	0
0	$2/3\ (b)$	0	0
$1/3$	$2/3\ (b)$	0	0

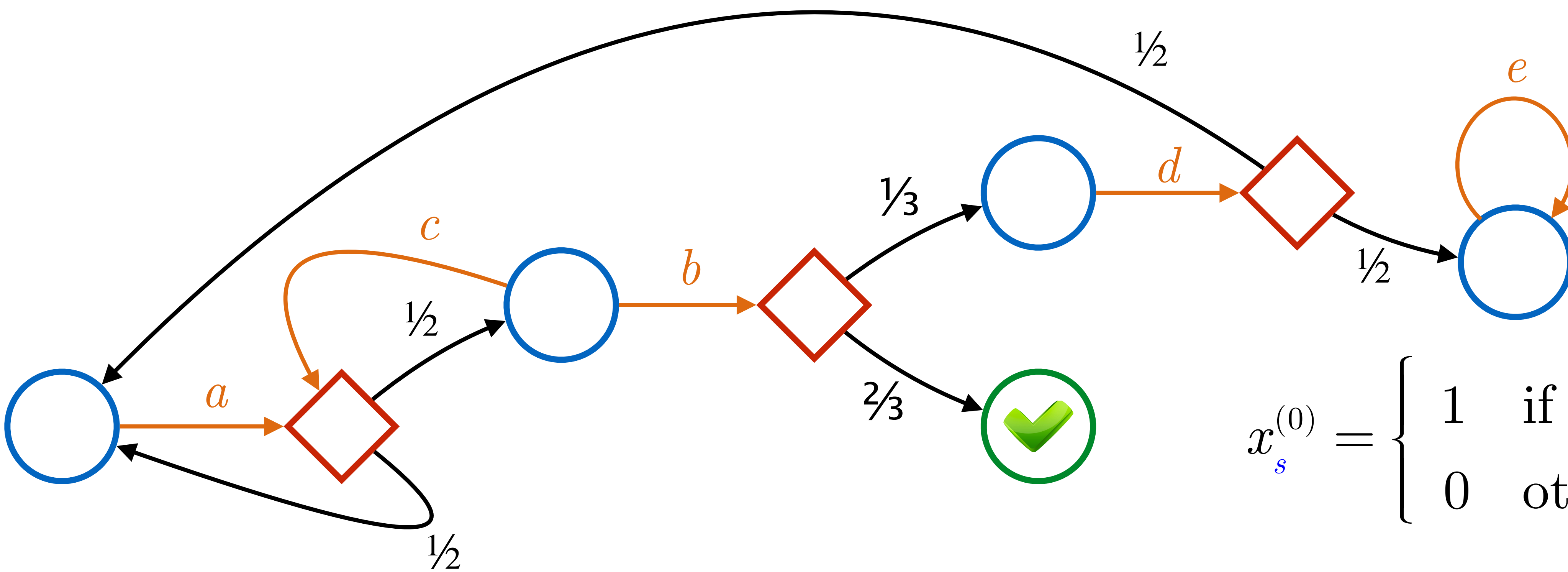


$$x_s^{(0)} = \begin{cases} 1 & \text{if } s = ✔ \\ 0 & \text{otherwise} \end{cases}$$

$$x_s^{(n+1)} = \max_{a \in \alpha} \sum_{s' \in S} \delta(s,a)(s') \times x_{s'}^{(n)}$$

Value iteration

0	0	0	0
0	$2/3\ (b)$	0	0
$1/3$	$2/3\ (b)$	0	0
$1/2$	$2/3\ (b)$	$1/6$	0

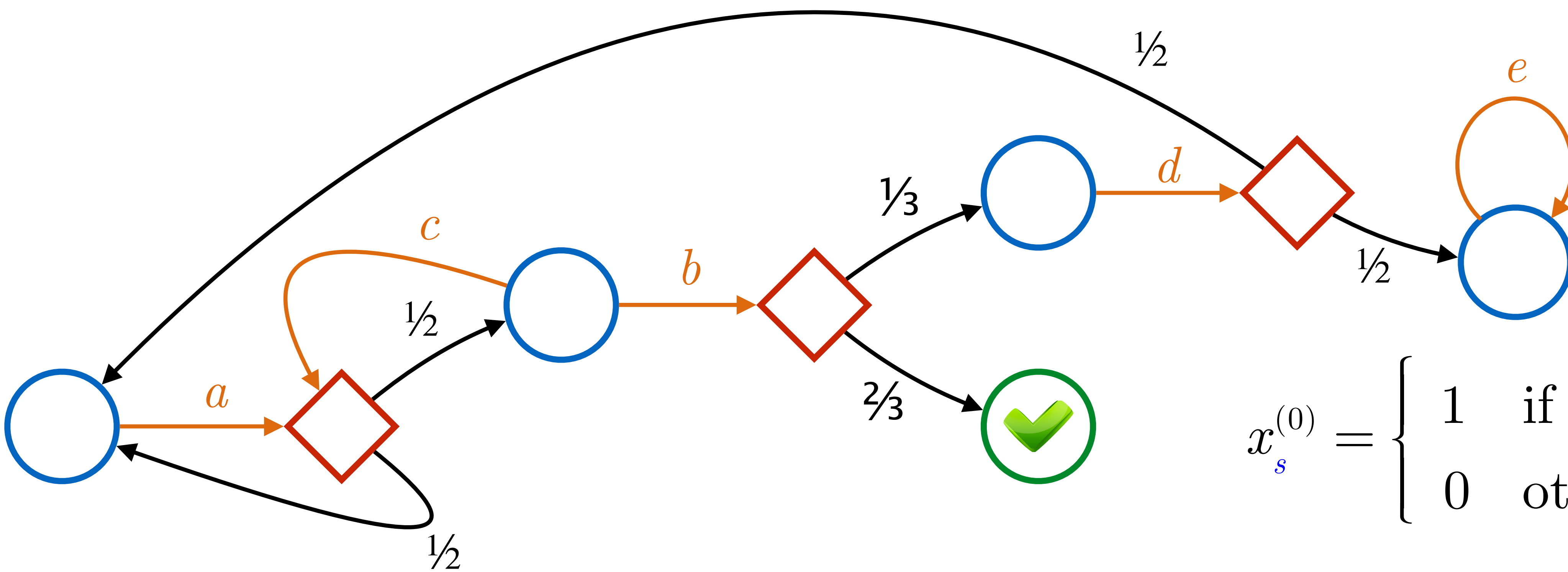


$$x_s^{(0)} = \begin{cases} 1 & \text{if } s = ✔ \\ 0 & \text{otherwise} \end{cases}$$

$$x_s^{(n+1)} = \max_{a \in \alpha} \sum_{s' \in S} \delta(s,a)(s') \times x_{s'}^{(n)}$$

Value iteration

0	0	0	0
0	2/3 (b)	0	0
1/3	2/3 (b)	0	0
1/2	2/3 (b)	1/6	0
7/12	13/18 (b)	1/4	0

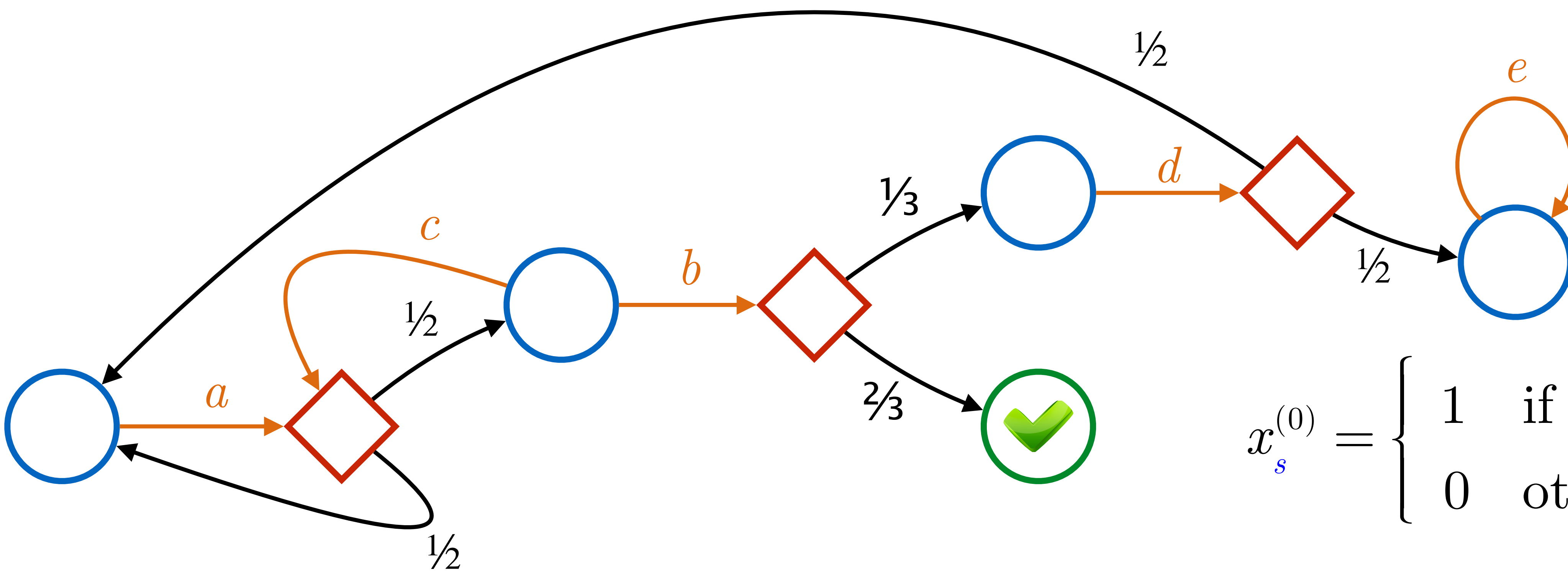


$$x_s^{(0)} = \begin{cases} 1 & \text{if } s = ✔ \\ 0 & \text{otherwise} \end{cases}$$

$$x_s^{(n+1)} = \max_{a \in \alpha} \sum_{s' \in S} \delta(s,a)(s') \times x_{s'}^{(n)}$$

Value iteration

0	0	0	0
0	2/3 (*b*)	0	0
1/3	2/3 (*b*)	0	0
1/2	2/3 (*b*)	1/6	0
7/12	13/18 (*b*)	1/4	0
...	...	...	...

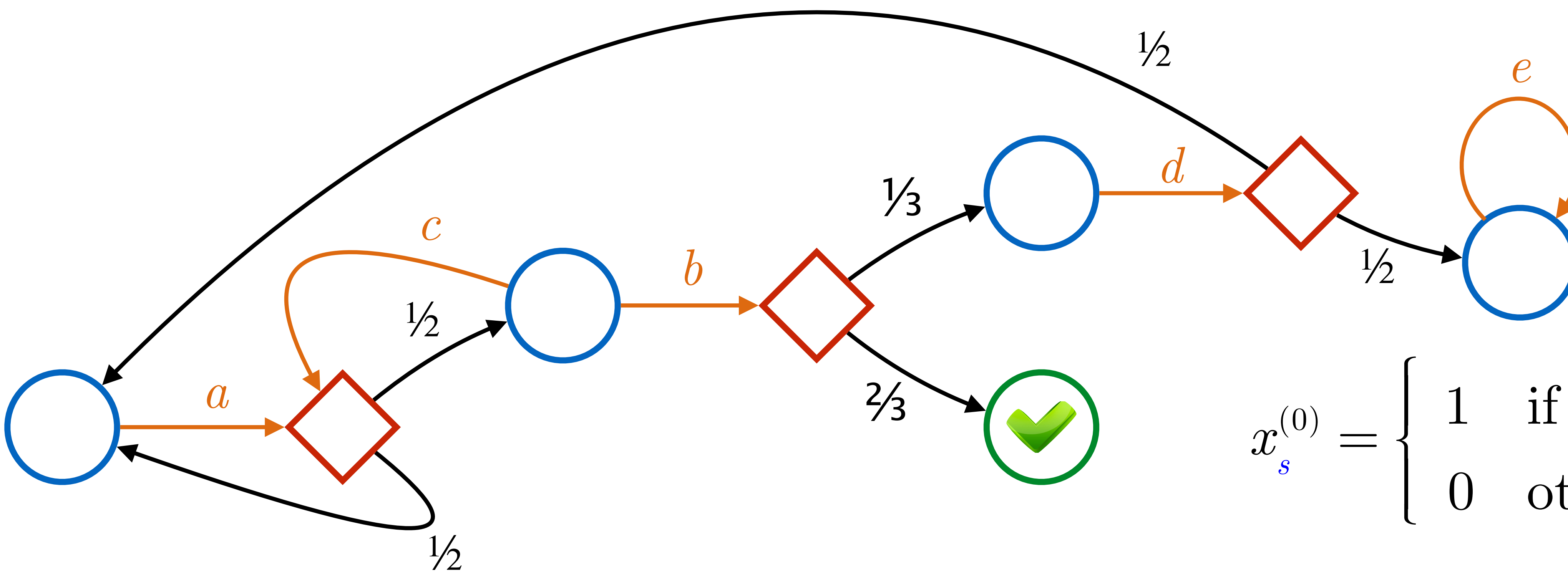


$$x_s^{(0)} = \begin{cases} 1 & \text{if } s = \checkmark \\ 0 & \text{otherwise} \end{cases}$$

$$x_s^{(n+1)} = \max_{a \in \alpha} \sum_{s' \in S} \delta(s,a)(s') \times x_{s'}^{(n)}$$

Value iteration

0	0	0	0
0	$2/3\ (b)$	0	0
$1/3$	$2/3\ (b)$	0	0
$1/2$	$2/3\ (b)$	$1/6$	0
$7/12$	$13/18\ (b)$	$1/4$	0
...	...	...	...
0.7969	0.7988 (b)	0.3977	0

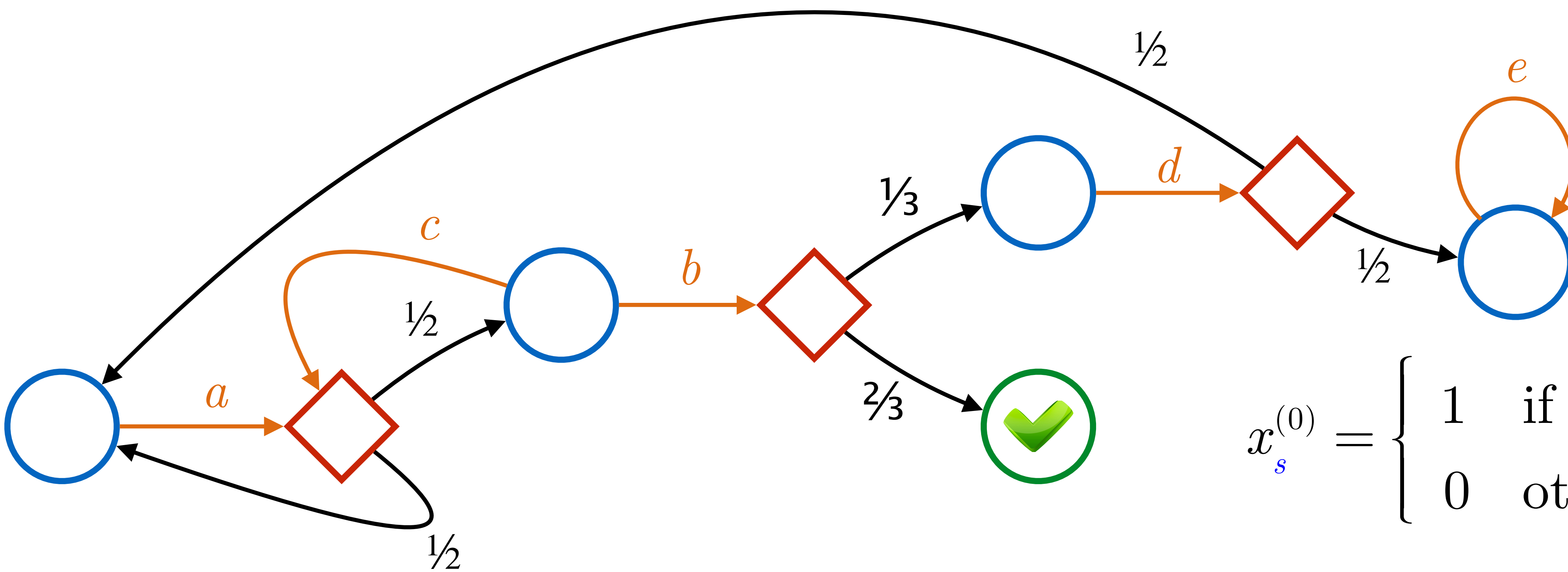


$$x_s^{(0)} = \begin{cases} 1 & \text{if } s = \checkmark \\ 0 & \text{otherwise} \end{cases}$$

$$x_s^{(n+1)} = \max_{a \in \alpha} \sum_{s' \in S} \delta(s, a)(s') \times x_{s'}^{(n)}$$

Value iteration

0	0	0	0
0	$2/3\ (b)$	0	0
$1/3$	$2/3\ (b)$	0	0
$1/2$	$2/3\ (b)$	$1/6$	0
$7/12$	$13/18\ (b)$	$1/4$	0
...	...	...	...
0.7969	$0.7988\ (b)$	0.3977	0
0.7978	$0.7992\ (b)$	0.3984	0



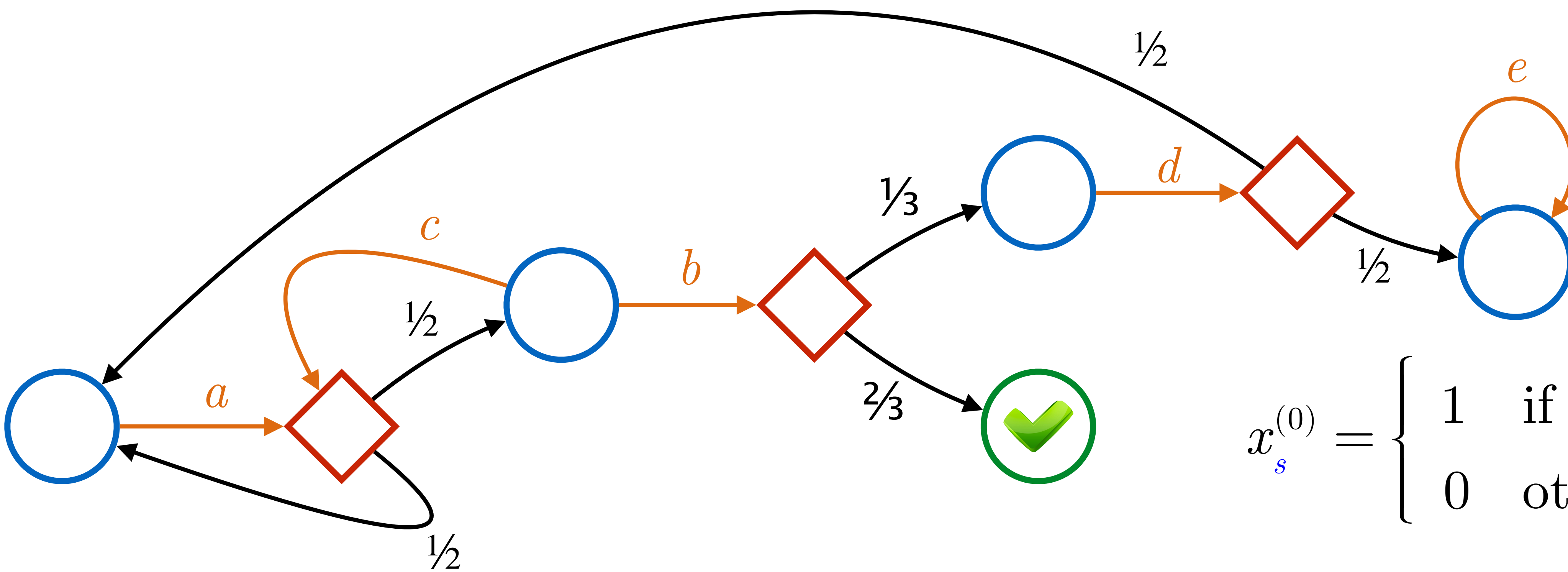
$$x_s^{(0)} = \begin{cases} 1 & \text{if } s = ✔ \\ 0 & \text{otherwise} \end{cases}$$

$$x_s^{(n+1)} = \max_{a \in \alpha} \sum_{s' \in S} \delta(s,a)(s') \times x_{s'}^{(n)}$$

Value iteration

0	0	0	0
0	$2/3$ (b)	0	0
$1/3$	$2/3$ (b)	0	0
$1/2$	$2/3$ (b)	$1/6$	0
$7/12$	$13/18$ (b)	$1/4$	0
...	...	...	...
0.7969	0.7988 (b)	0.3977	0
0.7978	0.7992 (b)	0.3984	0

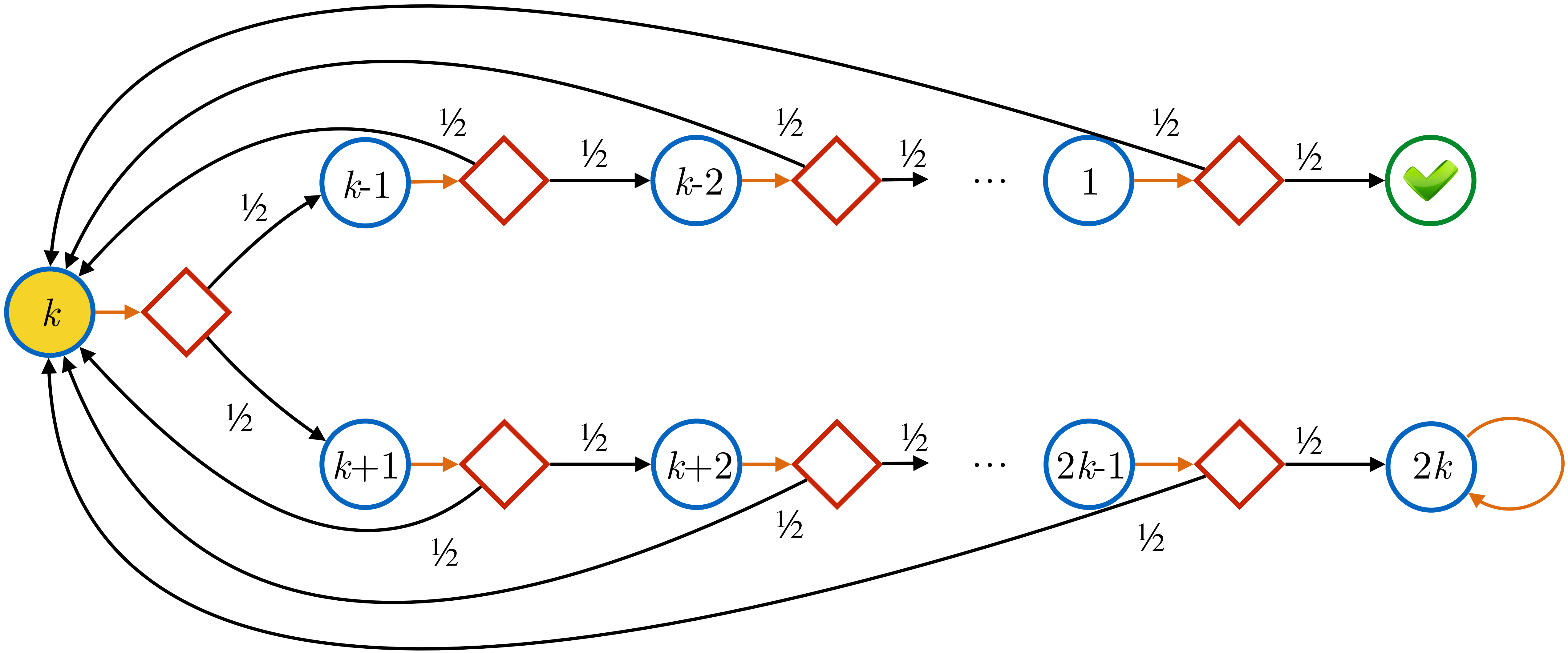
≤ 0.001



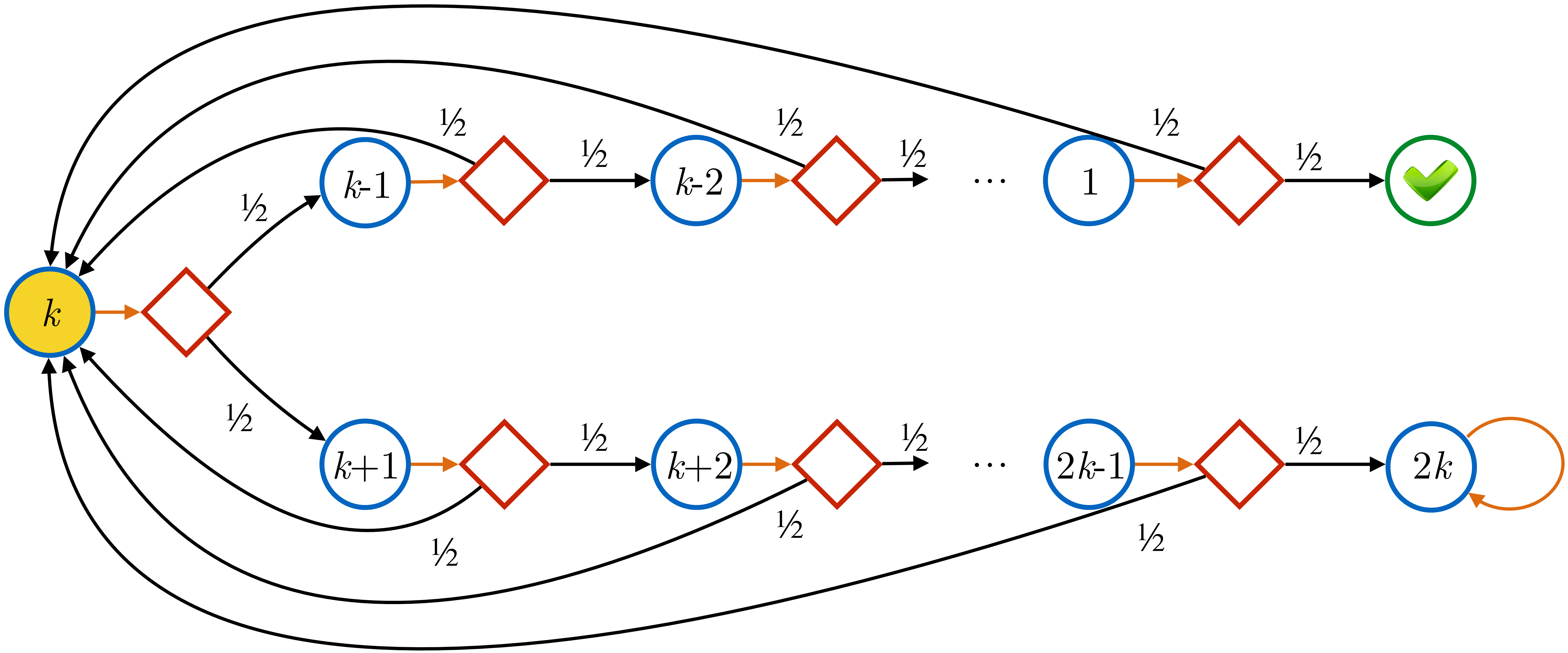
$$x_s^{(0)} = \begin{cases} 1 & \text{if } s = ✔ \\ 0 & \text{otherwise} \end{cases}$$

$$x_s^{(n+1)} = \max_{a \in \alpha} \sum_{s' \in S} \delta(s,a)(s') \times x_{s'}^{(n)}$$

Value iteration: which guarantees?

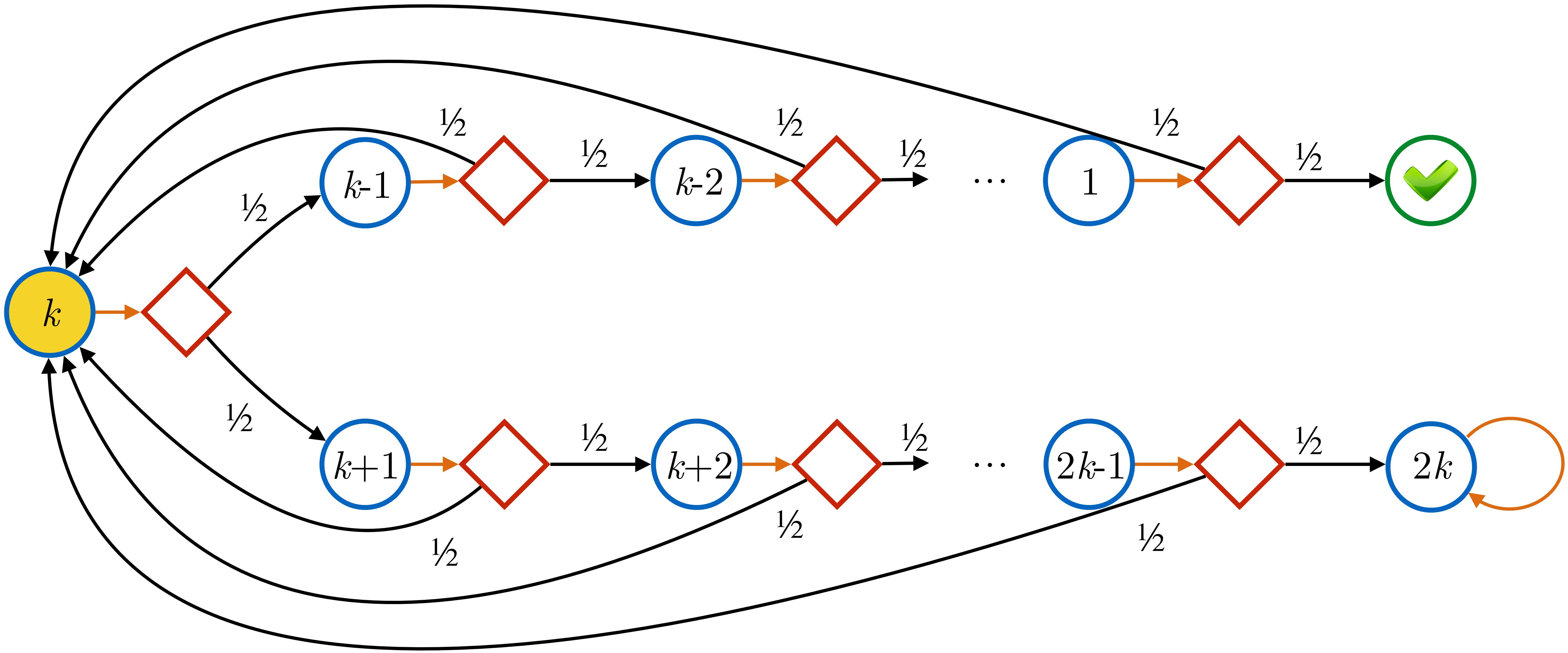


Value iteration: which guarantees?



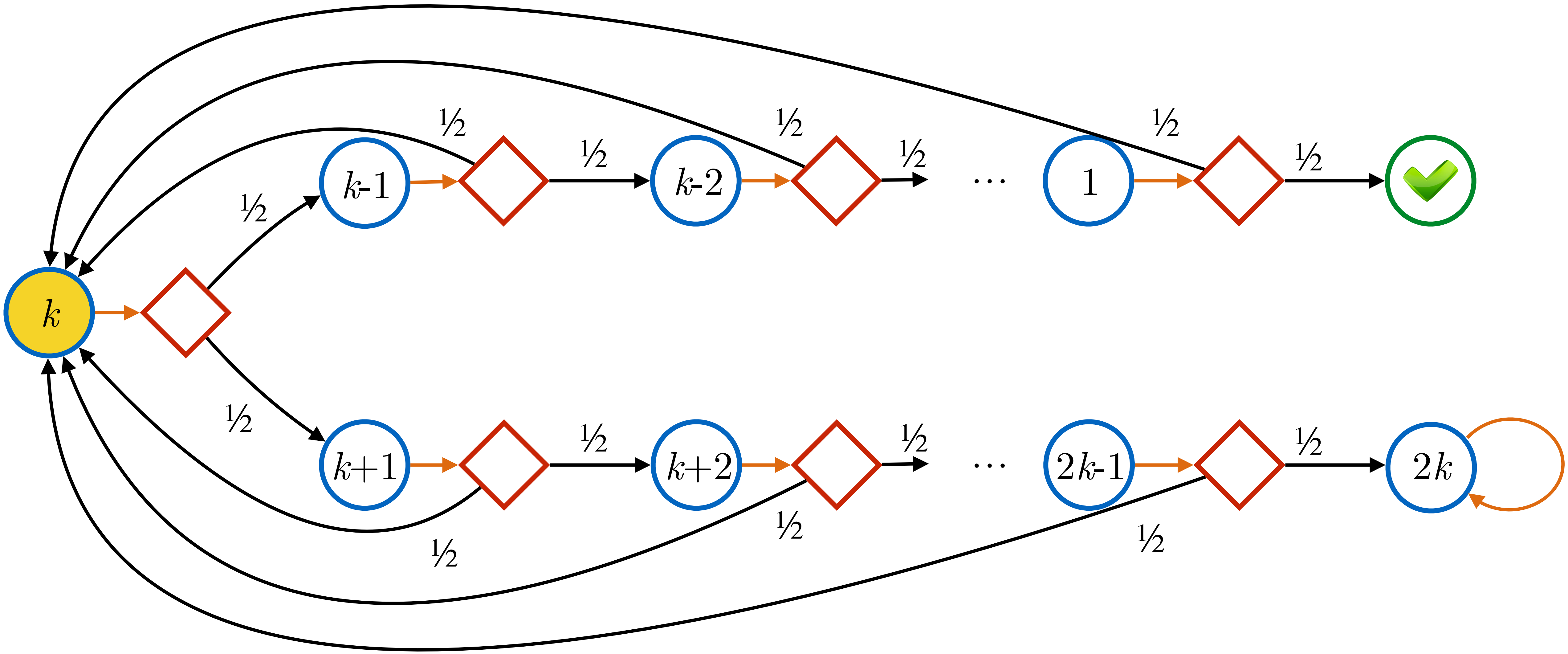
State	0	1	2	3	…	k-1	k	k+1	…	2k

Value iteration: which guarantees?



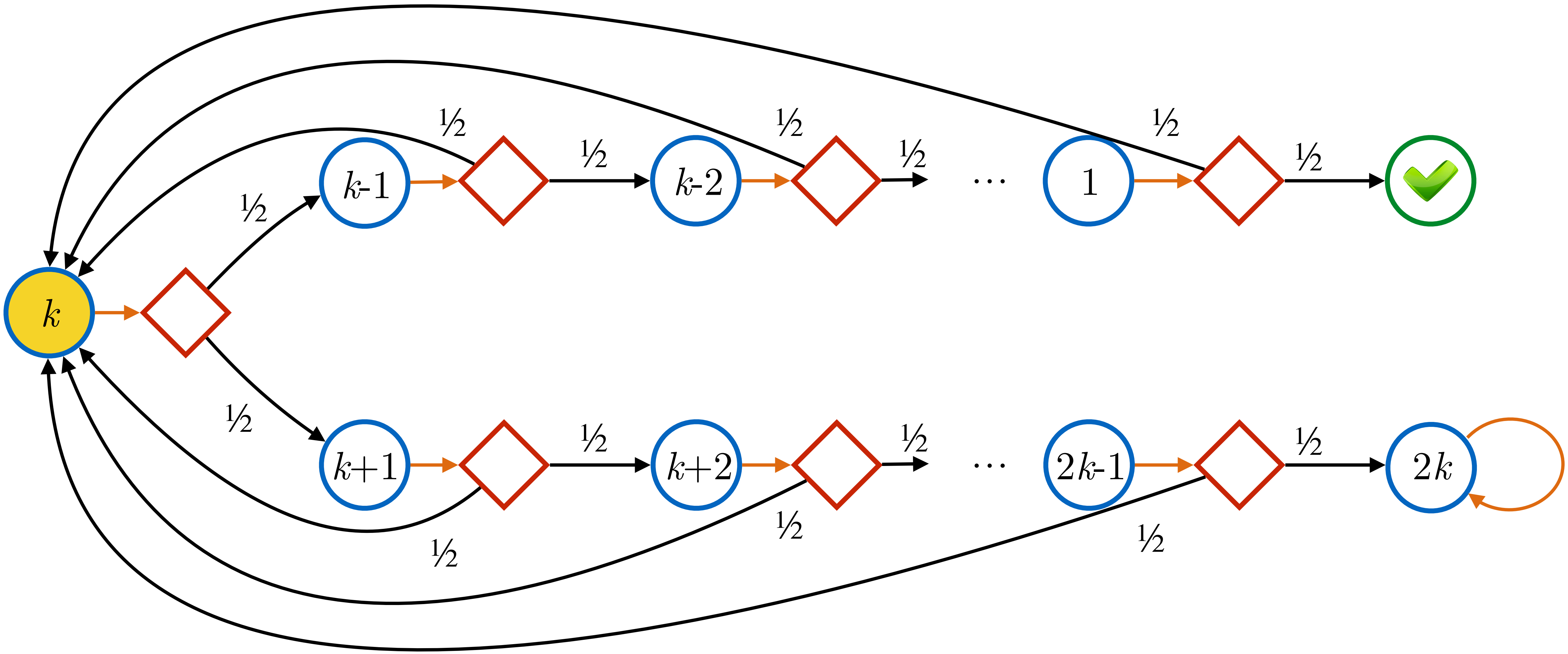
State	0	1	2	3	…	k-1	k	k+1	…	2k
Step 1	1	0	0	0	…	0	0	0	…	0

Value iteration: which guarantees?



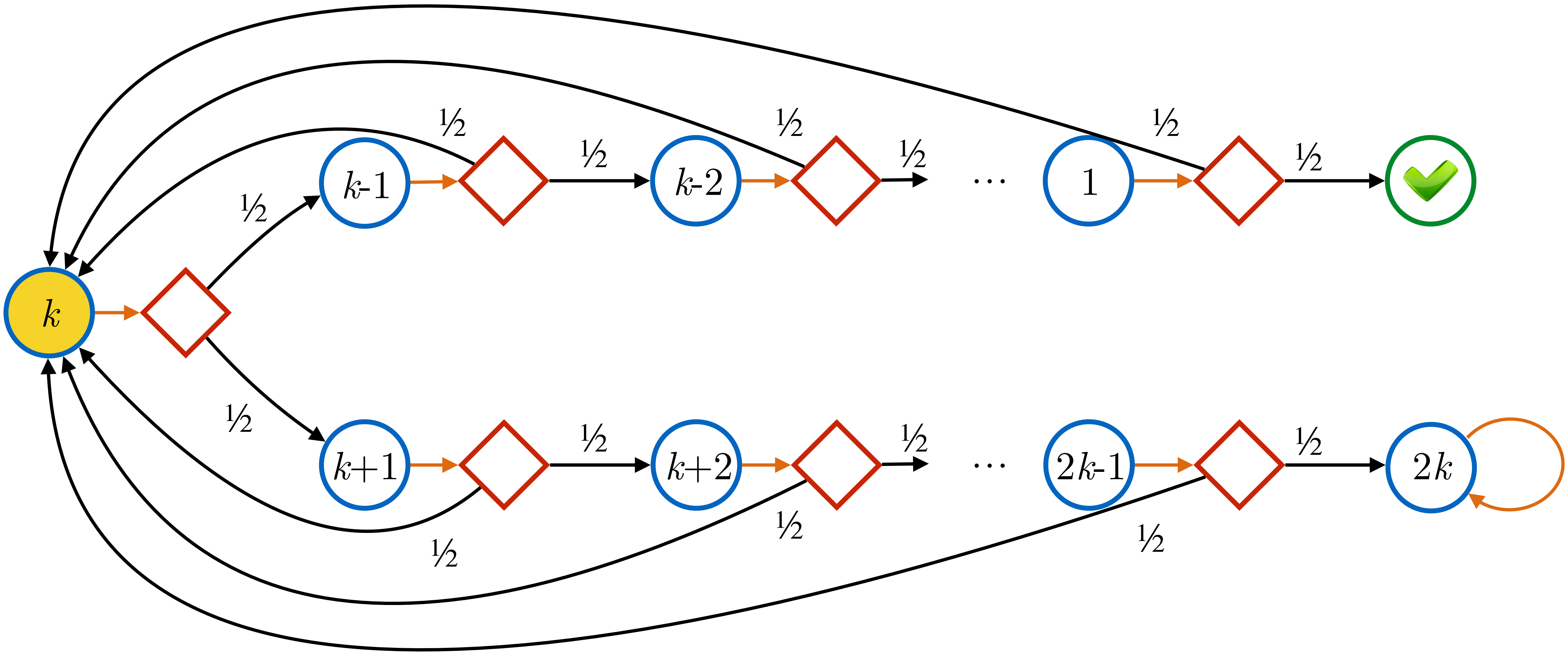
State	0	1	2	3	…	k-1	k	k+1	…	$2k$
Step 1	1	0	0	0	…	0	0	0	…	0
Step 2	1	1/2	0	0	…	0	0	0	…	0

Value iteration: which guarantees?



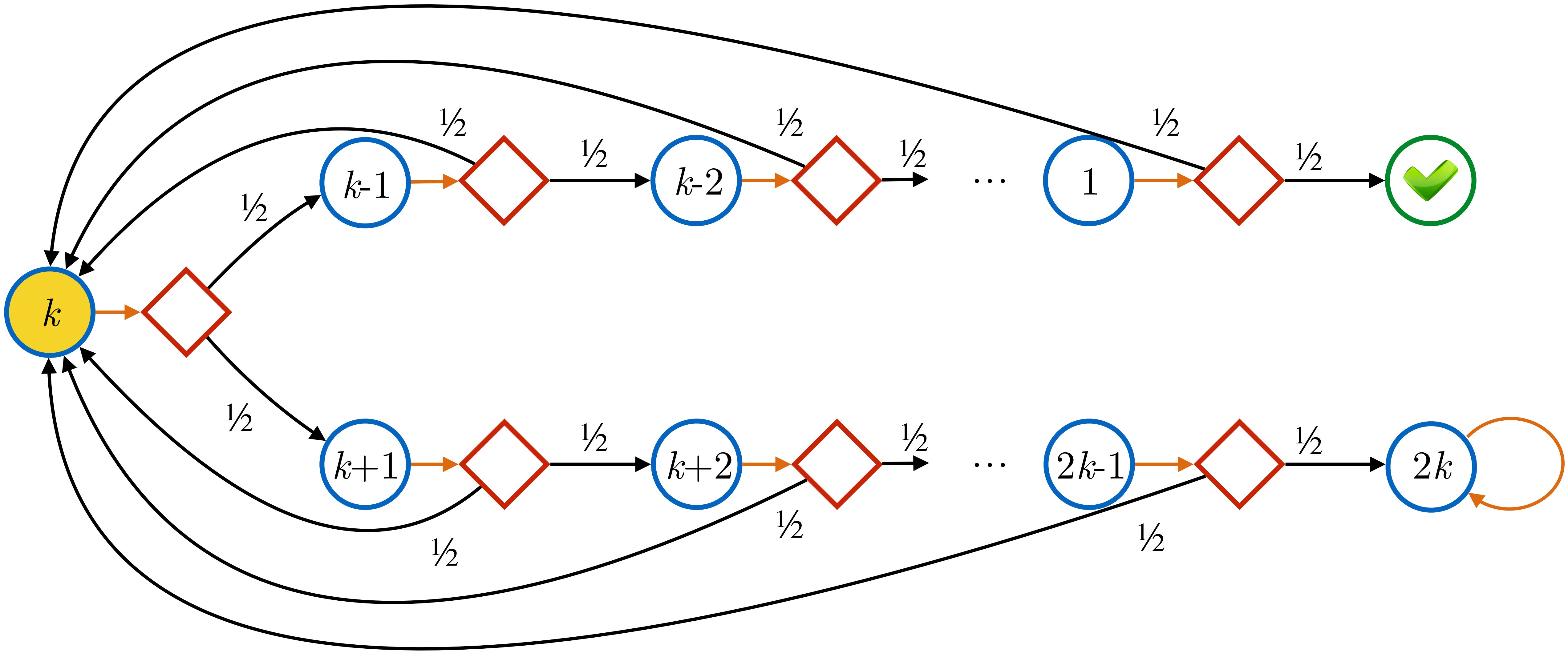
State	0	1	2	3	...	k-1	k	k+1	...	2k
Step 1	1	0	0	0	...	0	0	0	...	0
Step 2	1	1/2	0	0	...	0	0	0	...	0
Step 3	1	1/2	1/4	0	...	0	0	0	...	0

Value iteration: which guarantees?



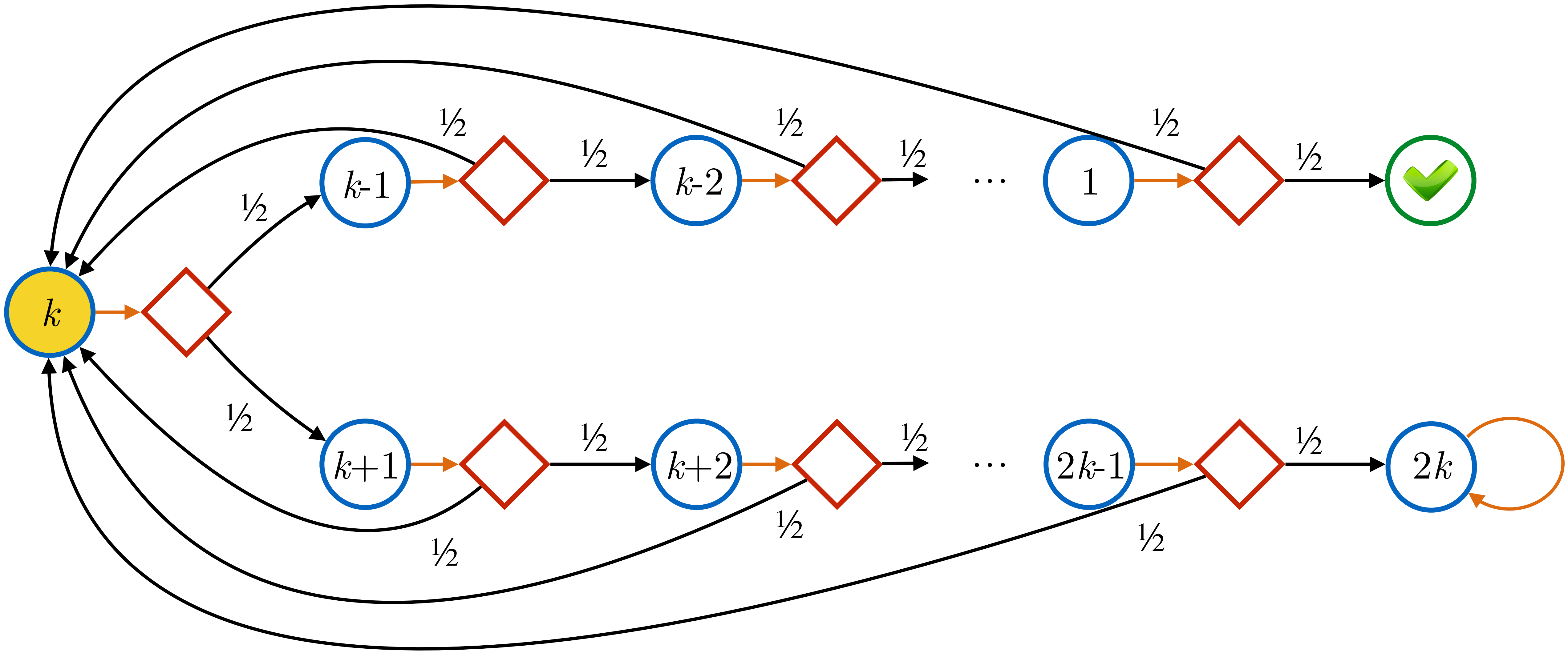
State	0	1	2	3	…	k-1	k	k+1	…	2k
Step 1	1	0	0	0	…	0	0	0	…	0
Step 2	1	1/2	0	0	…	0	0	0	…	0
Step 3	1	1/2	1/4	0	…	0	0	0	…	0
Step 4	1	1/2	1/4	1/8	…	0	0	0	…	0

Value iteration: which guarantees?



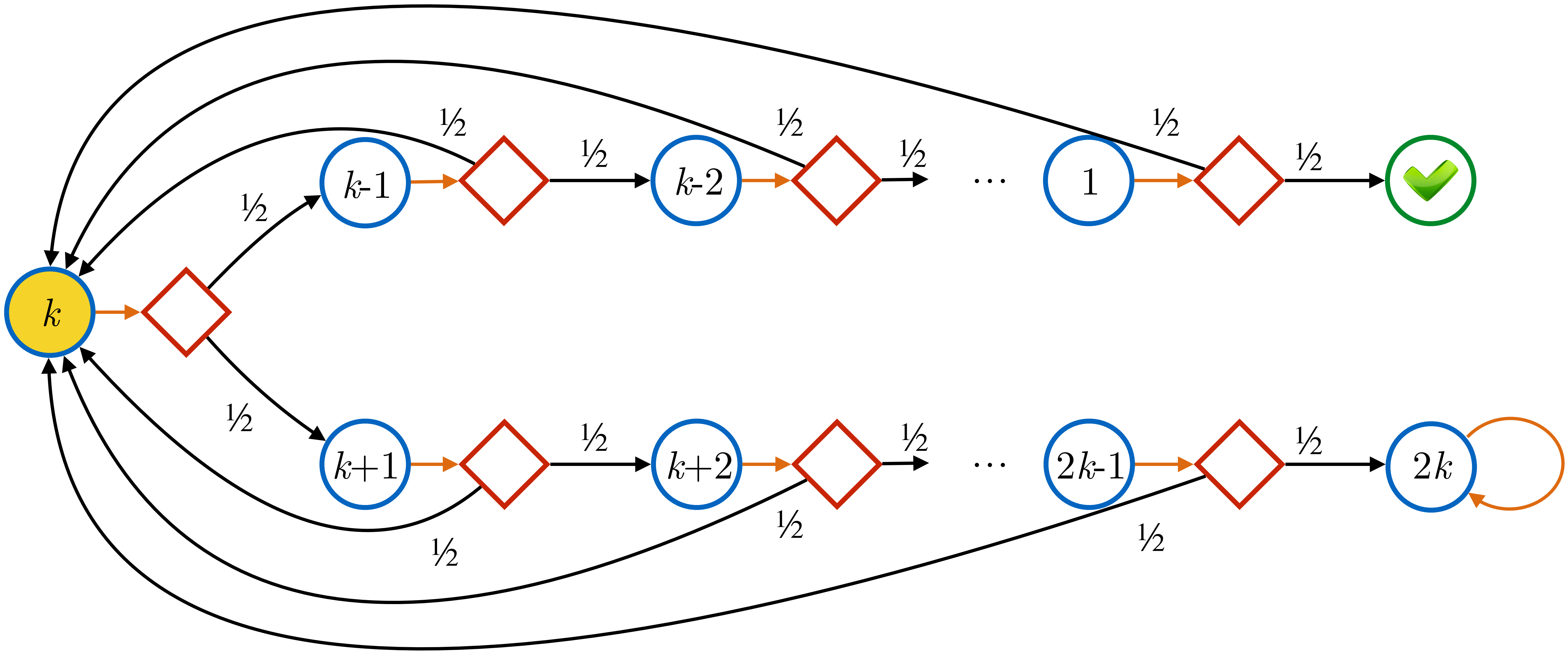
State	0	1	2	3	…	k-1	k	k+1	…	2k
Step 1	1	0	0	0	…	0	0	0	…	0
Step 2	1	1/2	0	0	…	0	0	0	…	0
Step 3	1	1/2	1/4	0	…	0	0	0	…	0
Step 4	1	1/2	1/4	1/8	…	0	0	0	…	0
…	…	…	…	…	…	…	…	…	…	…
Step k	1	1/2	1/4	1/8	…	$1/2^{k-1}$	0	0	…	0

Value iteration: which guarantees?



State	0	1	2	3	…	k-1	k	k+1	…	2k
Step 1	1	0	0	0	…	0	0	0	…	0
Step 2	1	1/2	0	0	…	0	0	0	…	0
Step 3	1	1/2	1/4	0	…	0	0	0	…	0
Step 4	1	1/2	1/4	1/8	…	0	0	0	…	0
…	…	…	…	…	…	…	…	…	…	…
Step k	1	1/2	1/4	1/8	…	$1/2^{k-1}$	0	0	…	0
Step k+1	1	1/2	1/4	1/8	…	$1/2^{k-1}$	$1/2^{k}$	0	…	0

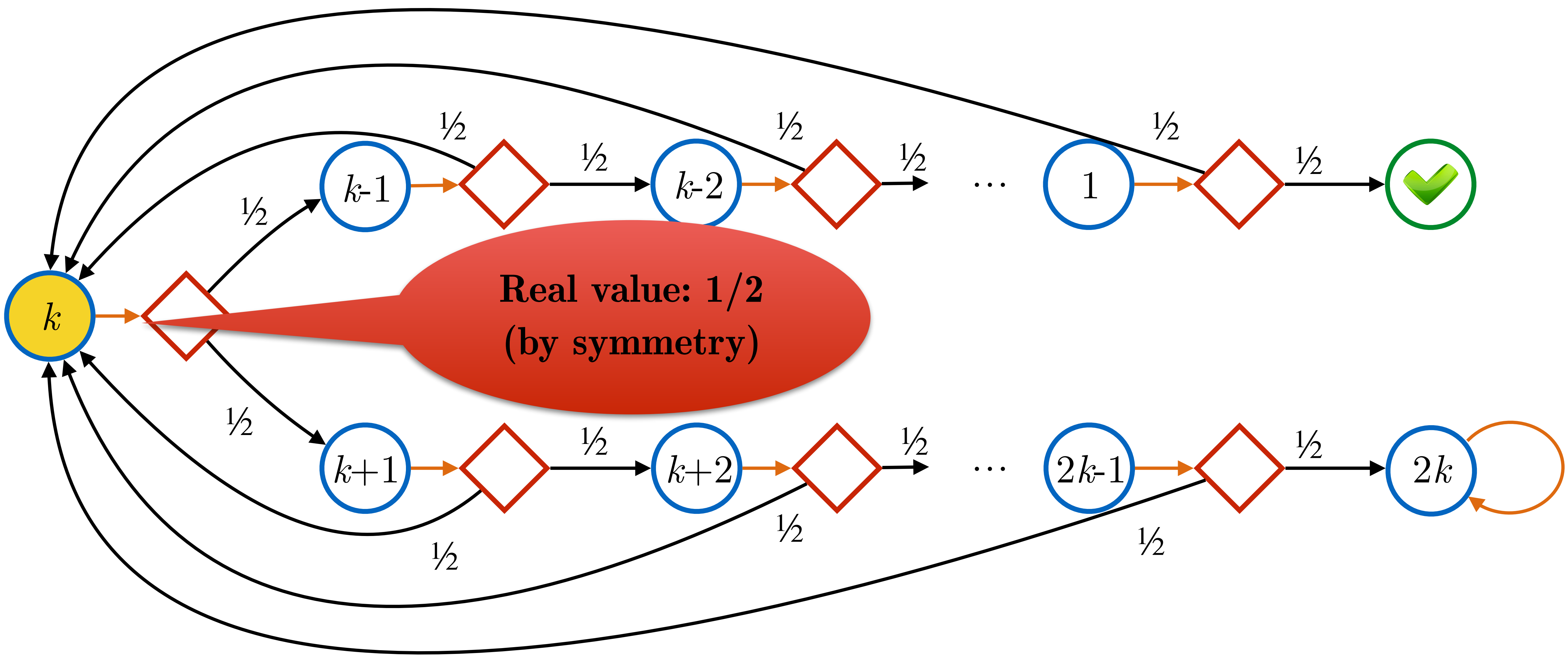
Value iteration: which guarantees?



State	0	1	2	3	...	k-1	k	k+1	...	2k
Step 1	1	0	0	0	...	0	0	0	...	0
Step 2	1	1/2	0	0	...	0	0	0	...	0
Step 3	1	1/2	1/4	0	...	0	0	0	...	0
Step 4	1	1/2	1/4	1/8	...	0	0	0	...	0
...	...	...	...	...	...	...	...	...	...	...
Step k	1	1/2	1/4	1/8	...	$1/2^{k-1}$	0	0	...	0
Step k+1	1	1/2	1/4	1/8	...	$1/2^{k-1}$	$1/2^k$	0	...	0

$\leq 1/2^k$

Value iteration: which guarantees?



State	0	1	2	3	…	k-1	k	k+1	…	2k
Step 1	1	0	0	0	…	0	0	0	…	0
Step 2	1	1/2	0	0	…	0	0	0	…	0
Step 3	1	1/2	1/4	0	…	0	0	0	…	0
Step 4	1	1/2	1/4	1/8	…	0	0	0	…	0
…	…	…	…	…	…	…	…	…	…	…
Step k	1	1/2	1/4	1/8	…	$1/2^{k-1}$	0	0	…	0
Step k+1	1	1/2	1/4	1/8	…	$1/2^{k-1}$	$1/2^k$	0	…	0

$\leq 1/2^k$

Contributions

Contributions

1. Enhanced value iteration algorithm with **strong guarantees**

Contributions

1. Enhanced value iteration algorithm with **strong guarantees**
    - performs **two** value iterations in **parallel**

Contributions

1. Enhanced value iteration algorithm with **strong guarantees**
    - performs **two** value iterations in **parallel**
    - keeps an **interval** of possible optimal values

Contributions

1. Enhanced value iteration algorithm with **strong guarantees**
   - performs **two** value iterations in **parallel**
   - keeps an **interval** of possible optimal values
   - uses the interval for the **stopping criterion**

Contributions

1. Enhanced value iteration algorithm with **strong guarantees**
   - performs **two** value iterations in **parallel**
   - keeps an **interval** of possible optimal values
   - uses the interval for the **stopping criterion**
2. Study of the **speed of convergence**

Contributions

1. Enhanced value iteration algorithm with **strong guarantees**
   - performs **two** value iterations in **parallel**
   - keeps an **interval** of possible optimal values
   - uses the interval for the **stopping criterion**
2. Study of the **speed of convergence**
   - also applies to classical value iteration

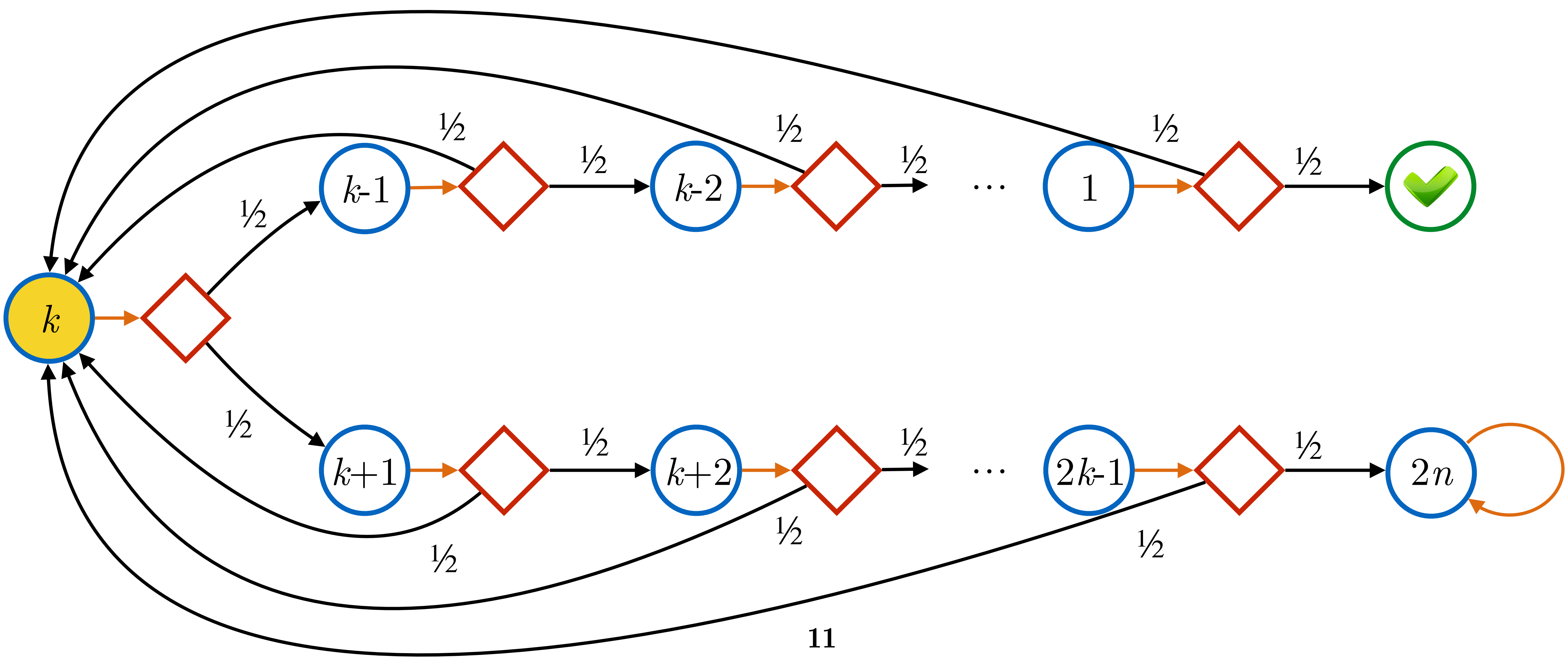
Contributions

1. Enhanced value iteration algorithm with **strong guarantees**
    - performs **two** value iterations in **parallel**
    - keeps an **interval** of possible optimal values
    - uses the interval for the **stopping criterion**
2. Study of the **speed of convergence**
    - also applies to classical value iteration
3. Improved **rounding** procedure for **exact** computation

Interval iteration

$$x_s^{(0)} = \begin{cases} 1 & \text{if } s = \text{✔} \\ 0 & \text{otherwise} \end{cases}$$

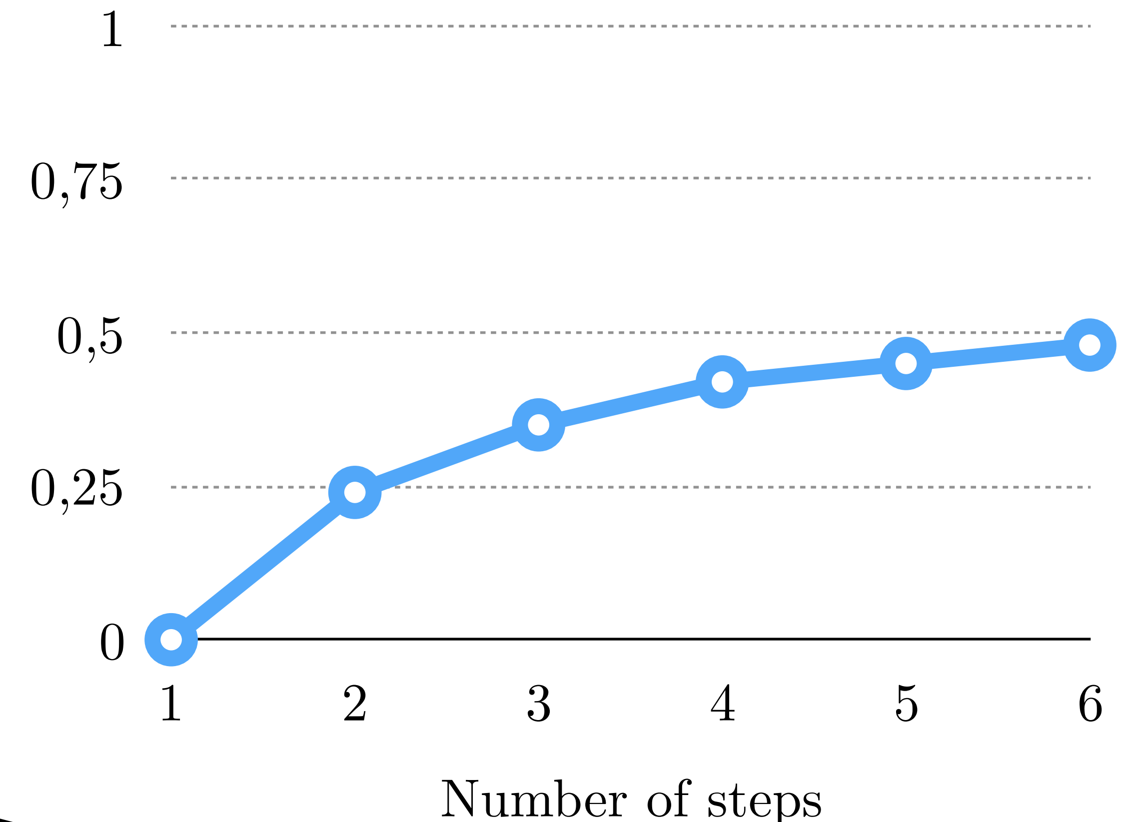
$$x_s^{(n+1)} = \max_{a \in \alpha} \sum_{s' \in S} \delta(s,a)(s') \times x_{s'}^{(n)}$$

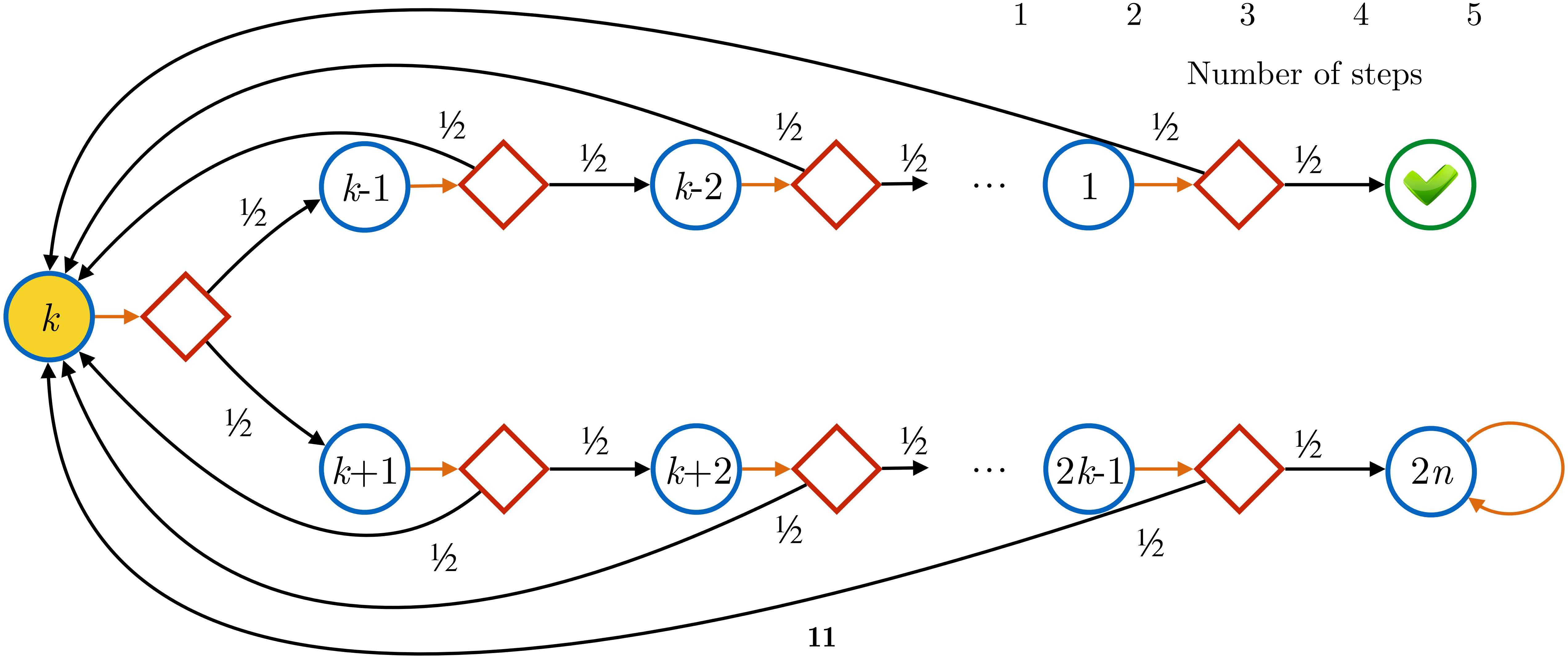


Interval iteration

$$x_s^{(0)} = \begin{cases} 1 & \text{if } s = ✔ \\ 0 & \text{otherwise} \end{cases}$$

$$x_s^{(n+1)} = \max_{a \in \alpha} \sum_{s' \in S} \delta(s,a)(s') \times x_{s'}^{(n)}$$



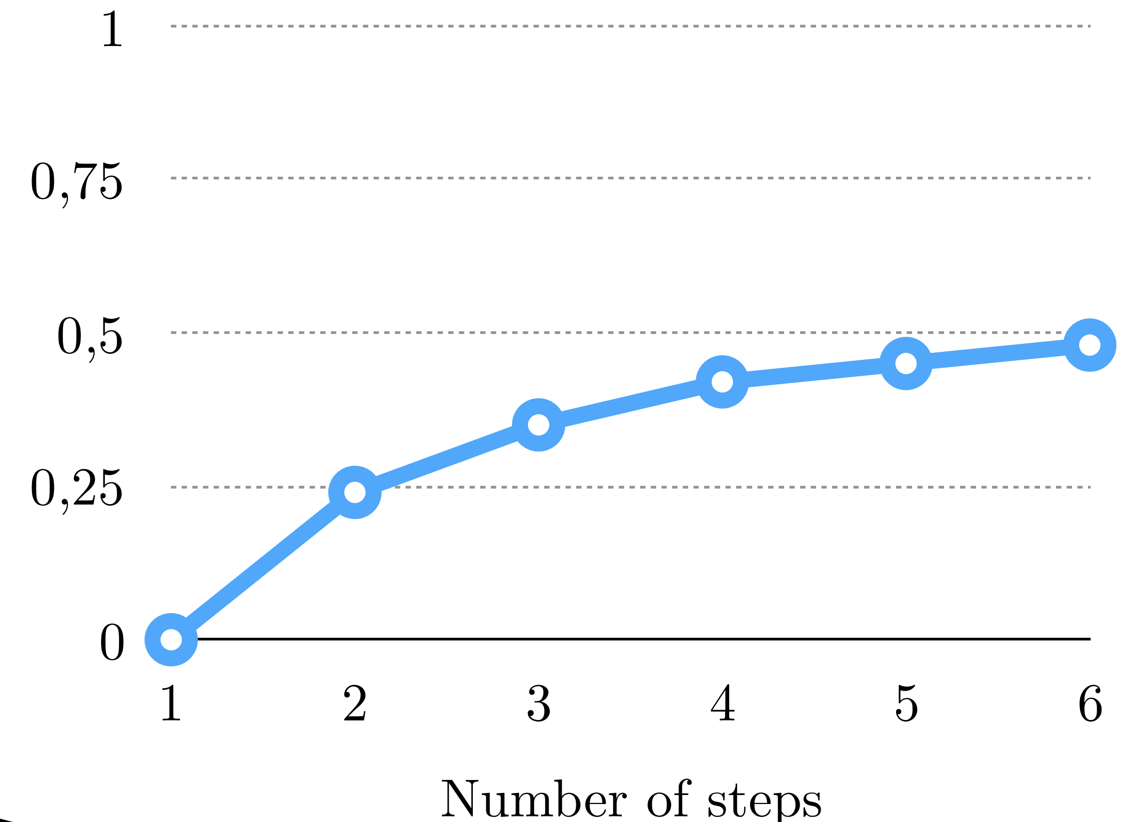


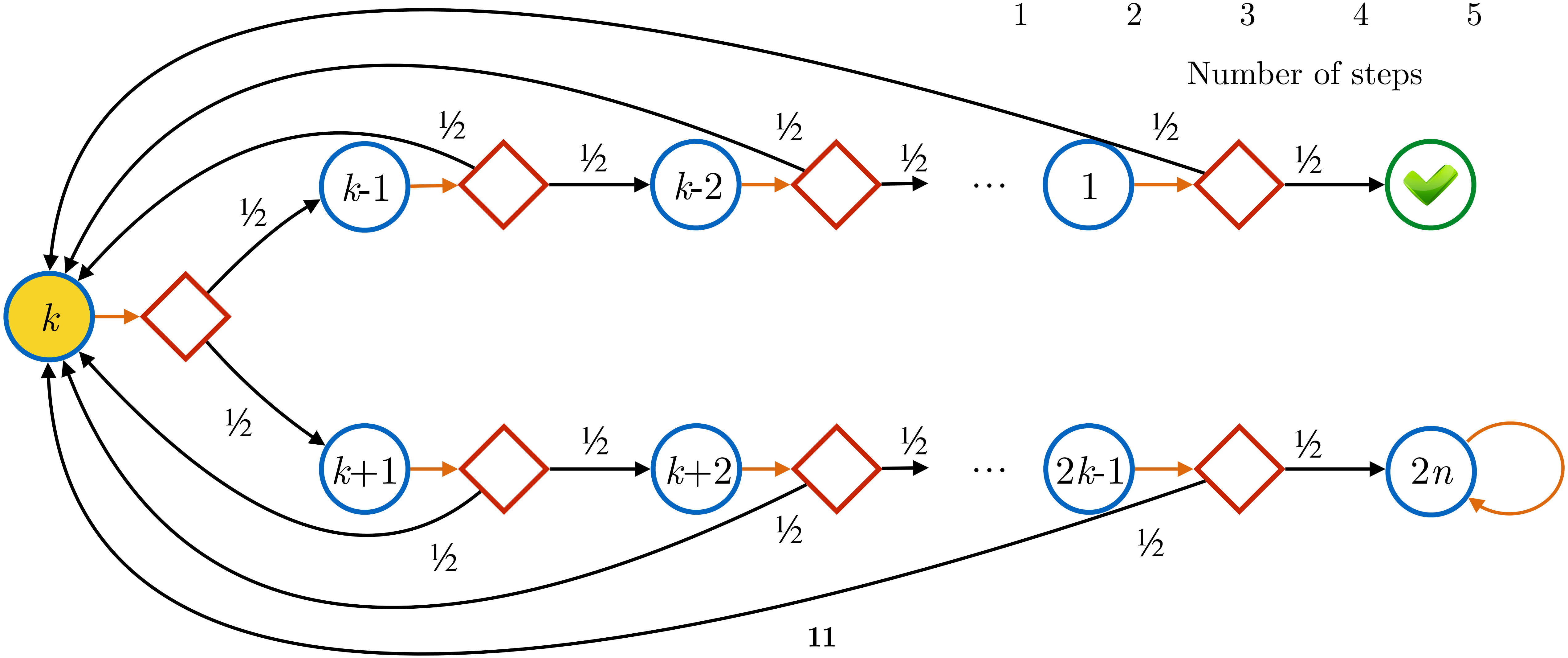
Interval iteration

$$x^{(0)}_{s} = \begin{cases} 1 & \text{if } s = ✔ \\ 0 & \text{otherwise} \end{cases}$$

$$x^{(n+1)} = f_{\max}(x^{(n)})$$

$$f_{\max}(x)_s = \max_{a \in \alpha} \sum_{s' \in S} \delta(s,a)(s') \times x_{s'}$$



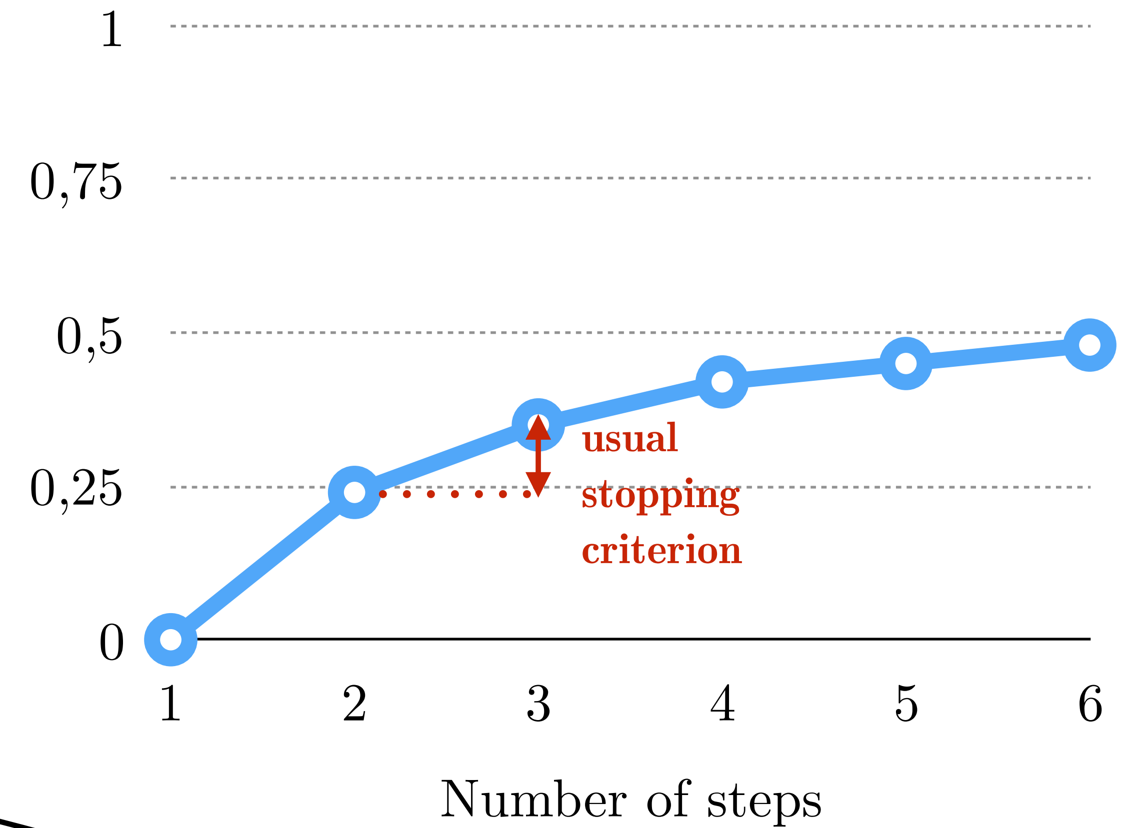


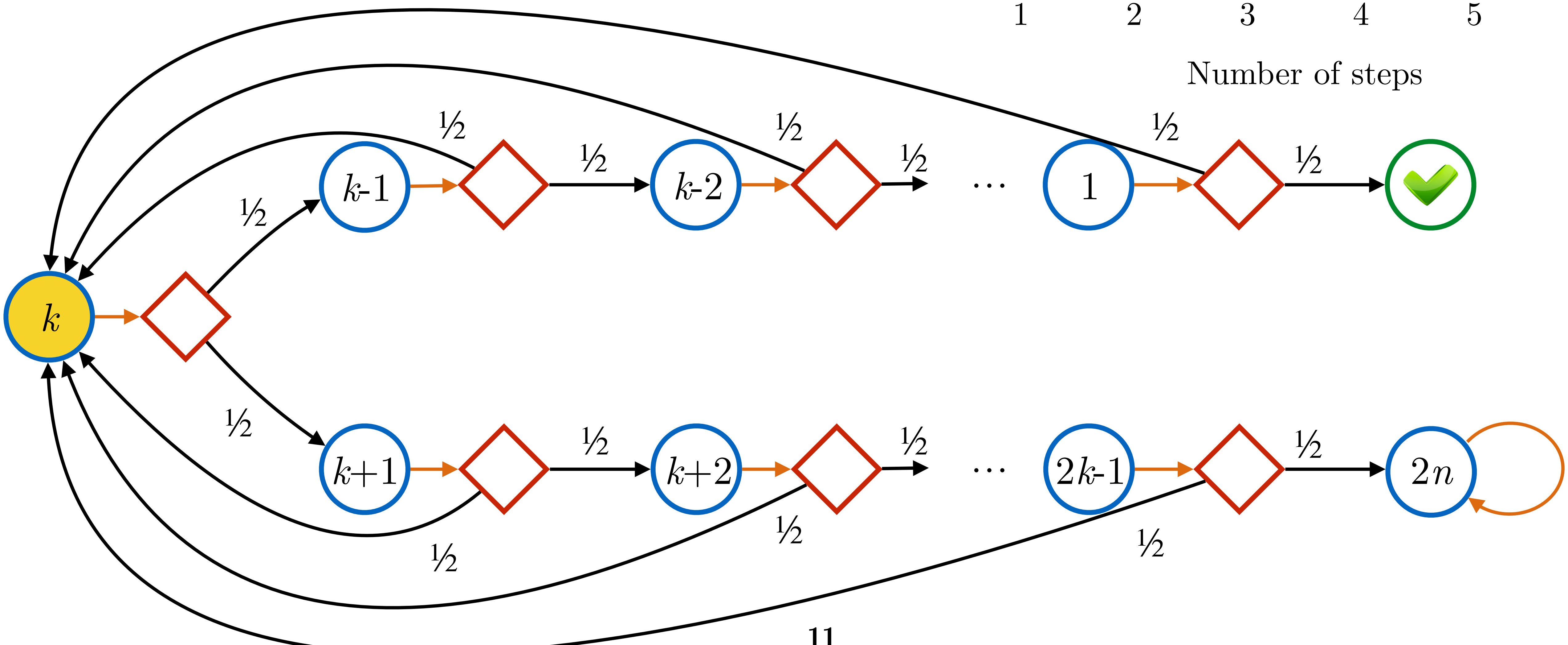
Interval iteration

$$x^{(0)}_{s} = \begin{cases} 1 & \text{if } s = \text{✓} \\ 0 & \text{otherwise} \end{cases}$$

$$x^{(n+1)} = f_{\max}(x^{(n)})$$

$$f_{\max}(x)_{s} = \max_{a \in \alpha} \sum_{s' \in S} \delta(s, a)(s') \times x_{s'}$$



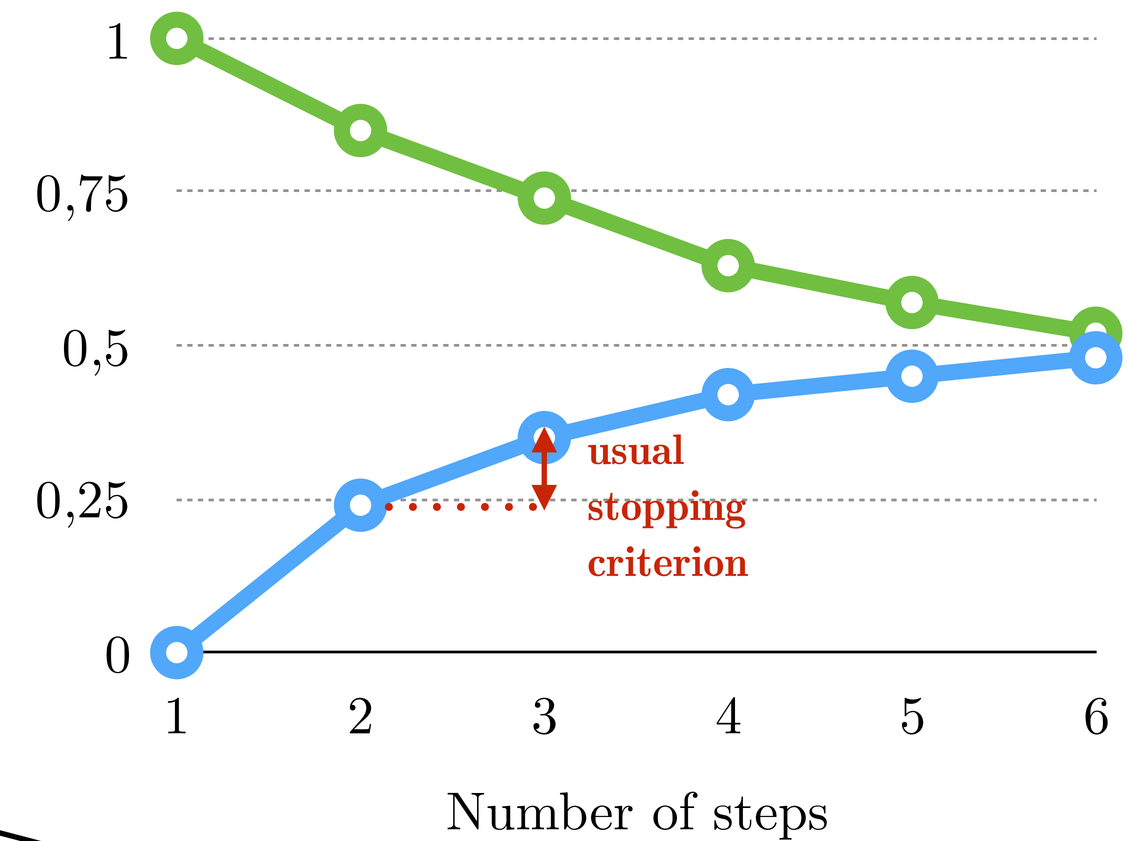


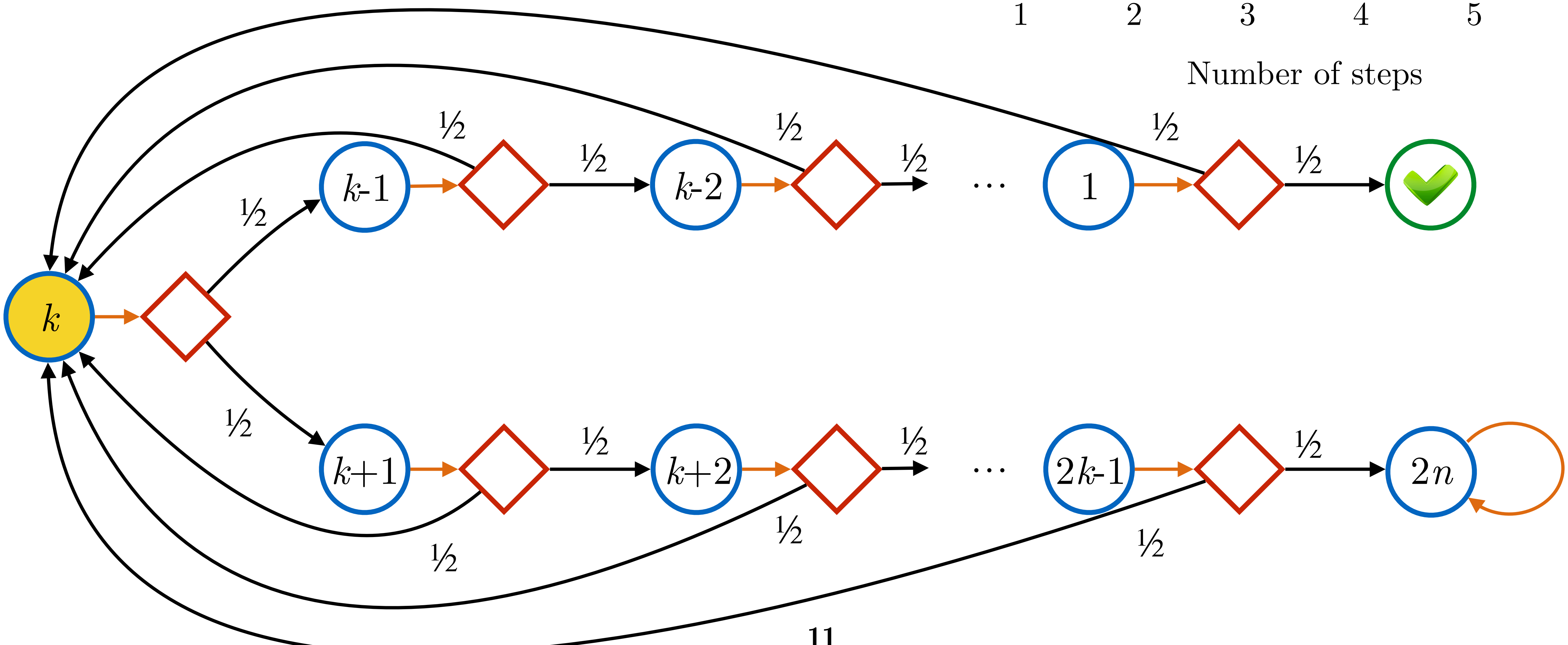
Interval iteration

$$x^{(0)}_s = \begin{cases} 1 & \text{if } s = \checkmark \\ 0 & \text{otherwise} \end{cases}$$

$$x^{(n+1)} = f_{\max}(x^{(n)})$$

$$f_{\max}(x)_s = \max_{a\in\alpha} \sum_{s'\in S} \delta(s,a)(s') \times x_{s'}$$



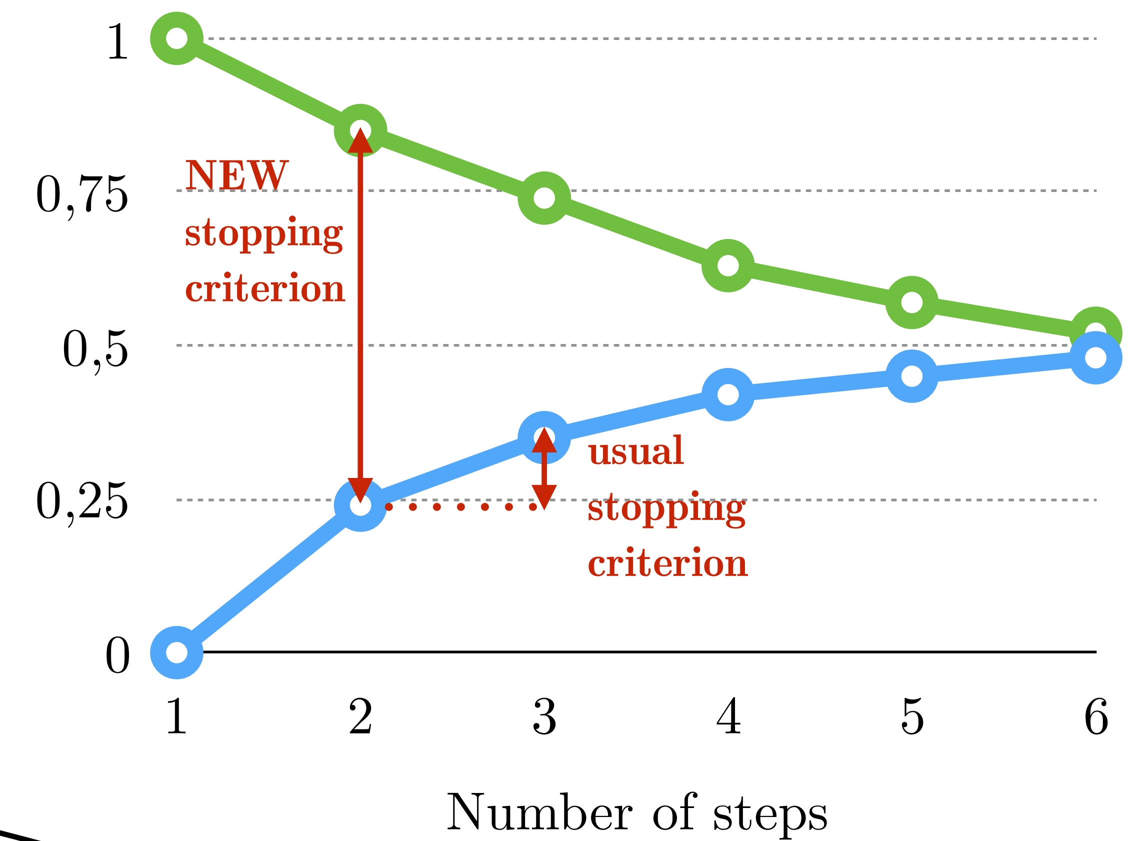


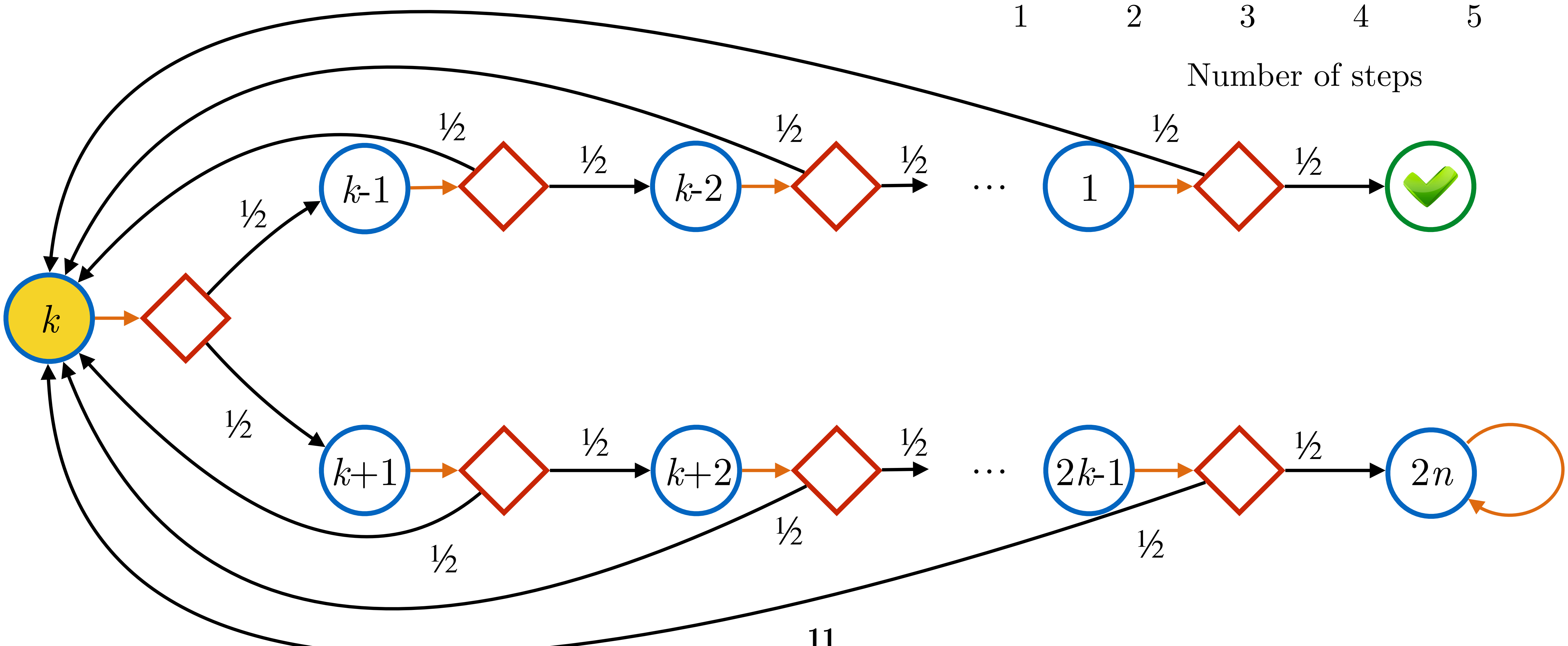
Interval iteration

$$x_s^{(0)} = \begin{cases} 1 & \text{if } s = ✓ \\ 0 & \text{otherwise} \end{cases}$$

$$x^{(n+1)} = f_{\max}(x^{(n)})$$

$$f_{\max}(x)_s = \max_{a \in \alpha} \sum_{s' \in S} \delta(s,a)(s') \times x_{s'}$$



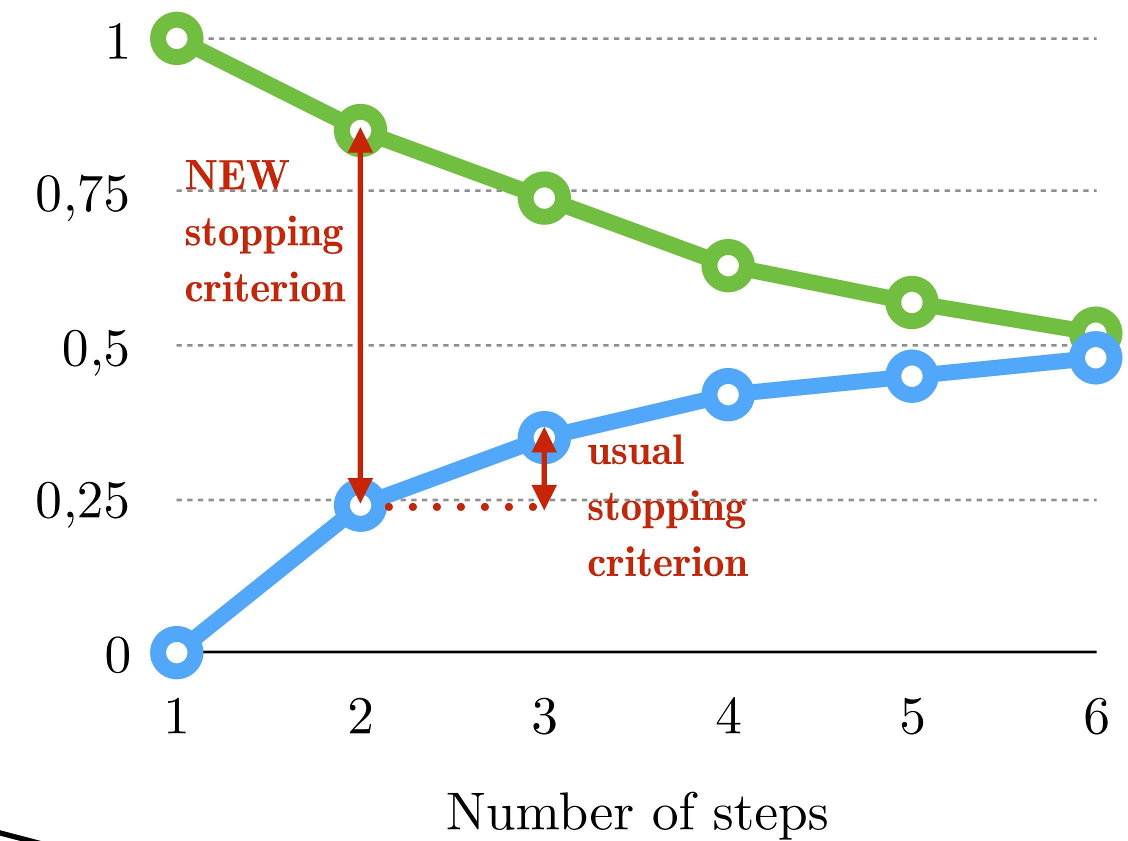


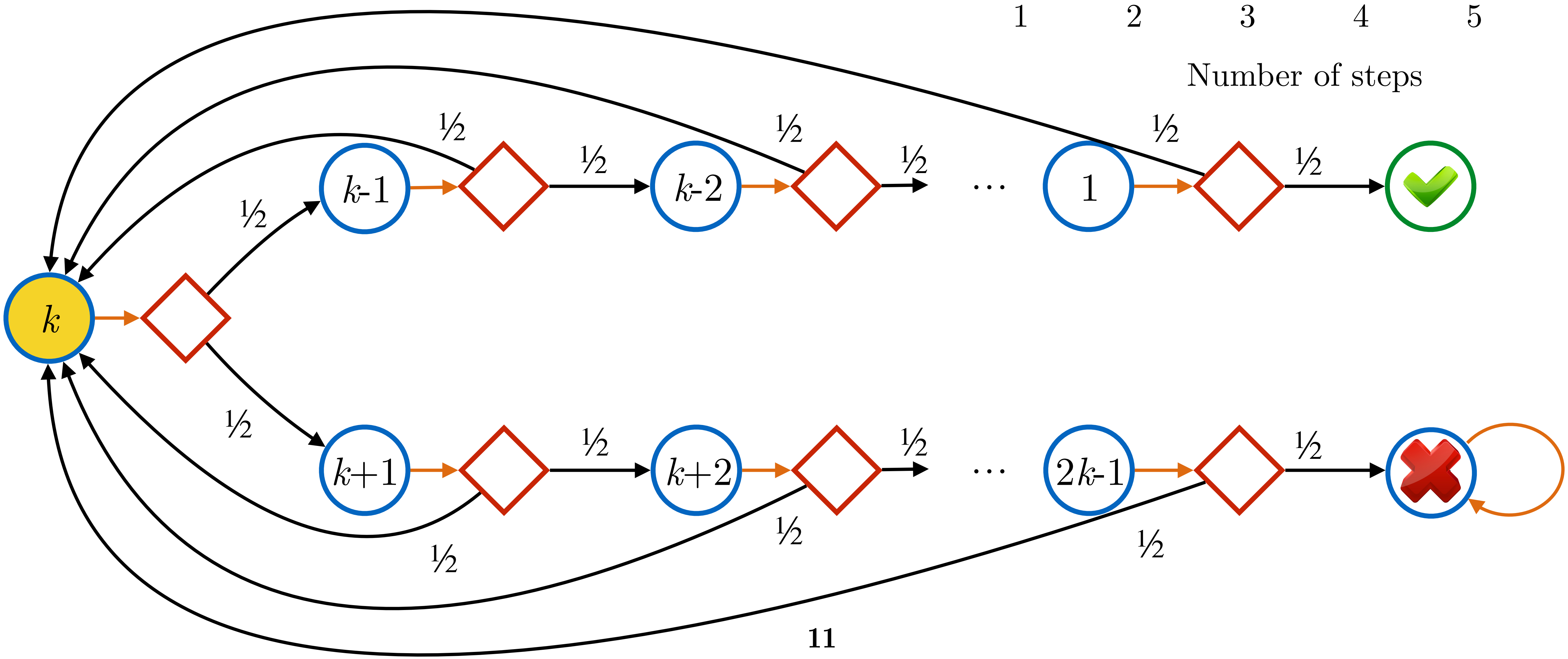
Interval iteration

$$x_s^{(0)} = \begin{cases} 1 & \text{if } s = \checkmark \\ 0 & \text{otherwise} \end{cases} \qquad y_s^{(0)} = \begin{cases} 0 & \text{if } s = \text{✗} \\ 1 & \text{otherwise} \end{cases}$$

$$x^{(n+1)} = f_{\max}(x^{(n)}) \qquad y^{(n+1)} = f_{\max}(y^{(n)})$$

$$f_{\max}(x)_s = \max_{a \in \alpha} \sum_{s' \in S} \delta(s,a)(s') \times x_{s'}$$



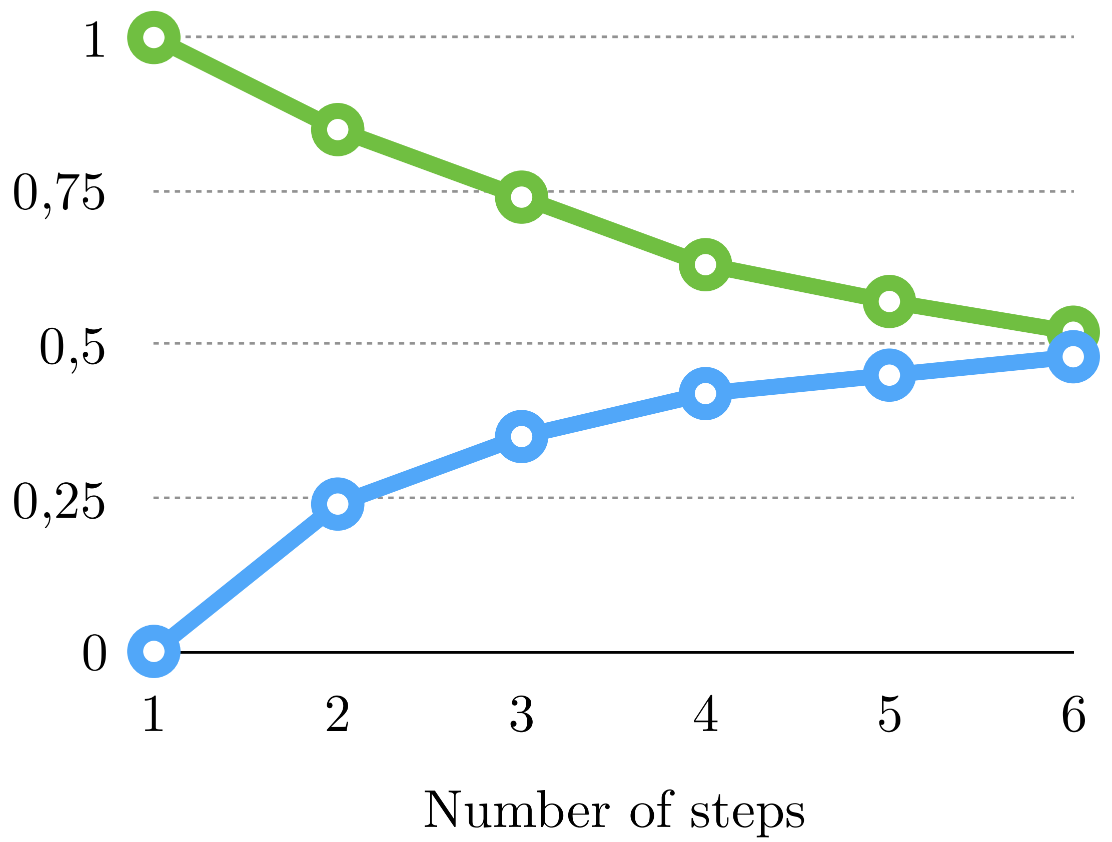


Fixed point characterization

$$\left(\Pr_s^{\max}(\mathsf{F}\,\checkmark)\right)_{s\in S} \text{ is the smallest fixed point of } f_{\max}.$$

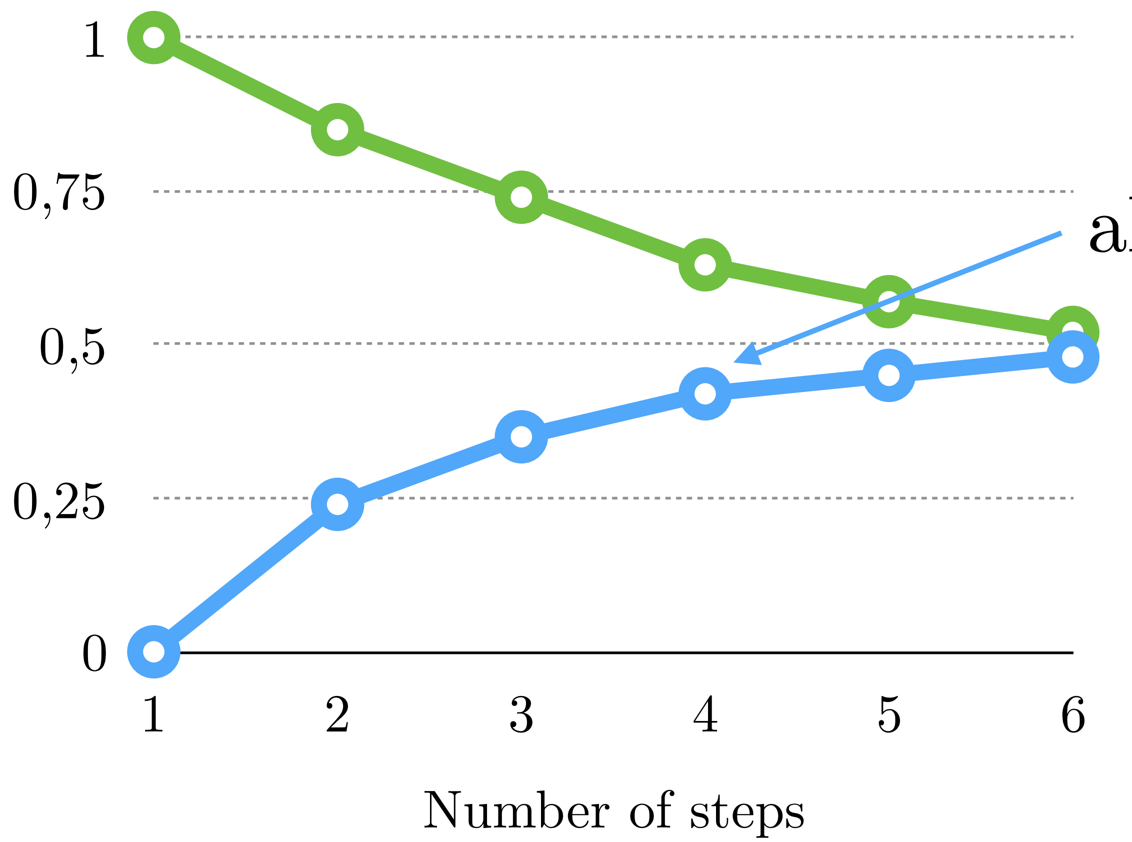
Fixed point characterization

$\left(\mathrm{Pr}_{s}^{\max}(\mathsf{F}\ ✓)\right)_{s\in S}$ is the smallest fixed point of $f_{\max}$.



Fixed point characterization

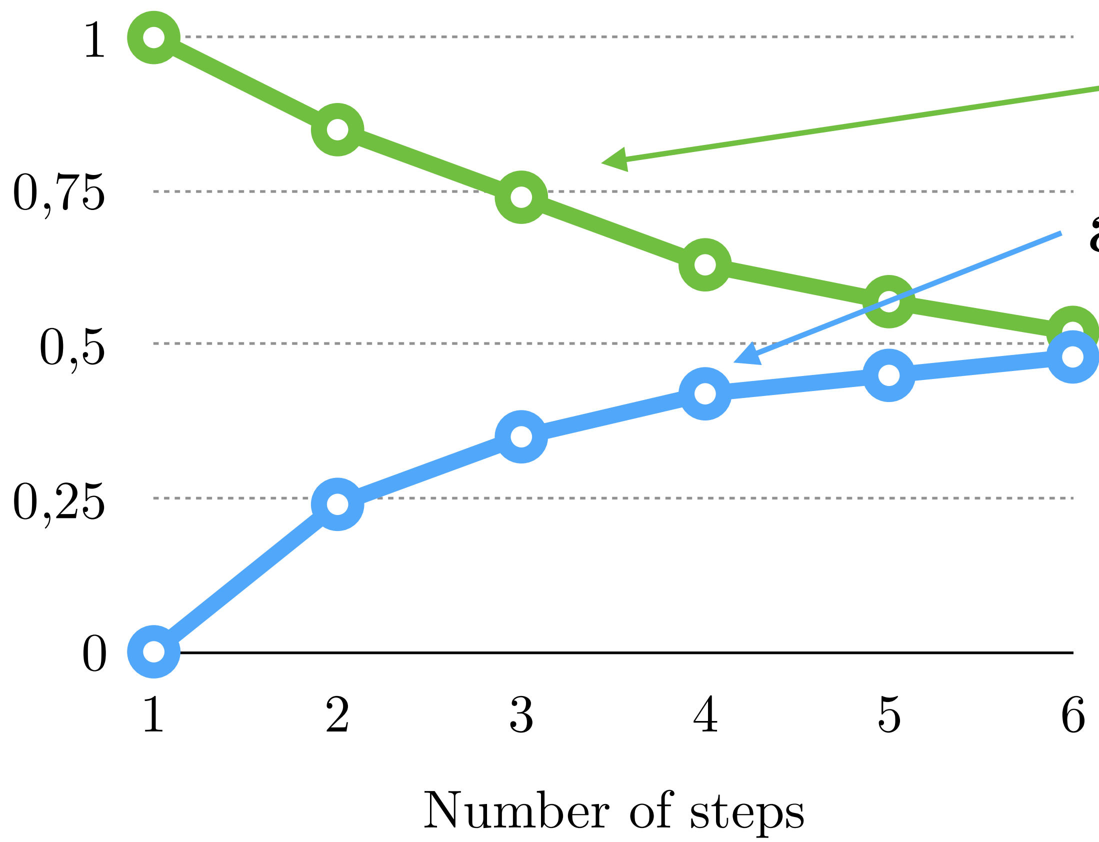
$\left(\Pr_s^{\max}(\mathsf{F}\ ✔)\right)_{s\in S}$ is the smallest fixed point of $f_{\max}$.



always converges towards $\left(\Pr_s^{\max}(\mathsf{F}\ ✔)\right)_{s\in S}$

Fixed point characterization

$\left(\mathrm{Pr}_s^{\max}(\mathsf{F}\ \checkmark)\right)_{s\in S}$ is the smallest fixed point of $f_{\max}$.

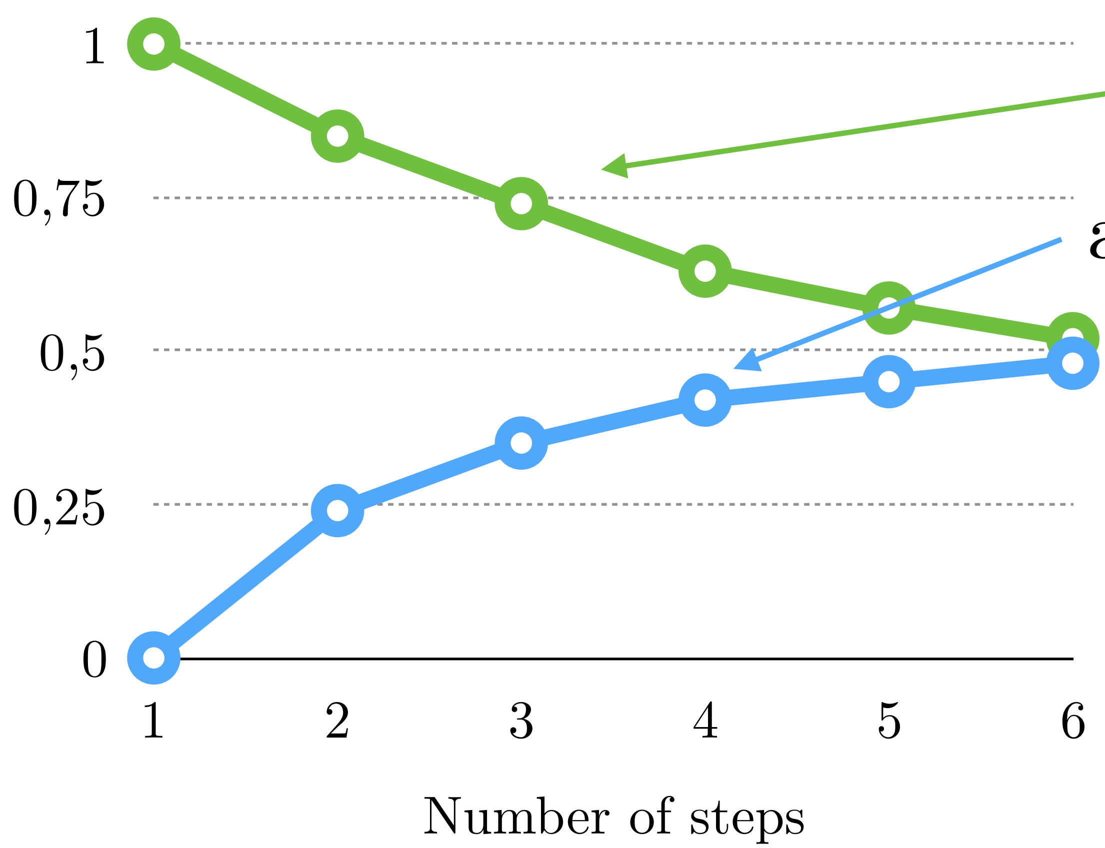


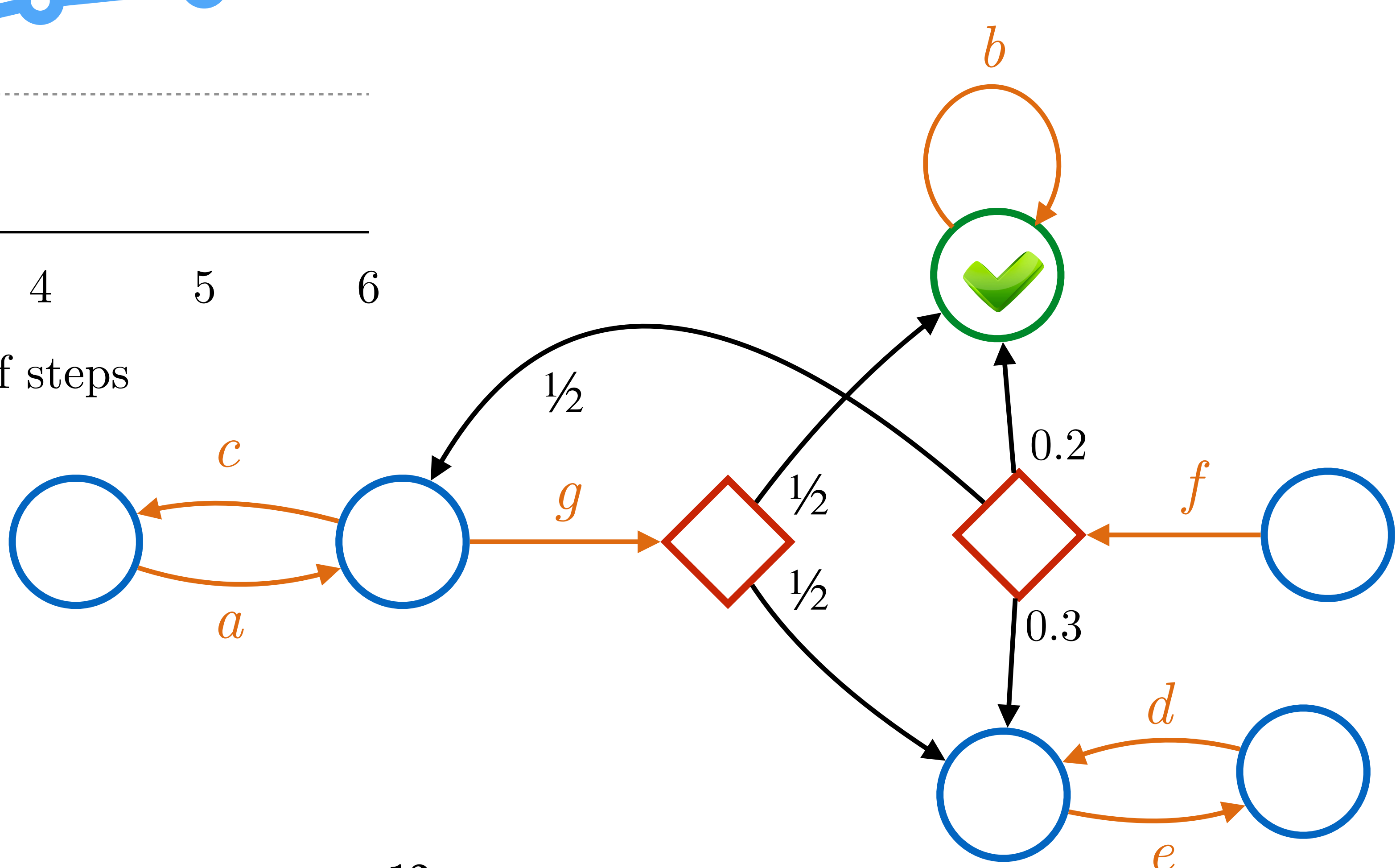
not always...!

always converges towards $\left(\mathrm{Pr}_s^{\max}(\mathsf{F}\ \checkmark)\right)_{s\in S}$

Fixed point characterization

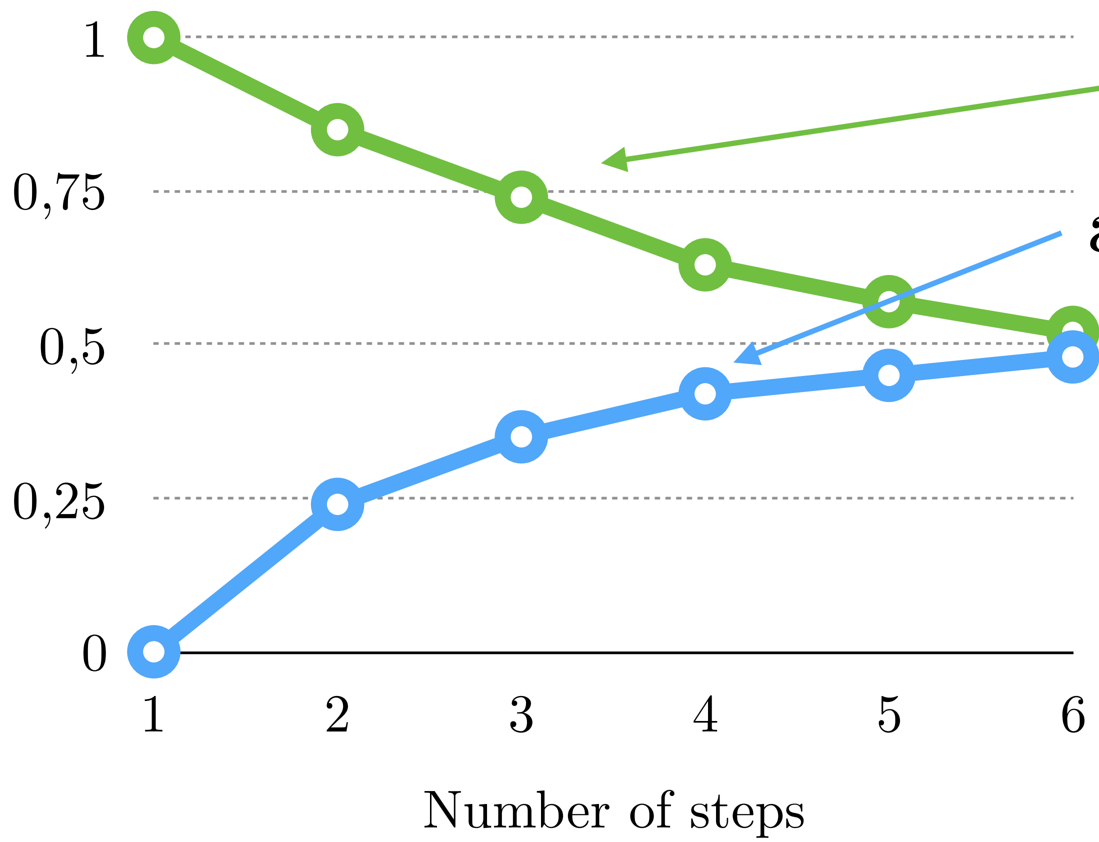
$\left(\Pr_s^{\max}(\mathsf{F}\ ✓)\right)_{s\in S}$ is the smallest fixed point of $f_{\max}$.





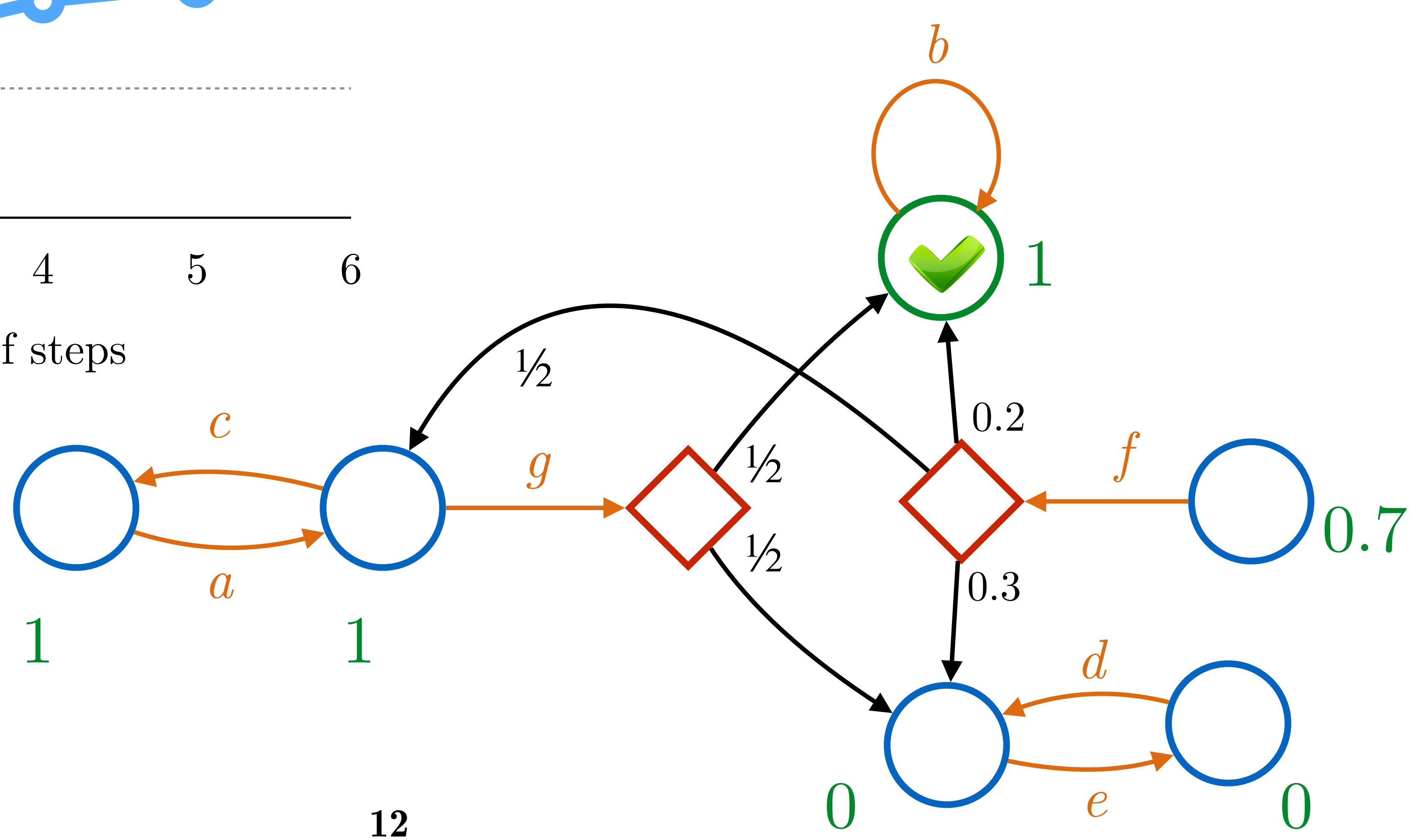
Fixed point characterization

$\left(\Pr_s^{\max}(\mathsf{F}\ ✔)\right)_{s\in S}$ is the smallest fixed point of $f_{\max}$.



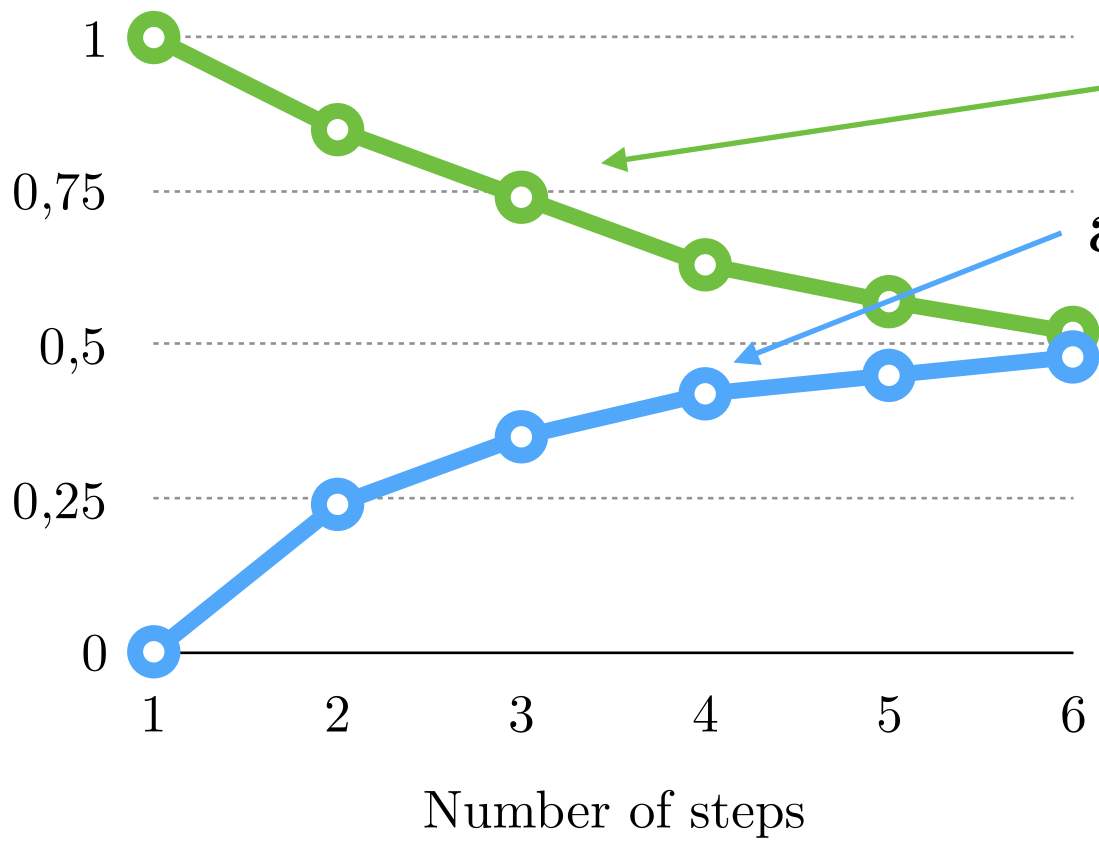
not always...!

always converges towards $\left(\Pr_s^{\max}(\mathsf{F}\ ✔)\right)_{s\in S}$



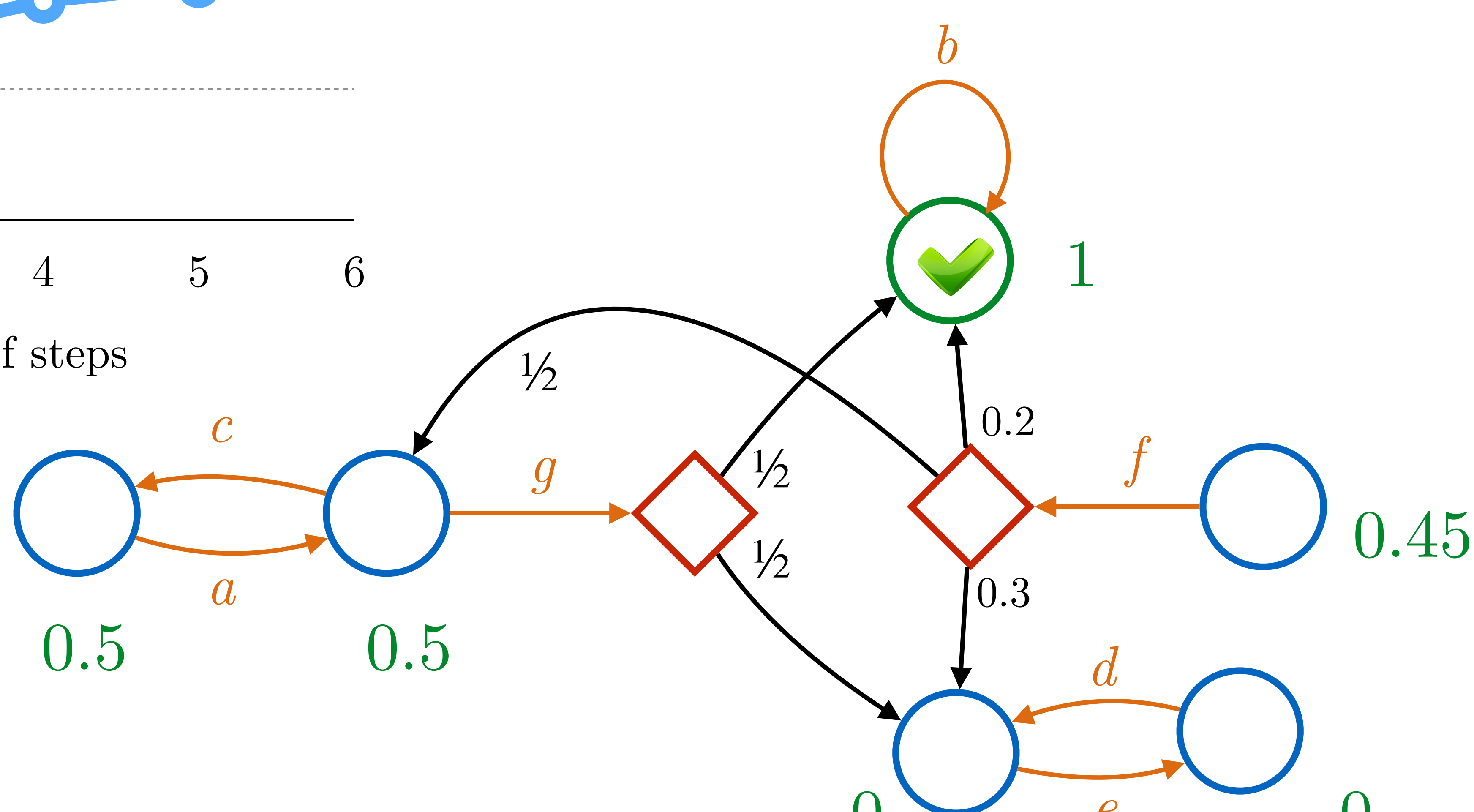
Fixed point characterization

$\left(\Pr_s^{\max}(\mathsf{F}\,✔)\right)_{s\in S}$ is the smallest fixed point of $f_{\max}$.



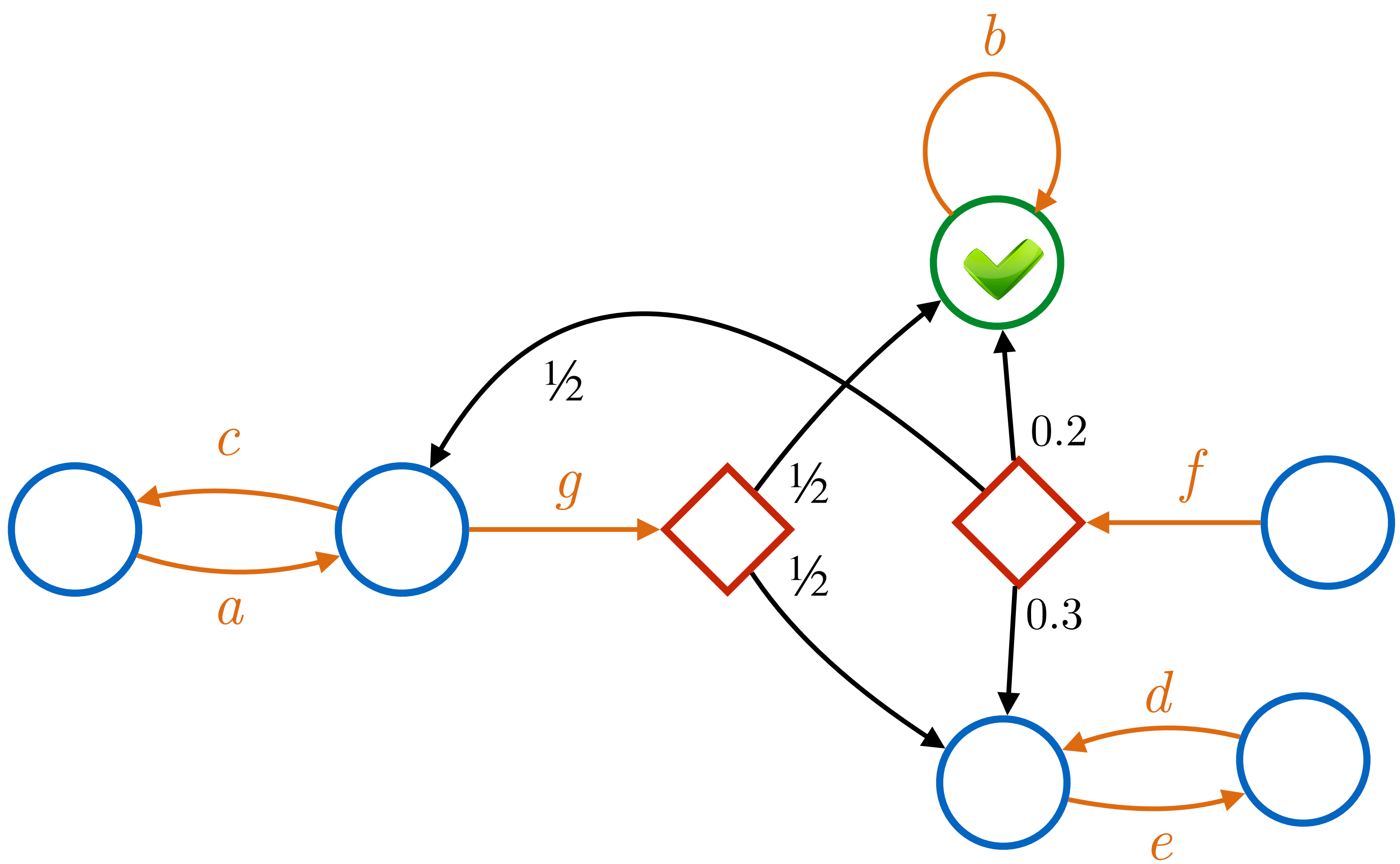
not always...!

always converges towards $\left(\Pr_s^{\max}(\mathsf{F}\,✔)\right)_{s\in S}$



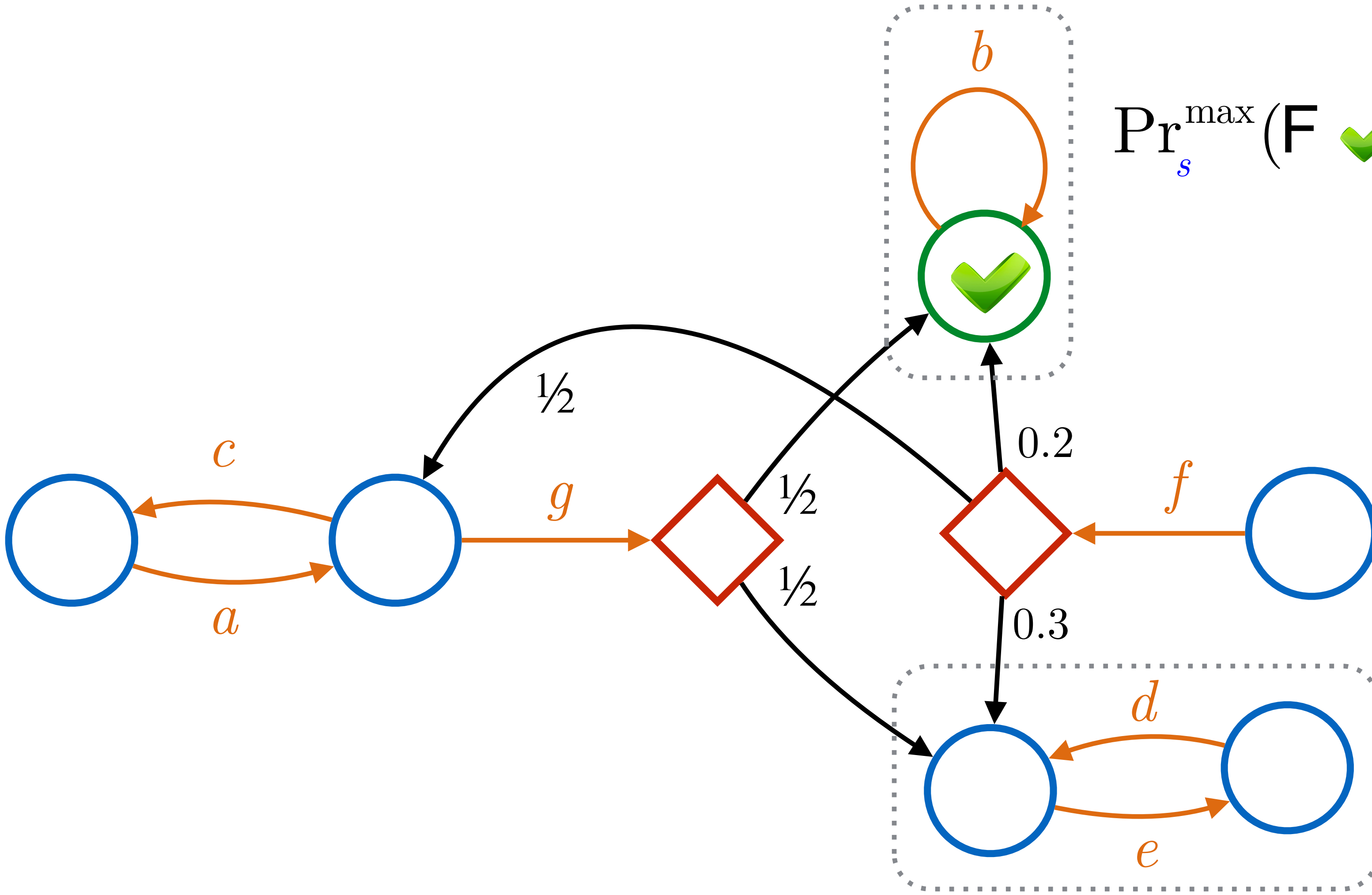
Solution: ensure uniqueness!

Usual techniques applied for MDPs do not apply…



Solution: ensure uniqueness!

Usual techniques applied for MDPs do not apply…

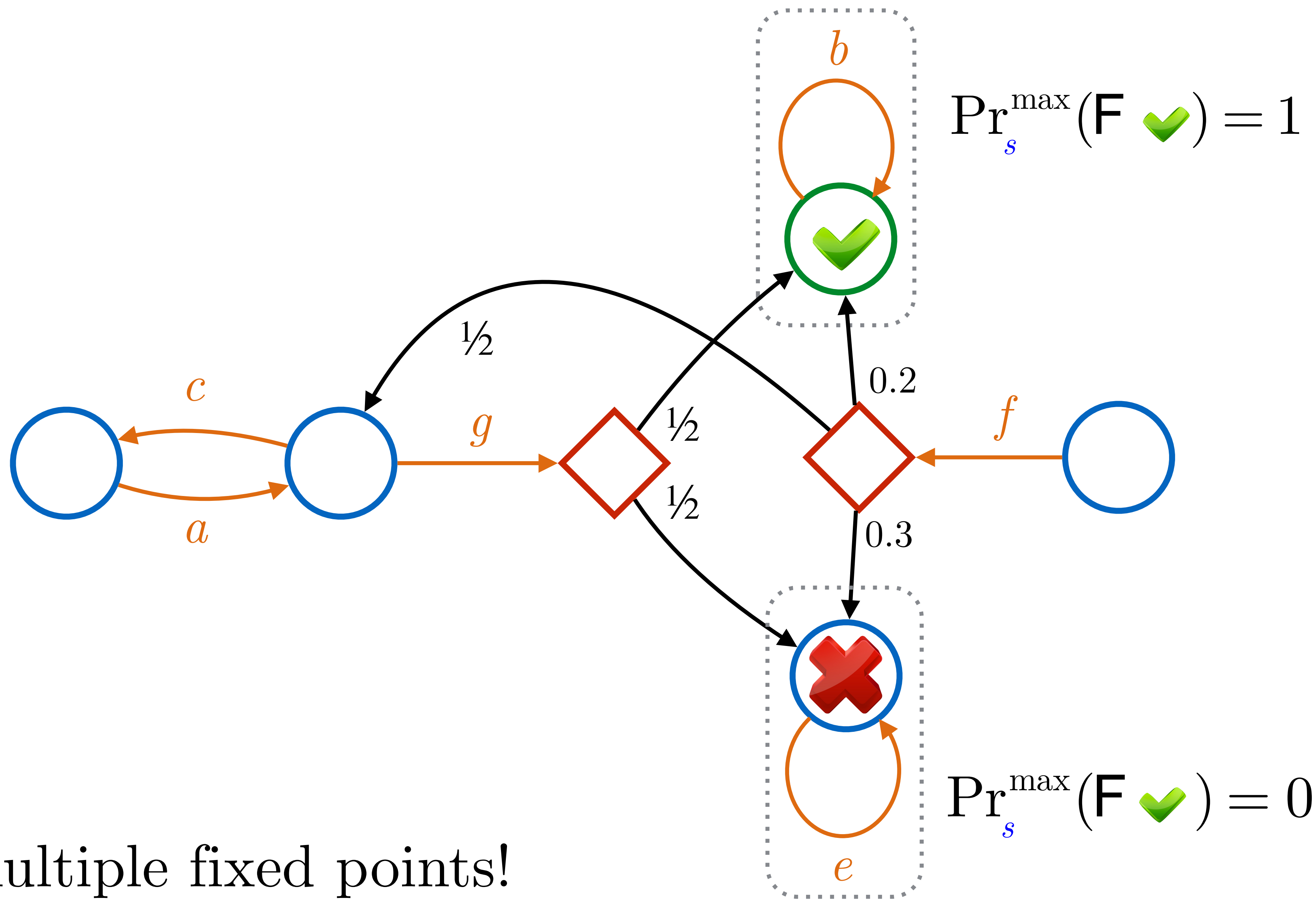


$\mathrm{Pr}_s^{\max}(\mathsf{F}\ ✓) = 1$

$\mathrm{Pr}_s^{\max}(\mathsf{F}\ ✓) = 0$

Solution: ensure uniqueness!

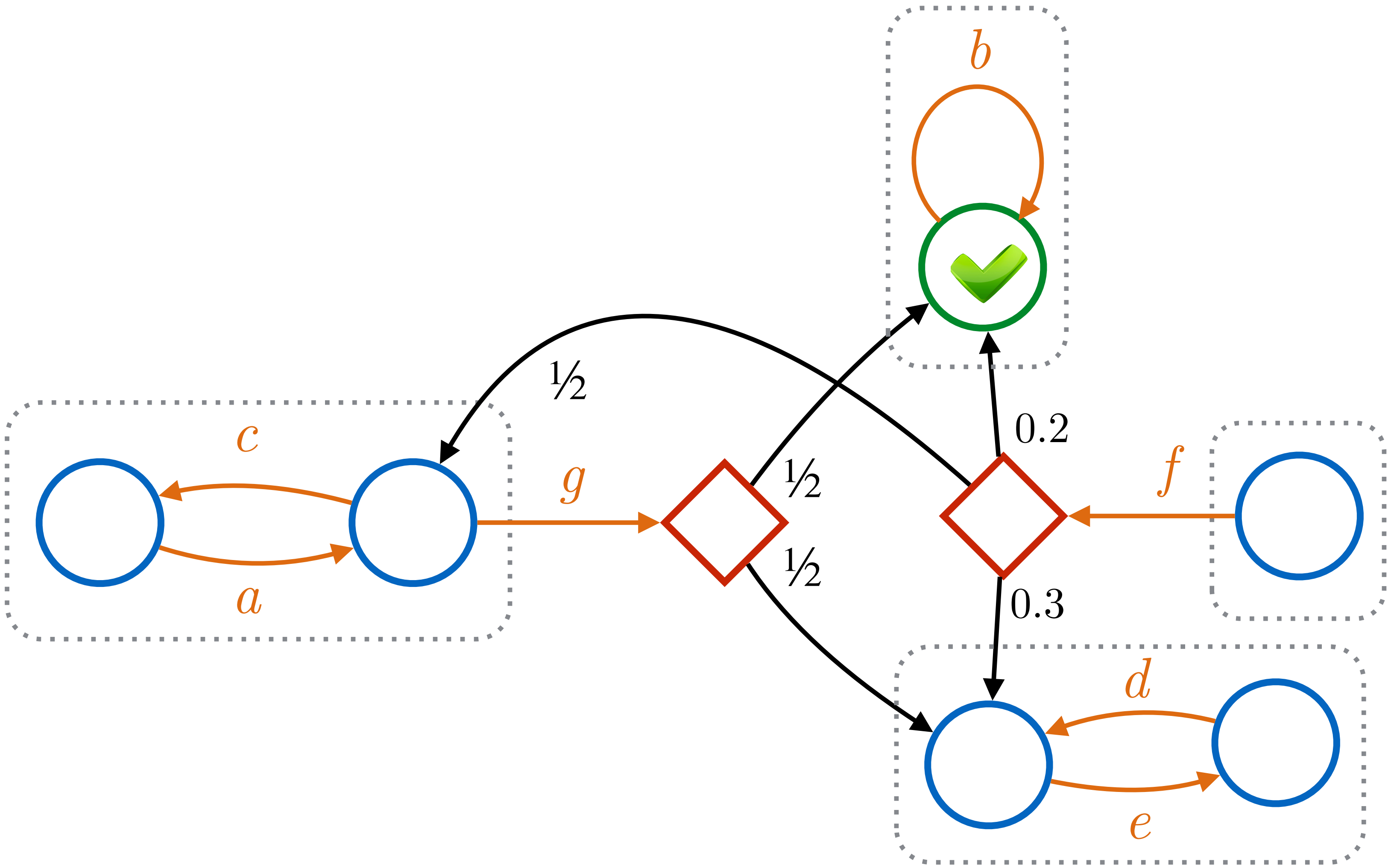
Usual techniques applied for MDPs do not apply…



Still multiple fixed points!

Solution: ensure uniqueness!

Usual techniques applied for MDPs do not apply…

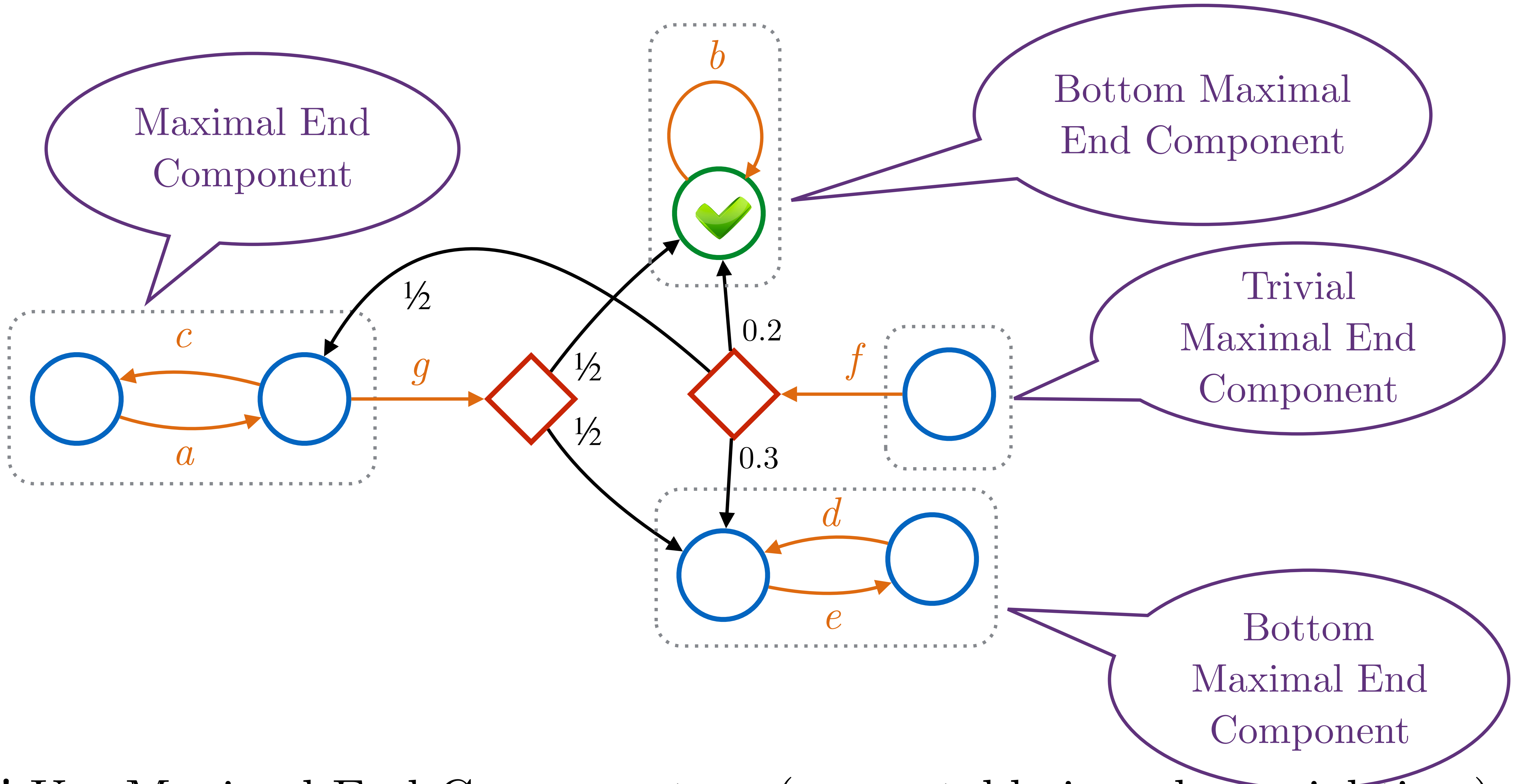


NEW! Use Maximal End Components… (computable in polynomial time)

[de Alfaro, 1997]

Solution: ensure uniqueness!

Usual techniques applied for MDPs do not apply...

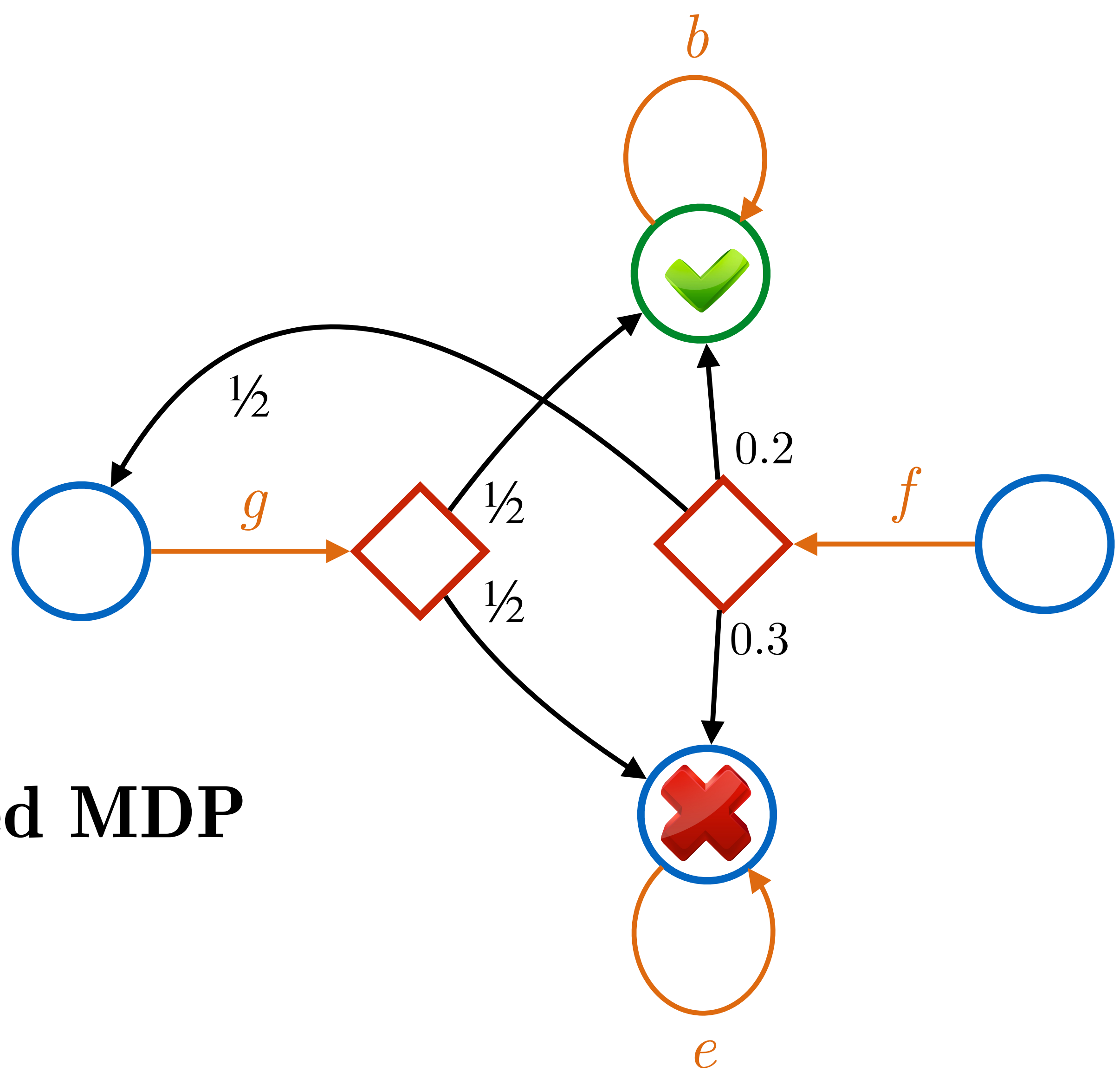


NEW! Use Maximal End Components... (computable in polynomial time)

[de Alfaro, 1997]

Solution: ensure uniqueness!

Usual techniques applied for MDPs do not apply…

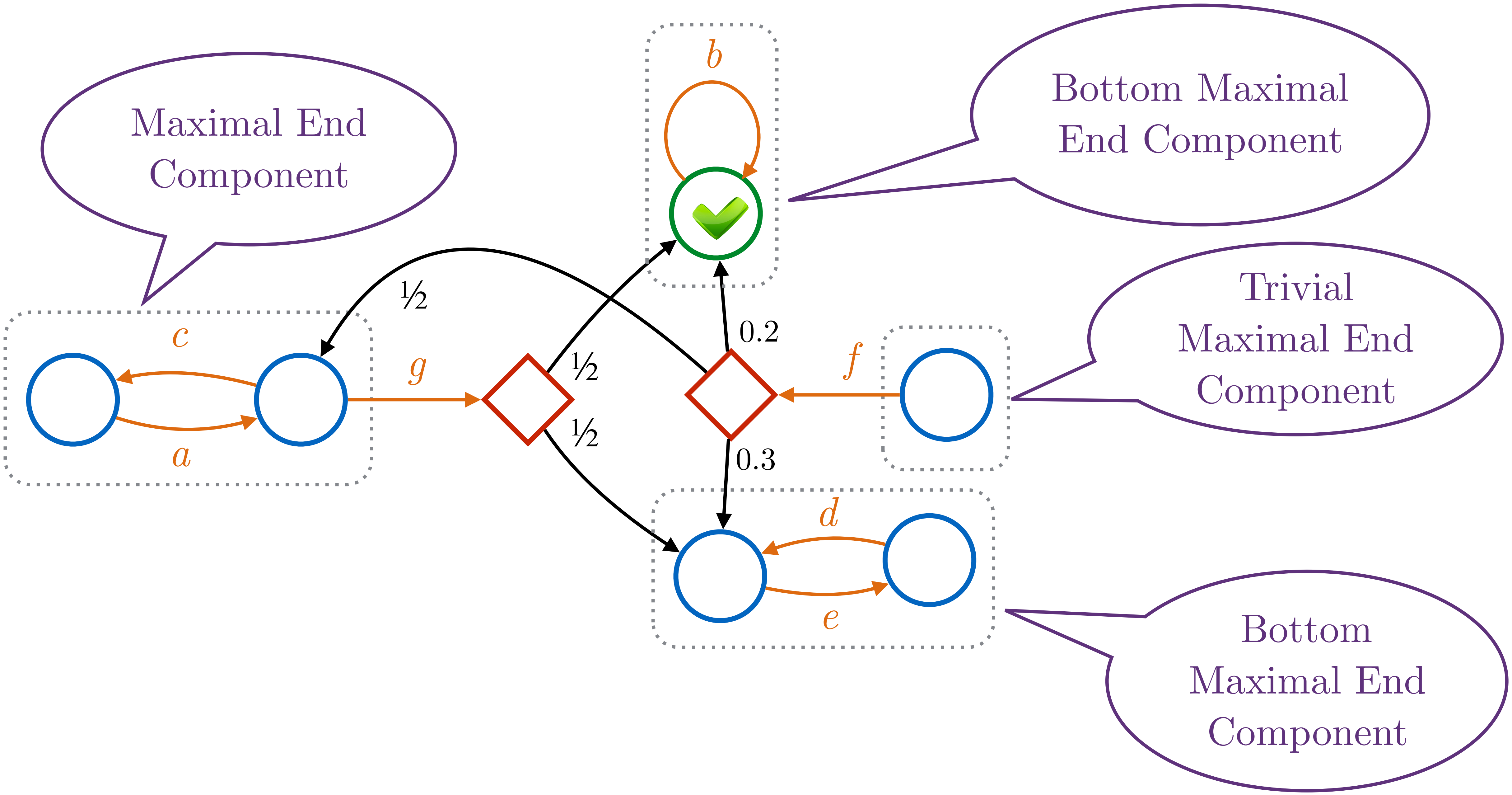


Max-reduced MDP

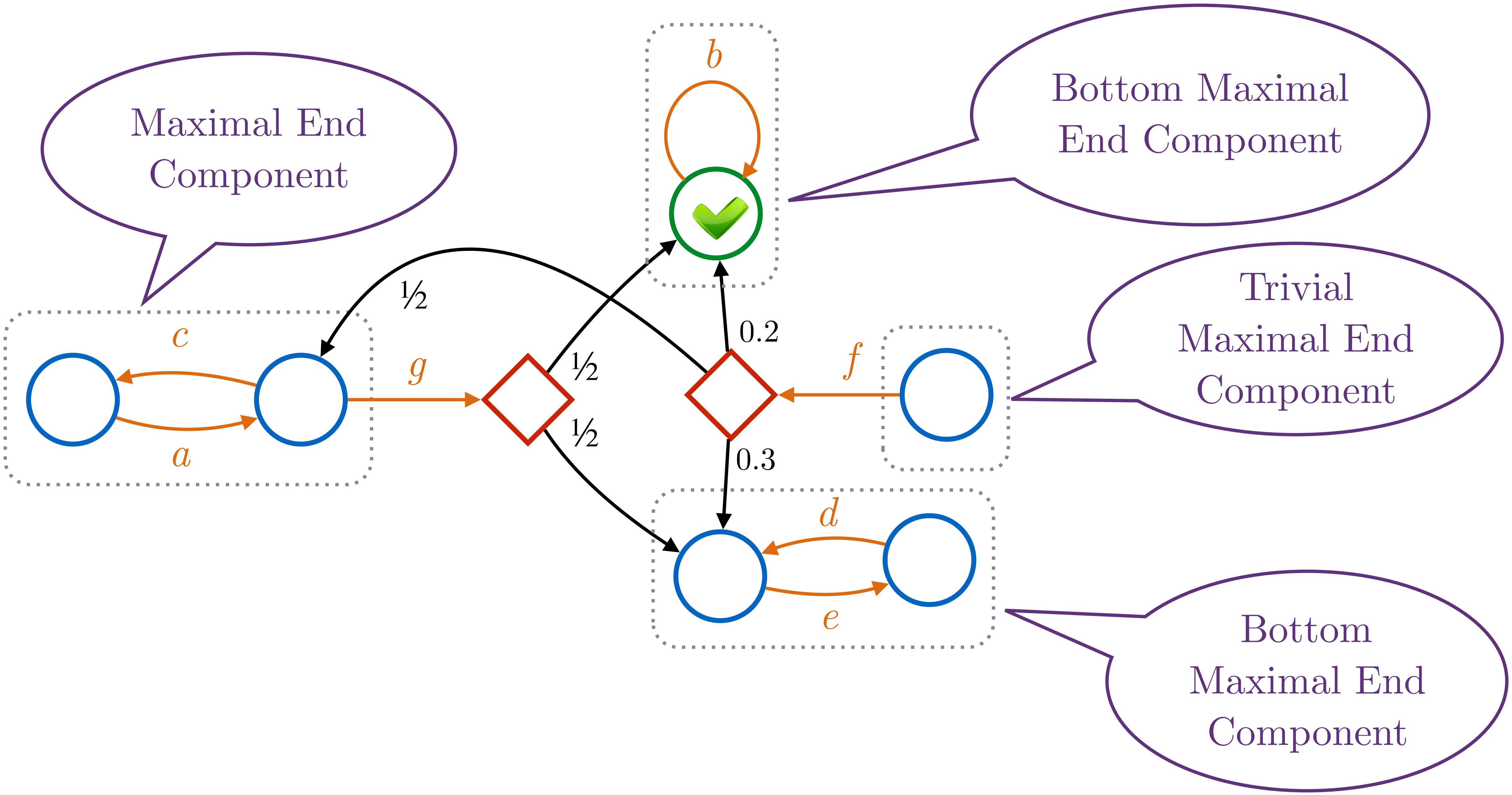
NEW! Use Maximal End Components… (computable in polynomial time)
and trivialize them! Now, unicity of the fixed point

[de Alfaro, 1997]

An even smaller MDP for minimal probabilities

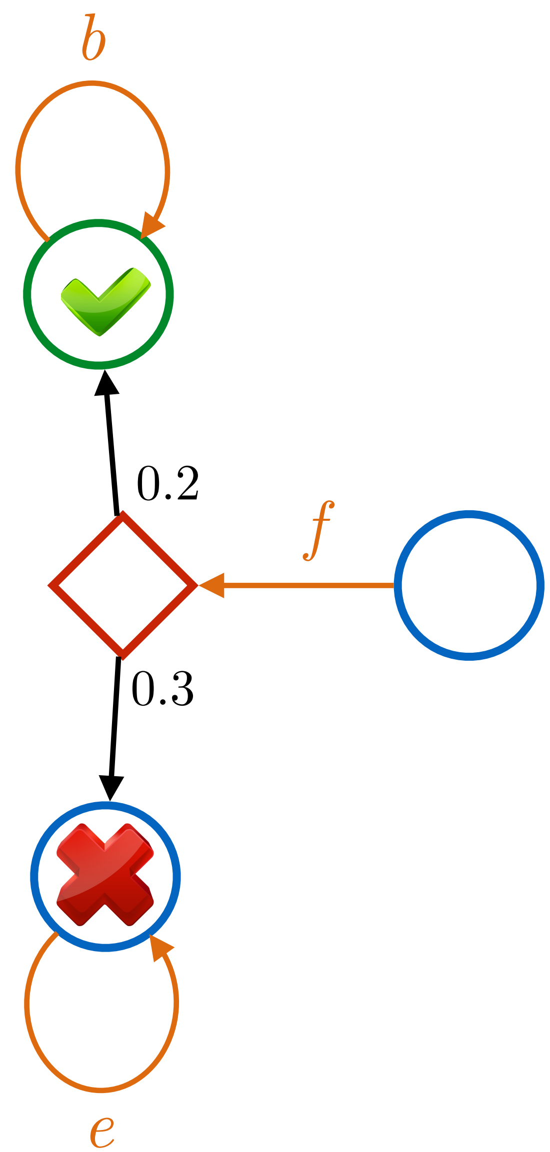


An even smaller MDP for minimal probabilities



Non-trivial (and non accepting) MEC
have null minimal probability!

An even smaller MDP for minimal probabilities



Min-reduced MDP

Non-trivial (and non accepting) MEC
have null minimal probability!

Interval iteration algorithm in reduced MDPs

Input: Min-reduced MDP $\mathcal{M} = (S, \alpha_{\mathcal{M}}, \delta_{\mathcal{M}})$, convergence threshold ε
Output: Under- and over-approximation of $Pr^{\min}_{\mathcal{M}}(\mathsf{F}\,✓)$

$x_{✓} := 1$; $x_{✗} := 0$; $y_{✓} := 1$; $y_{✗} := 0$
foreach $s \in S \setminus \{✓, ✗\}$ **do** $x_s := 0$; $y_s := 1$
repeat
  foreach $s \in S \setminus \{✓, ✗\}$ **do**
    $x'_s := \min_{a \in A(s)} \sum_{s' \in S} \delta_{\mathcal{M}}(s,a)(s')\, x_{s'}$
    $y'_s := \min_{a \in A(s)} \sum_{s' \in S} \delta_{\mathcal{M}}(s,a)(s')\, y_{s'}$
  $\delta := \max_{s \in S}(y'_s - x'_s)$
  foreach $s \in S \setminus \{✓, ✗\}$ **do** $x'_s := x_s$; $y'_s := y_s$
until $\delta \leqslant \varepsilon$
return $(x_s)_{s \in S}, (y_s)_{s \in S}$

Interval iteration algorithm in reduced MDPs

Input: Min-reduced MDP $\mathcal{M} = (S, \alpha_{\mathcal{M}}, \delta_{\mathcal{M}})$, convergence threshold ε

Output: Under- and over-approximation of $Pr^{\min}_{\mathcal{M}}(\mathsf{F}\,✔)$

```
x_✔ := 1; x_✘ := 0; y_✔ := 1; y_✘ := 0
foreach s ∈ S \ {✔, ✘} do x_s := 0; y_s := 1
repeat
    foreach s ∈ S \ {✔, ✘} do
        x'_s := min_{a∈A(s)} Σ_{s'∈S} δ_M(s,a)(s') x_{s'}
        y'_s := min_{a∈A(s)} Σ_{s'∈S} δ_M(s,a)(s') y_{s'}
    δ := max_{s∈S}(y'_s − x'_s)
    foreach s ∈ S \ {✔, ✘} do x'_s := x_s; y'_s := y_s
until δ ⩽ ε
return (x_s)_{s∈S}, (y_s)_{s∈S}
```

Sequences x and y converge towards the minimal probability to reach ✔. Hence, the algorithm terminates by returning an interval of length at most ε for each state containing $\Pr^{\min}_{s}(\mathsf{F}\,✔)$.

Interval iteration algorithm in reduced MDPs

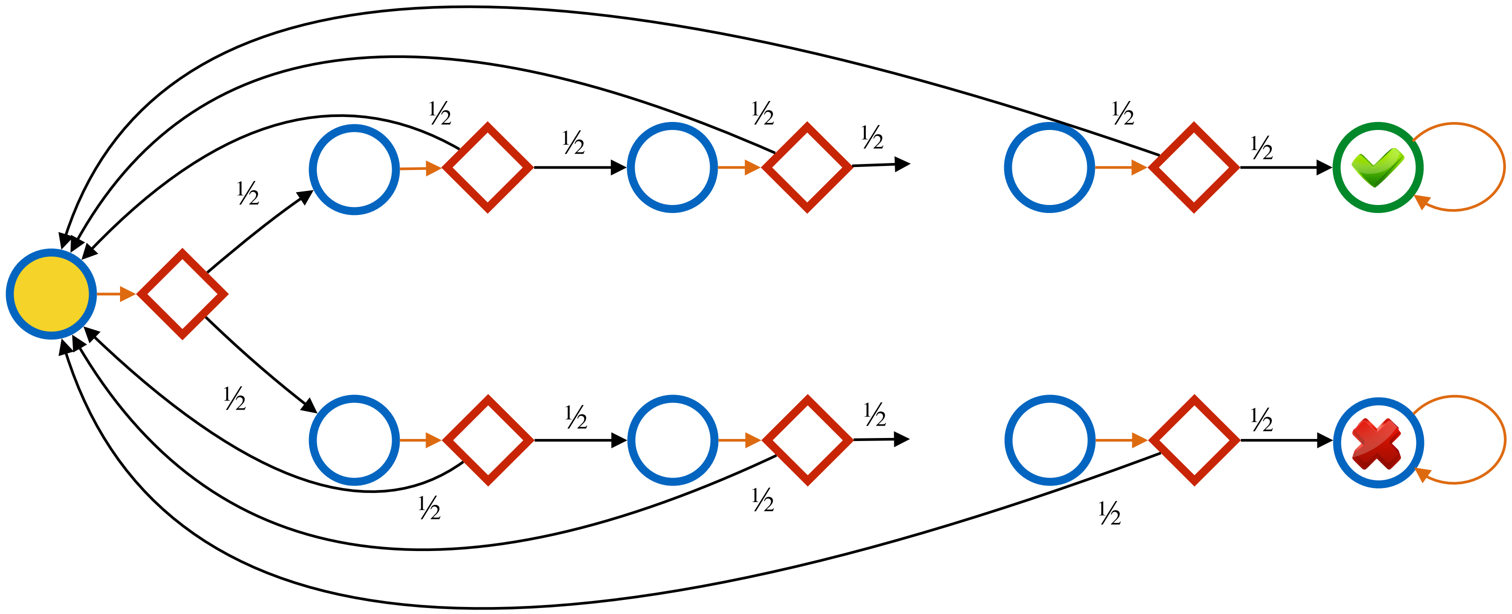
Input: Min-reduced MDP $\mathcal{M} = (S, \alpha_{\mathcal{M}}, \delta_{\mathcal{M}})$, convergence threshold ε
Output: Under- and over-approximation of $Pr^{\min}_{\mathcal{M}}(\mathsf{F}\checkmark)$

$x_{\checkmark} := 1;\ x_{\times} := 0;\ y_{\checkmark} := 1;\ y_{\times} := 0$
foreach $s \in S \setminus \{\checkmark, \times\}$ **do** $x_s := 0;\ y_s := 1$
repeat
  foreach $s \in S \setminus \{\checkmark, \times\}$ **do**
    $x'_s := \min_{a \in A(s)} \sum_{s' \in S} \delta_{\mathcal{M}}(s,a)(s')\, x_{s'}$
    $y'_s := \min_{a \in A(s)} \sum_{s' \in S} \delta_{\mathcal{M}}(s,a)(s')\, y_{s'}$
  $\delta := \max_{s \in S}(y'_s - x'_s)$
  foreach $s \in S \setminus \{\checkmark, \times\}$ **do** $x'_s := x_s;\ y'_s := y_s$
until $\delta \leqslant \varepsilon$
return $(x_s)_{s \in S}, (y_s)_{s \in S}$

Sequences x and y converge towards the minimal probability to reach $\checkmark$. Hence, the algorithm terminates by returning an interval of length at most ε for each state containing $\mathrm{Pr}^{\min}_{s}(\mathsf{F}\checkmark)$.

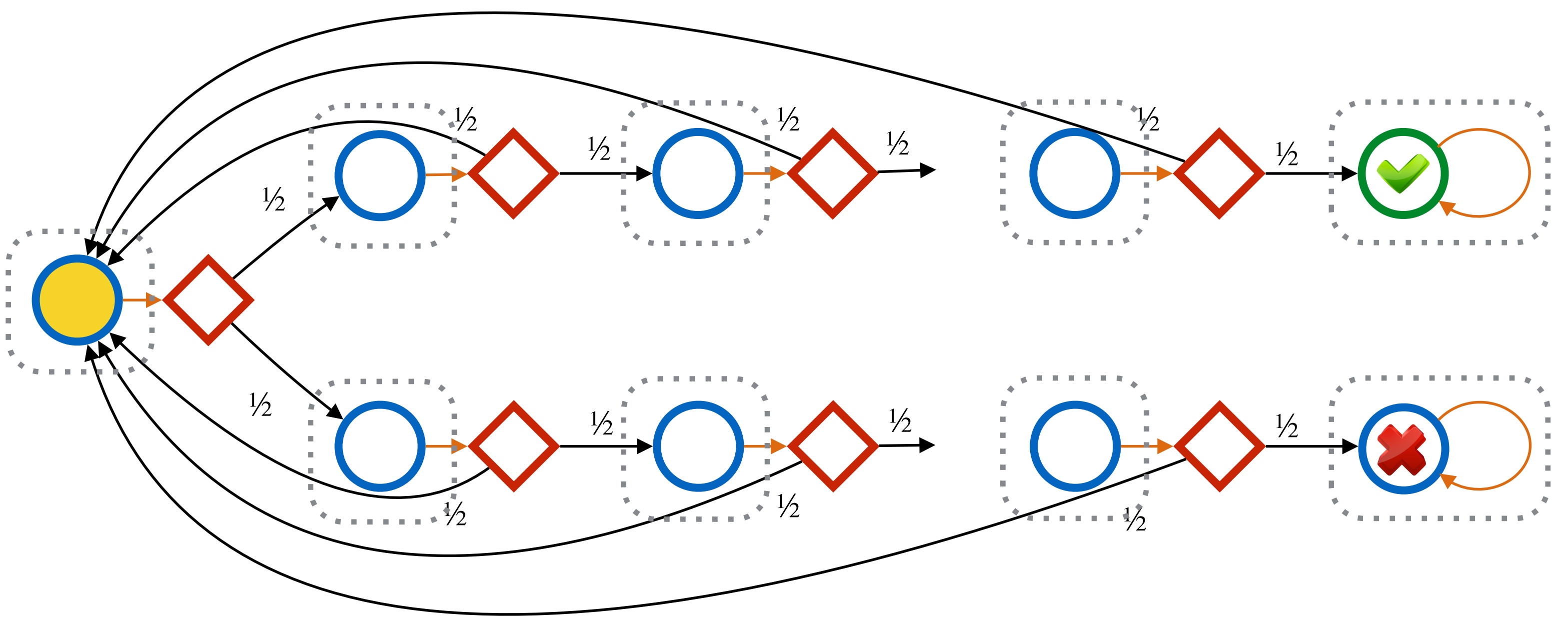
Possible speed-up: only check size of interval for a given state...

Rate of convergence



x stores reachability probabilities, y stores safety probabilities, i.e., after n iterations: $x_s = \Pr_s^{\min}(\mathsf{F}^{\leq n} ✓)$ $y_s = \Pr_s^{\min}(\mathsf{G}^{\leq n}(\neg ✗))$

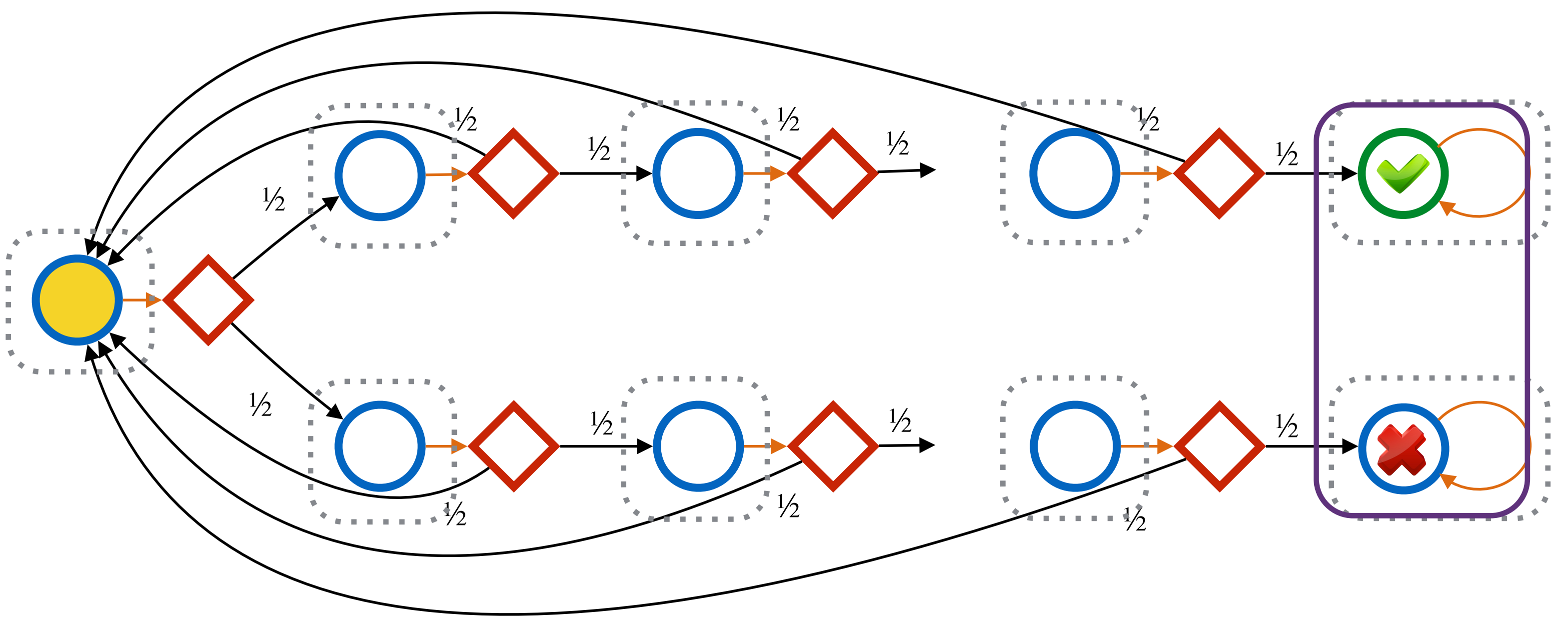
Rate of convergence



2 BMECs and only trivial MECs

x stores reachability probabilities, y stores safety probabilities, i.e., after n iterations: $x_s = \Pr_s^{\min}(\mathsf{F}^{\leq n} ✔)$ $y_s = \Pr_s^{\min}(\mathsf{G}^{\leq n}(\neg ✖))$

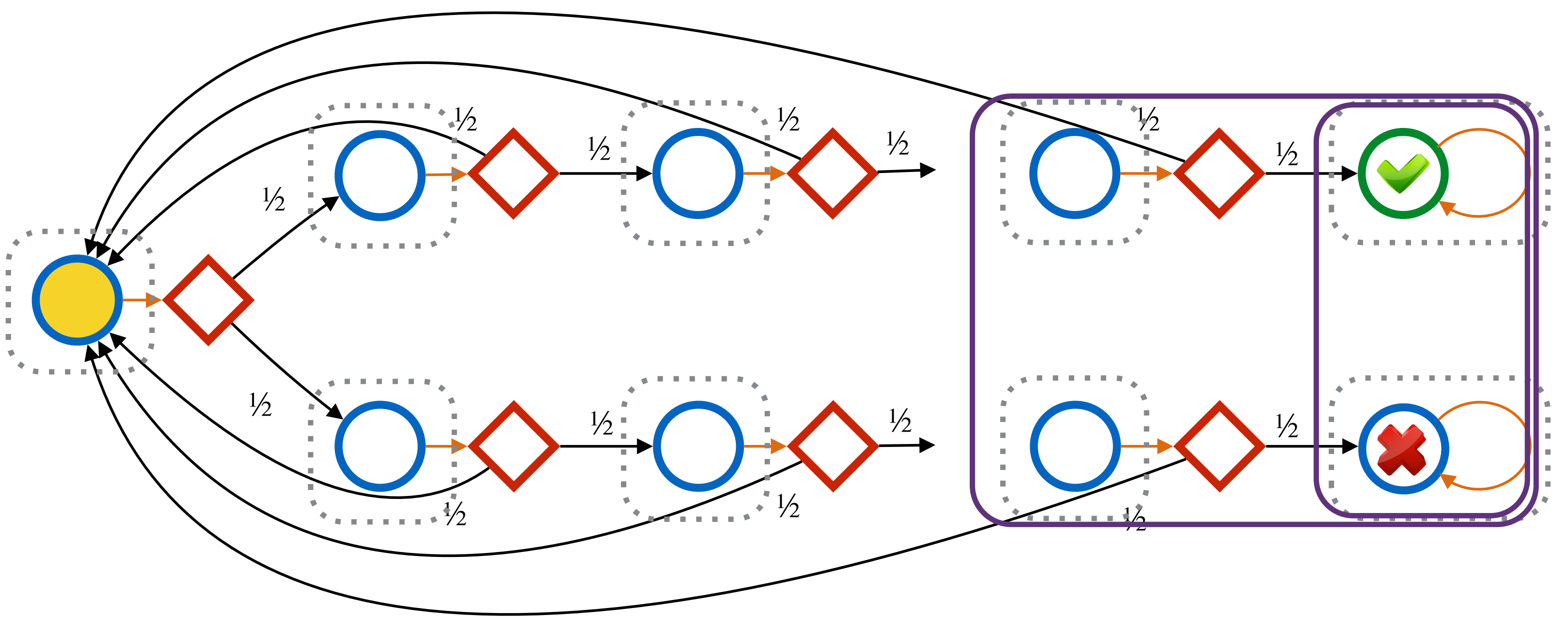
Rate of convergence



2 BMECs and only trivial MECs
attractor decomposition: length I

x stores reachability probabilities, y stores safety probabilities, i.e., after n iterations: $x_s = \Pr_s^{\min}(\mathsf{F}^{\leq n}$ ✓$)$ $y_s = \Pr_s^{\min}(\mathsf{G}^{\leq n}(\neg$ ✗$))$

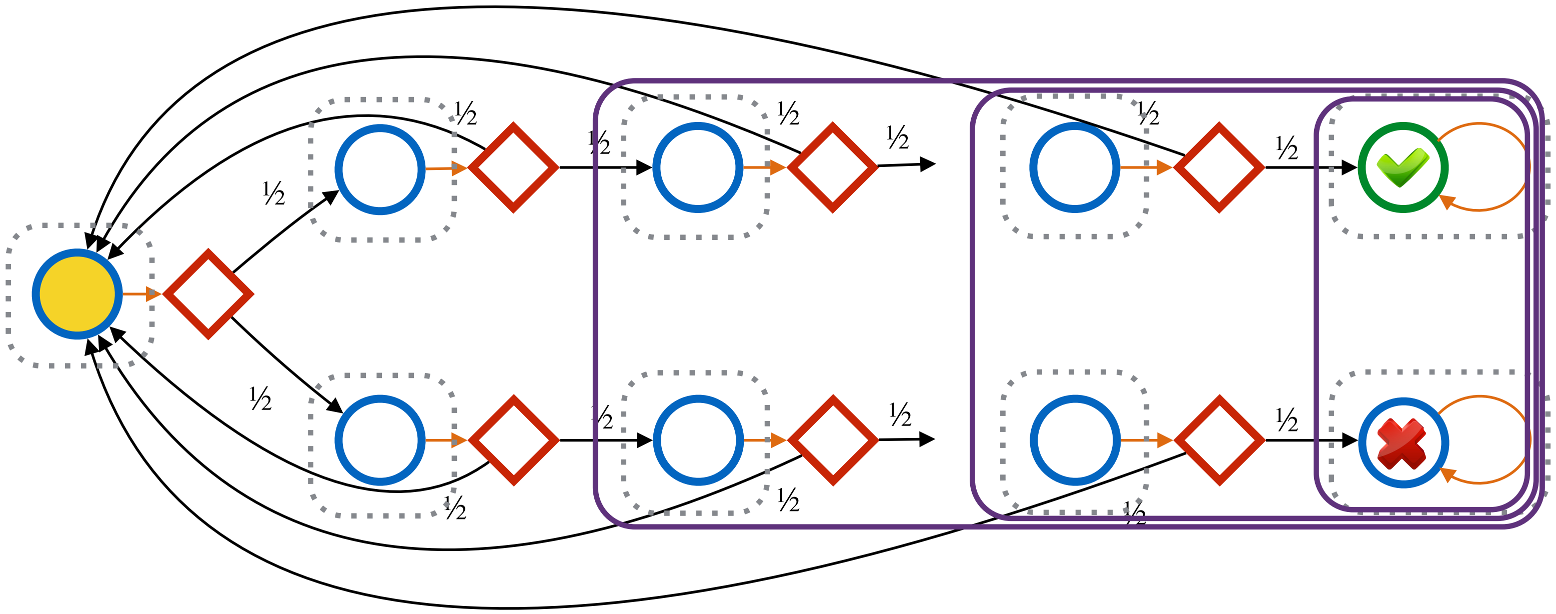
Rate of convergence



2 BMECs and only trivial MECs
attractor decomposition: length I

x stores reachability probabilities, y stores safety probabilities, i.e., after n iterations: $x_s = \Pr_s^{\min}(\mathsf{F}^{\leq n} ✔)$ $\quad y_s = \Pr_s^{\min}(\mathsf{G}^{\leq n}(\neg ✖))$

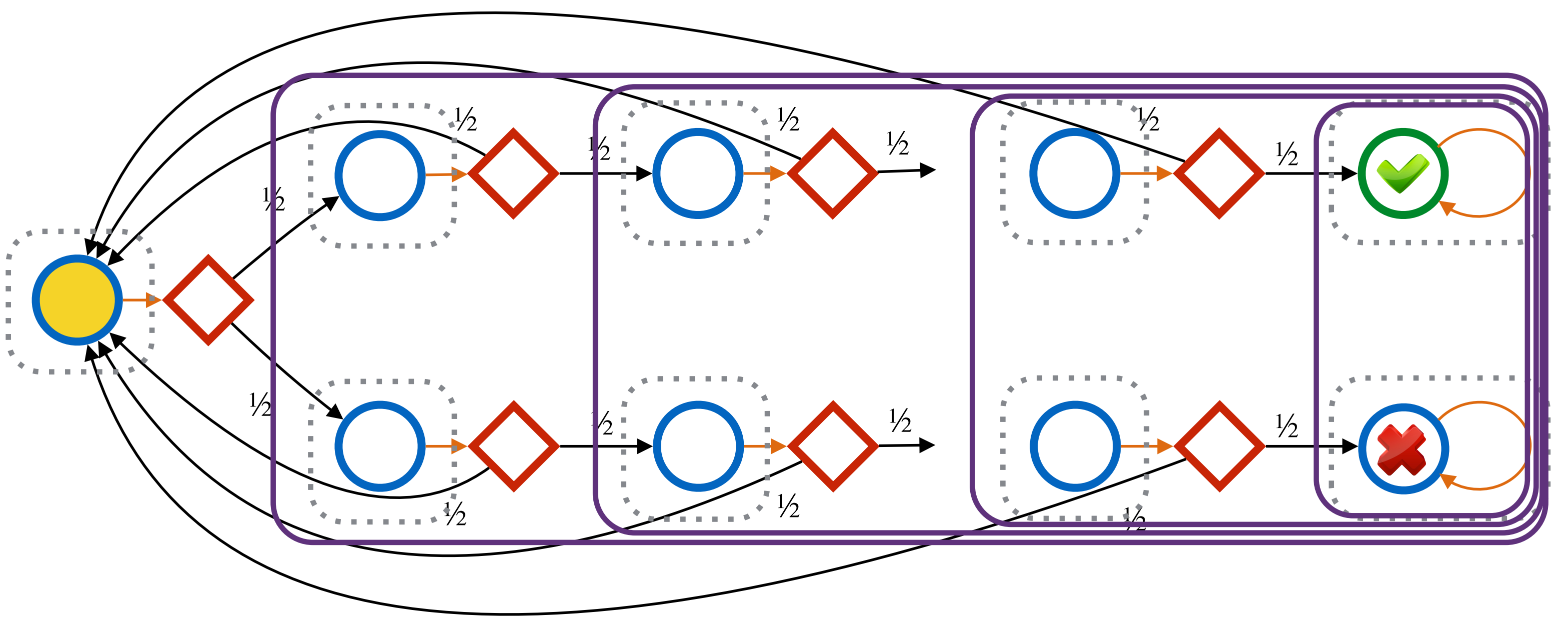
Rate of convergence



2 BMECs and only trivial MECs
attractor decomposition: length I

x stores reachability probabilities, y stores safety probabilities,
i.e., after n iterations: $x_s = \mathrm{Pr}_s^{\min}(\mathsf{F}^{\leq n}$ ✔$)$ $\quad y_s = \mathrm{Pr}_s^{\min}(\mathsf{G}^{\leq n}(\neg$ ✖$))$

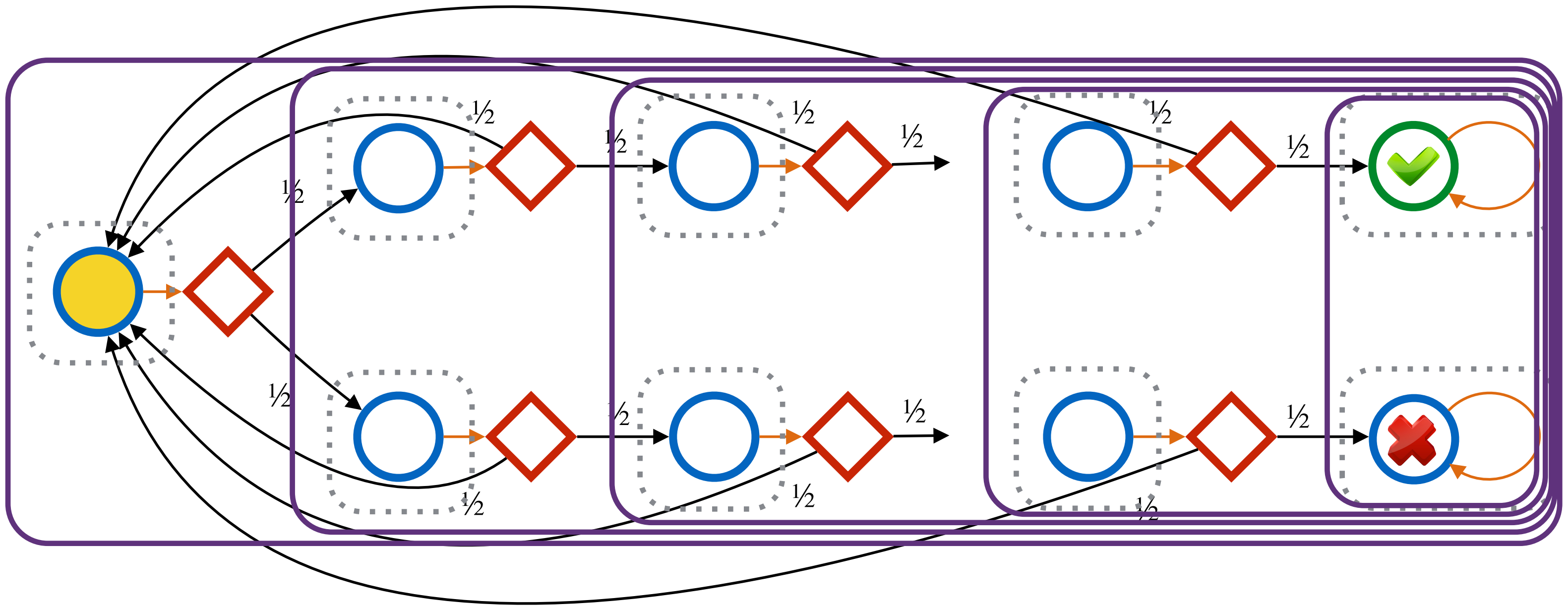
Rate of convergence



2 BMECs and only trivial MECs
attractor decomposition: length I

x stores reachability probabilities, y stores safety probabilities,
i.e., after n iterations: $x_s = \Pr_s^{\min}(\mathsf{F}^{\leq n} ✔)$ $\quad y_s = \Pr_s^{\min}(\mathsf{G}^{\leq n}(\neg ✖))$

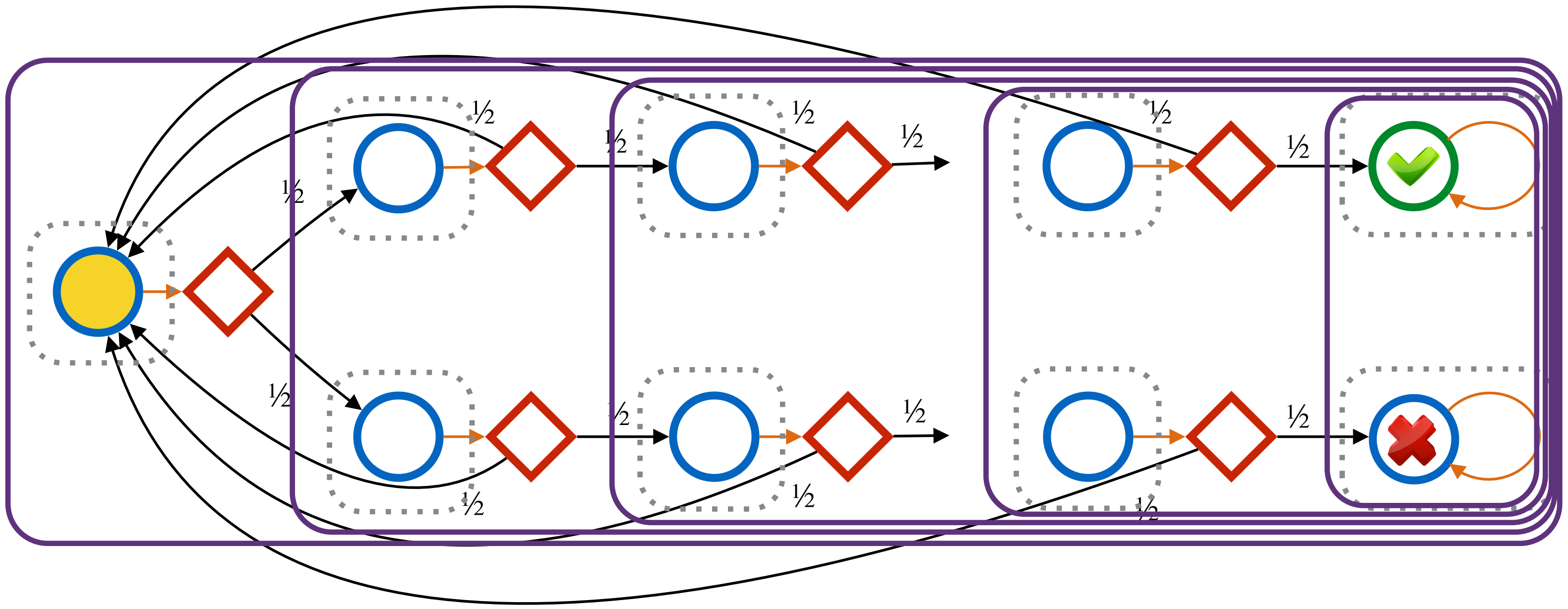
Rate of convergence



2 BMECs and only trivial MECs
attractor decomposition: length I

x stores reachability probabilities, y stores safety probabilities,
i.e., after n iterations: $x_s = \Pr_s^{\min}(\mathsf{F}^{\leq n}\,✔)$ $\quad y_s = \Pr_s^{\min}(\mathsf{G}^{\leq n}(\neg\,✖))$

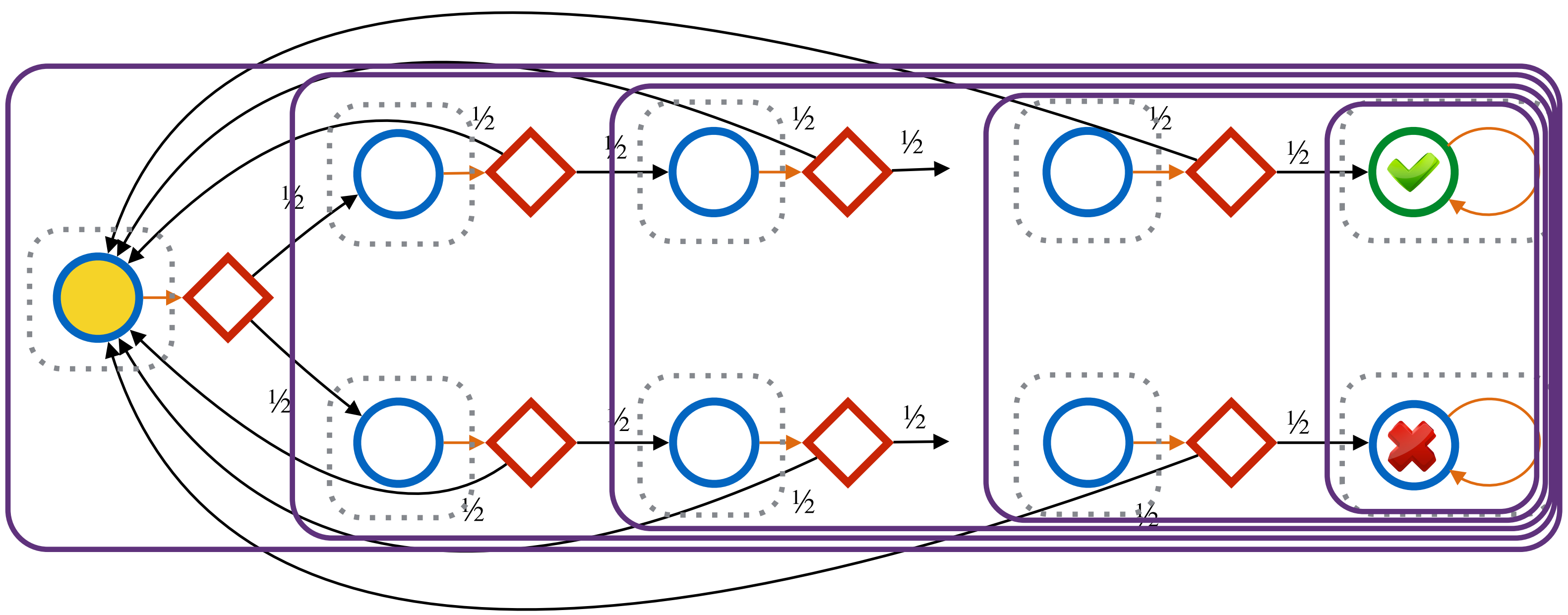
Rate of convergence



2 BMECs and only trivial MECs
attractor decomposition: length I
smallest positive probability: η

x stores reachability probabilities, y stores safety probabilities,
i.e., after n iterations: $x_s = \Pr_s^{\min}(\mathsf{F}^{\leq n}$ ✔$)$ $y_s = \Pr_s^{\min}(\mathsf{G}^{\leq n}(\neg$ ✖$))$

Rate of convergence

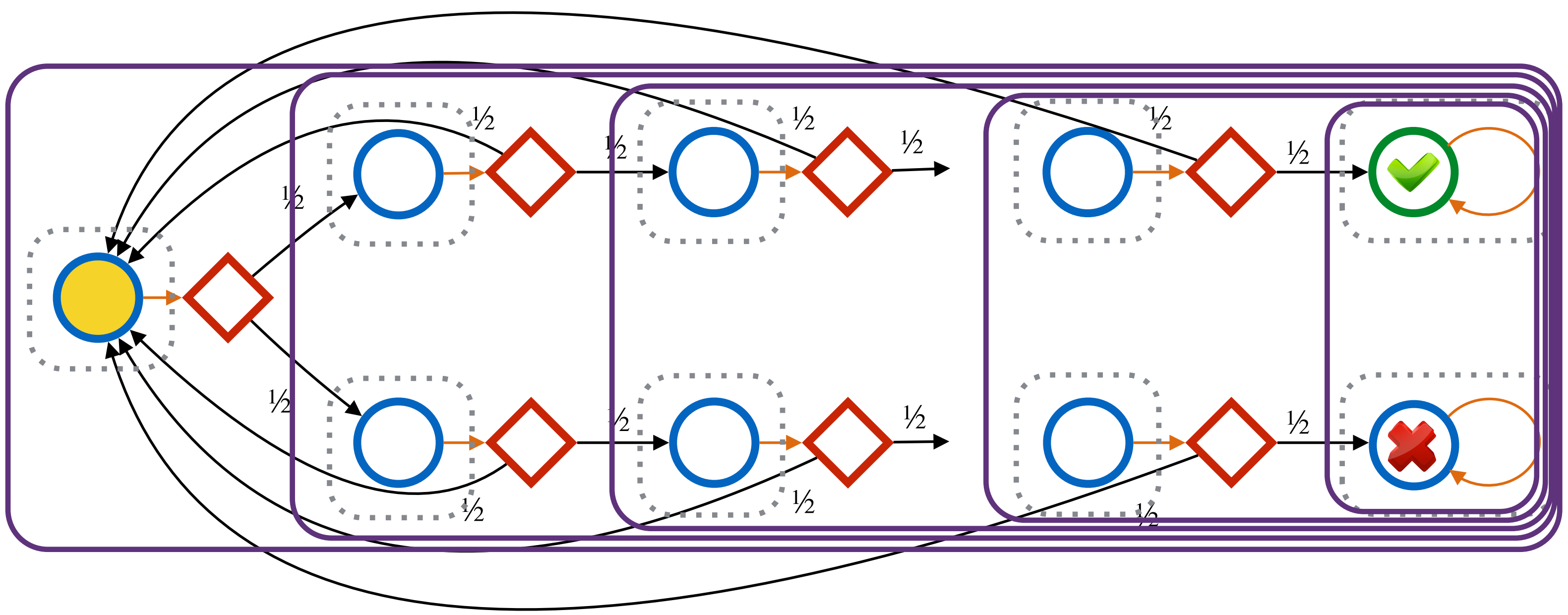


2 BMECs and only trivial MECs
attractor decomposition: length I
smallest positive probability: η

x stores reachability probabilities, y stores safety probabilities,
i.e., after n iterations: $x_s = \Pr_s^{\min}(\mathsf{F}^{\leq n} \checkmark)$ $\quad y_s = \Pr_s^{\min}(\mathsf{G}^{\leq n}(\neg \times))$

Leaking property: $\forall n \in \mathbb{N} \quad \Pr_s^{\max}(\mathsf{G}^{\leq nI} \neg(\checkmark \vee \times)) \leq (1-\eta^I)^n$

Rate of convergence



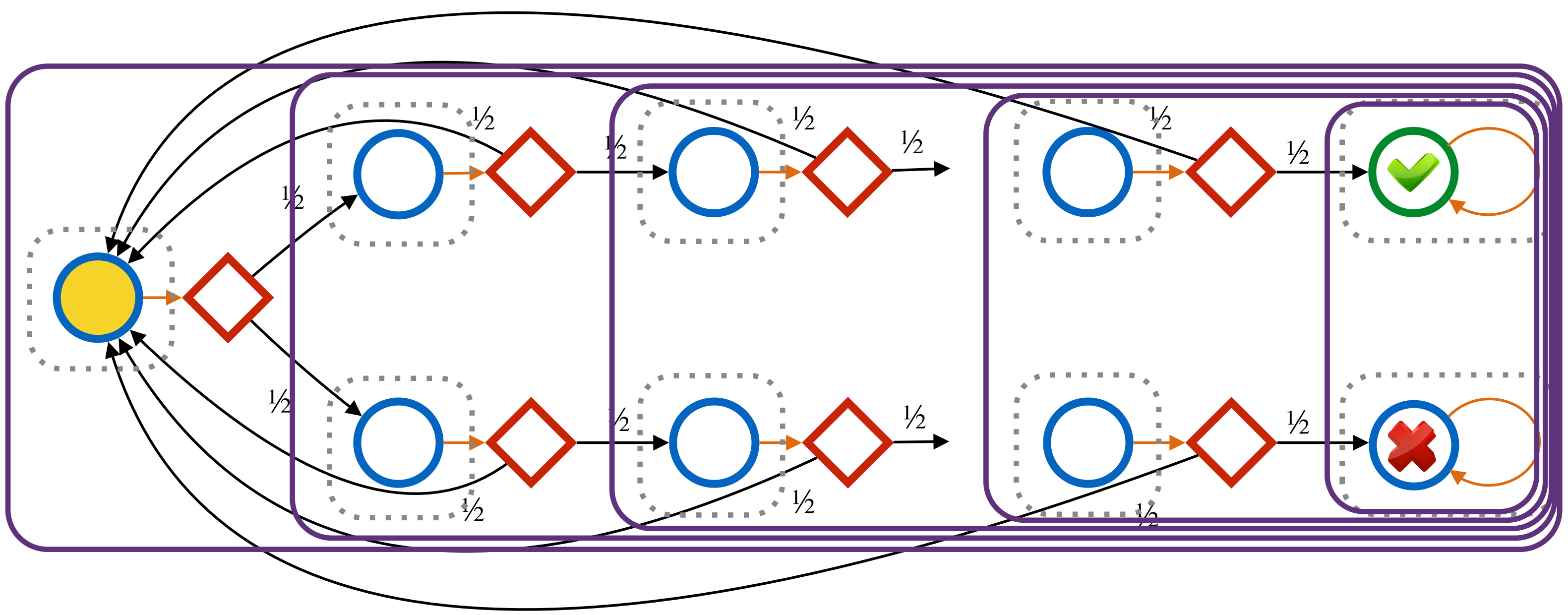
2 BMECs and only trivial MECs
attractor decomposition: length I
smallest positive probability: η

x stores reachability probabilities, y stores safety probabilities, i.e., after n iterations: $x_s = \Pr_s^{\min}(\mathsf{F}^{\leq n}$ ✔$)$ $\quad y_s = \Pr_s^{\min}(\mathsf{G}^{\leq n}(\neg$ ✖$))$

Leaking property: $\forall n \in \mathbb{N} \quad \Pr_s^{\max}(\mathsf{G}^{\leq nI} \neg($✔$\vee$✖$)) \leq (1-\eta^I)^n$

$y_s^{(nI)} - x_s^{(nI)} = \Pr_s^{\sigma}(\mathsf{G}^{\leq nI}(\neg$ ✖$)) - \Pr_s^{\sigma'}(\mathsf{F}^{\leq nI}$ ✔$) \leq \Pr_s^{\sigma'}(\mathsf{G}^{\leq nI}(\neg$ ✖$)) - \Pr_s^{\sigma'}(\mathsf{F}^{\leq nI}$ ✔$)$

Rate of convergence



2 BMECs and only trivial MECs
attractor decomposition: length I
smallest positive probability: η

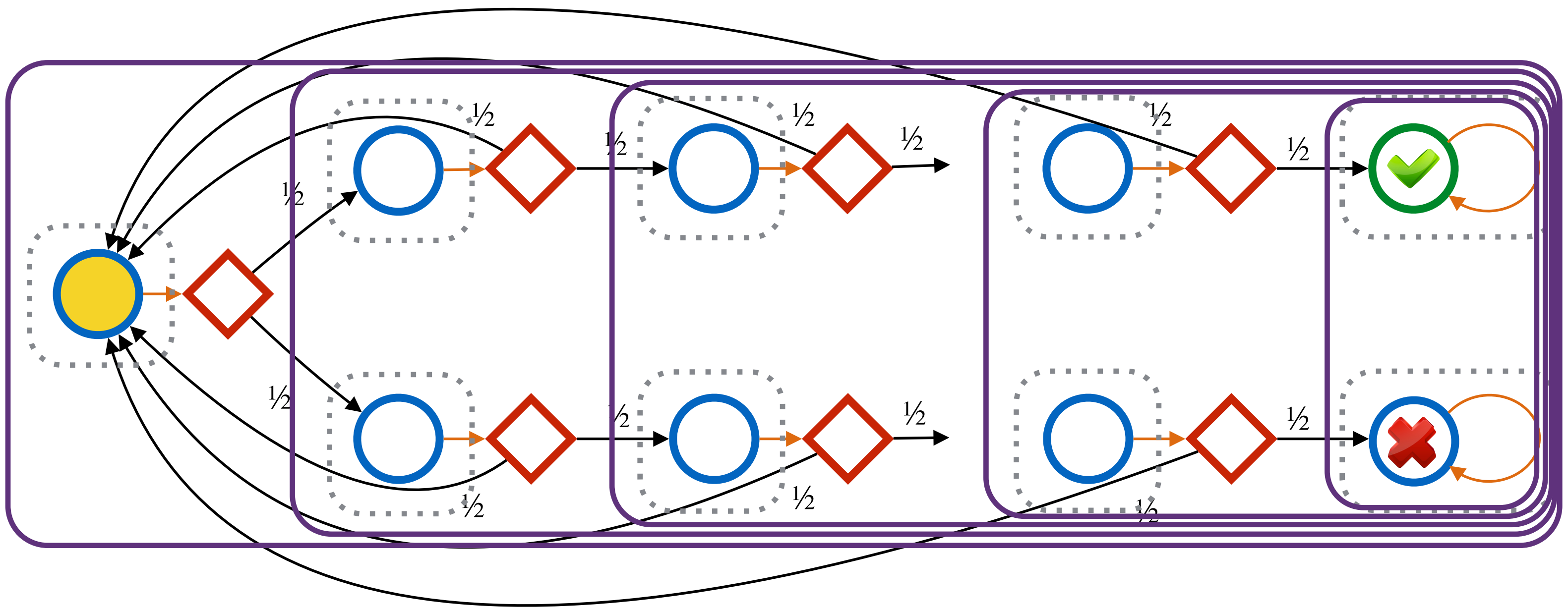
x stores reachability probabilities, y stores safety probabilities,
i.e., after n iterations: $x_s = \Pr_s^{\min}(\mathsf{F}^{\leq n} \checkmark) \quad y_s = \Pr_s^{\min}(\mathsf{G}^{\leq n}(\neg \times))$

Leaking property: $\forall n \in \mathbb{N} \quad \Pr_s^{\max}(\mathsf{G}^{\leq nI} \neg(\checkmark \vee \times)) \leq (1-\eta^I)^n$

$$y_s^{(nI)} - x_s^{(nI)} = \Pr_s^{\sigma}(\mathsf{G}^{\leq nI}(\neg \times)) - \Pr_s^{\sigma'}(\mathsf{F}^{\leq nI} \checkmark) \leq \Pr_s^{\sigma'}(\mathsf{G}^{\leq nI}(\neg \times)) - \Pr_s^{\sigma'}(\mathsf{F}^{\leq nI} \checkmark)$$
$$= \Pr_s^{\sigma'}(\mathsf{G}^{\leq nI} \neg(\checkmark \vee \times)) \leq (1-\eta^I)^n$$

since $\mathsf{G}^{\leq n}(\neg \times) \equiv \mathsf{G}^{\leq n} \neg(\checkmark \vee \times) \oplus \mathsf{F}^{\leq n} \checkmark$

Rate of convergence



2 BMECs and only trivial MECs
attractor decomposition: length I
smallest positive probability: η

x stores reachability probabilities, y stores safety probabilities, i.e., after n iterations: $x_s = \Pr_s^{\min}(\mathsf{F}^{\leq n} ✔)$ $y_s = \Pr_s^{\min}(\mathsf{G}^{\leq n}(\neg ✖))$

Leaking property: $\forall n \in \mathbb{N} \quad \Pr_s^{\max}(\mathsf{G}^{\leq nI}\neg(✔ \vee ✖)) \leq (1-\eta^I)^n$

The interval iteration algorithm converges in at most $I\left\lceil \frac{\log \varepsilon}{\log(1-\eta^I)} \right\rceil$ steps.

Stopping criterion for exact computation

MDPs with rational probabilities:

- d the largest denominator of transition probabilities
- N the number of states
- M the number of transitions with non-zero probabilities

Stopping criterion for exact computation

MDPs with rational probabilities:

d the largest denominator of transition probabilities

N the number of states

M the number of transitions with non-zero probabilities

[Chatterjee, Henzinger 2008] claim for exact computation possible after d^{8M} iterations of value iteration

Stopping criterion for exact computation

MDPs with rational probabilities:

d the largest denominator of transition probabilities

N the number of states

M the number of transitions with non-zero probabilities

[Chatterjee, Henzinger 2008] claim for exact computation possible after d^{8M} iterations of value iteration

Optimal probabilities and policies can be computed by the interval iteration algorithm in at most $8N^3 \left\lceil (1/\eta)^N \log_2 d \right\rceil$ steps.

Stopping criterion for exact computation

MDPs with rational probabilities:

d the largest denominator of transition probabilities

N the number of states

M the number of transitions with non-zero probabilities

[Chatterjee, Henzinger 2008] claim for exact computation possible after d^{8M} iterations of value iteration

Optimal probabilities and policies can be computed by the interval iteration algorithm in at most $8N^3 \lceil (1/\eta)^N \log_2 d \rceil$ steps.

Improvement since

$1/\eta \leq d \qquad N \leq M$

Stopping criterion for exact computation

MDPs with rational probabilities:

d the largest denominator of transition probabilities

N the number of states

M the number of transitions with non-zero probabilities

[Chatterjee, Henzinger 2008] claim for exact computation possible after d^{8M} iterations of value iteration

> Optimal probabilities and policies can be computed by the interval iteration algorithm in at most $8N^3\left\lceil (1/\eta)^N \log_2 d \right\rceil$ steps.

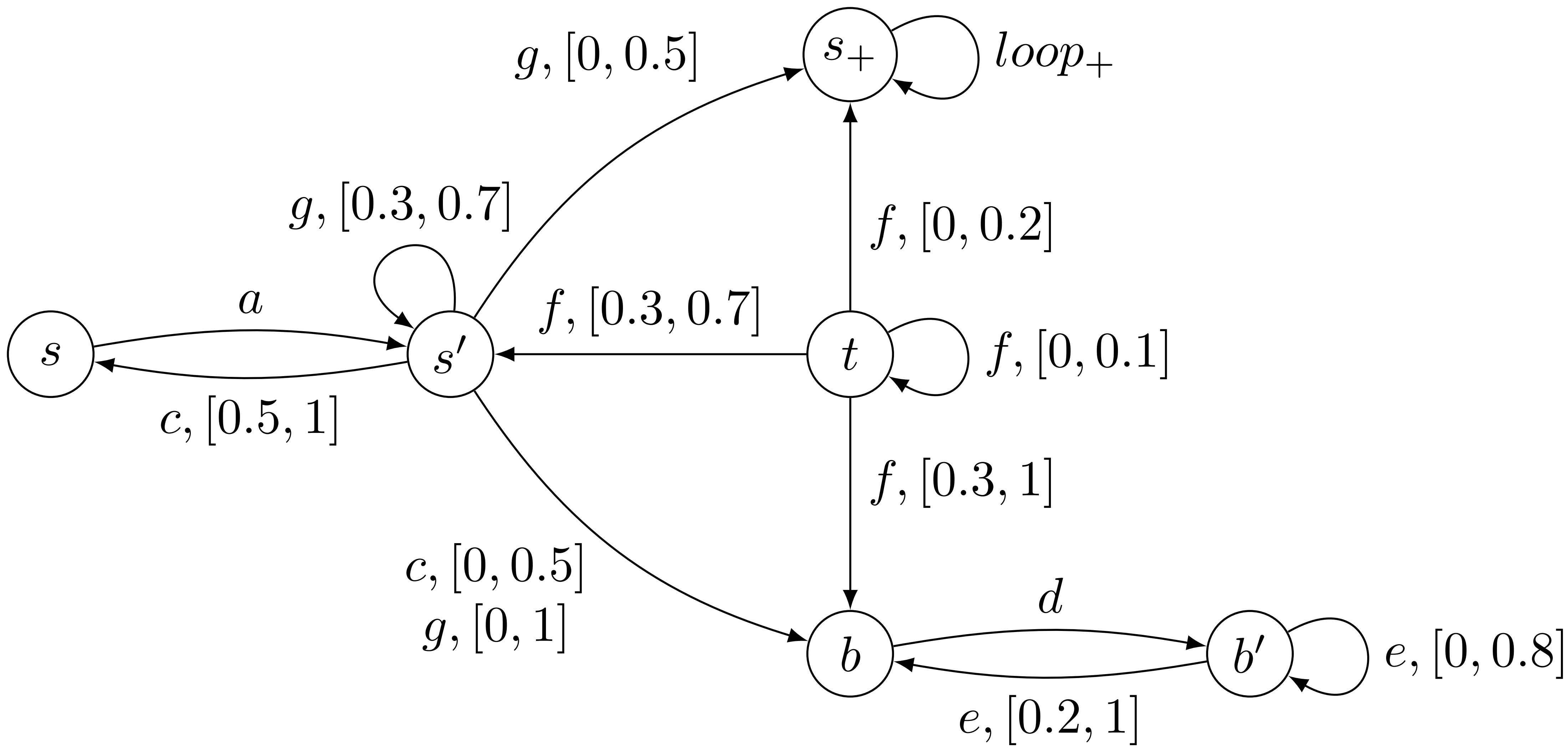
Improvement since

$1/\eta \leq d \qquad N \leq M$

Sketch of proof:

- use $\varepsilon = 1/2\alpha$ as threshold (with α gcd of optimal probabilities)
- upper bound on α based on matrix properties of Markov chains: $\alpha = \mathcal{O}(N^N d^{2N^2})$

Interval MDPs



$\mathcal{M} = (S, \alpha, \breve{\delta}, \widehat{\delta})$

$\delta : S \times \alpha \to [0,1]^S$

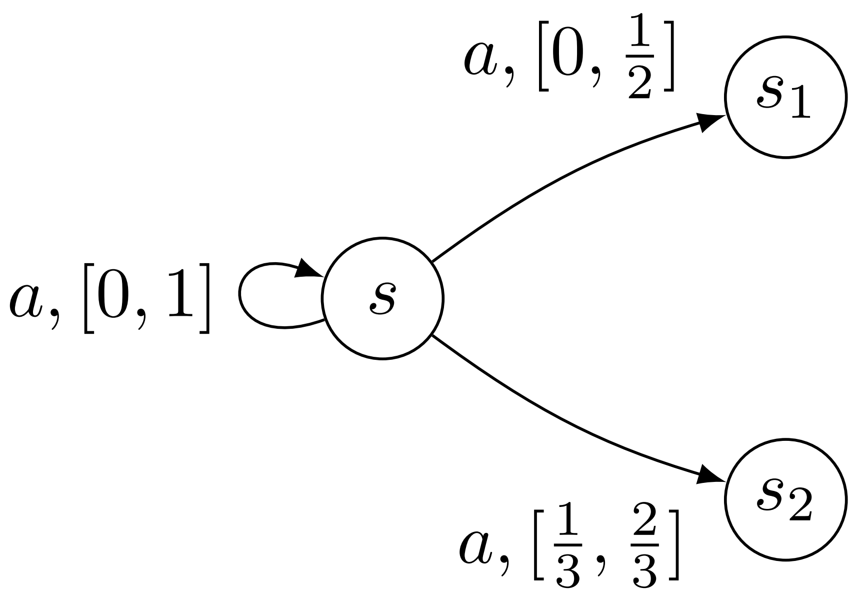
Policy $\sigma : (S \cdot \alpha)^\star \cdot S \to Dist(\alpha) \times (Dist(S))^\alpha$

IMDP vs MDP

- IMDPs = extension of MDPs with an infinite (uncountable) set of actions
- But, behaviours of IMDPs can be captured by MDPs

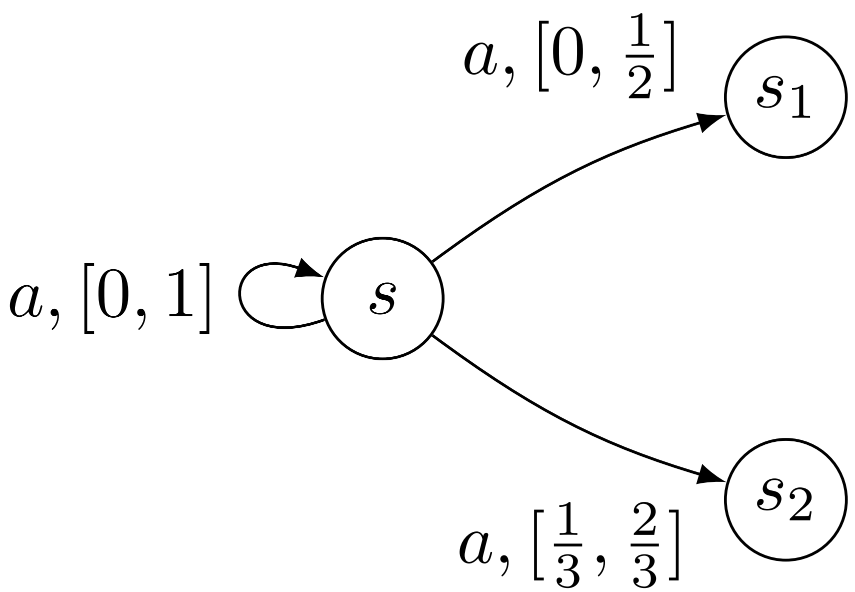
IMDP vs MDP

- IMDPs = extension of MDPs with an infinite (uncountable) set of actions
- But, behaviours of IMDPs can be captured by MDPs



IMDP vs MDP

- IMDPs = extension of MDPs with an infinite (uncountable) set of actions
- But, behaviours of IMDPs can be captured by MDPs



Possible distributions:

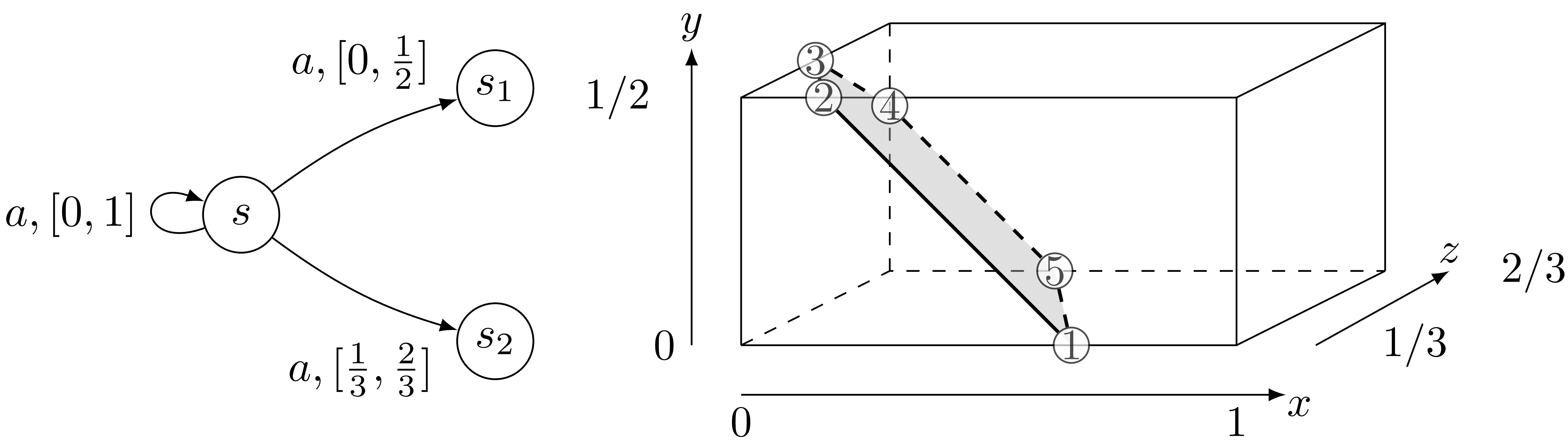
$$p \in Dist(S) \text{ such that } \sum_{s' \in S} p(s') = 1$$

$$\text{and } \forall s' \; \breve{\delta}(s' \mid s, a) \leq p(s') \leq \widehat{\delta}(s' \mid s, a)$$

Solutions of a (bounded) linear program!

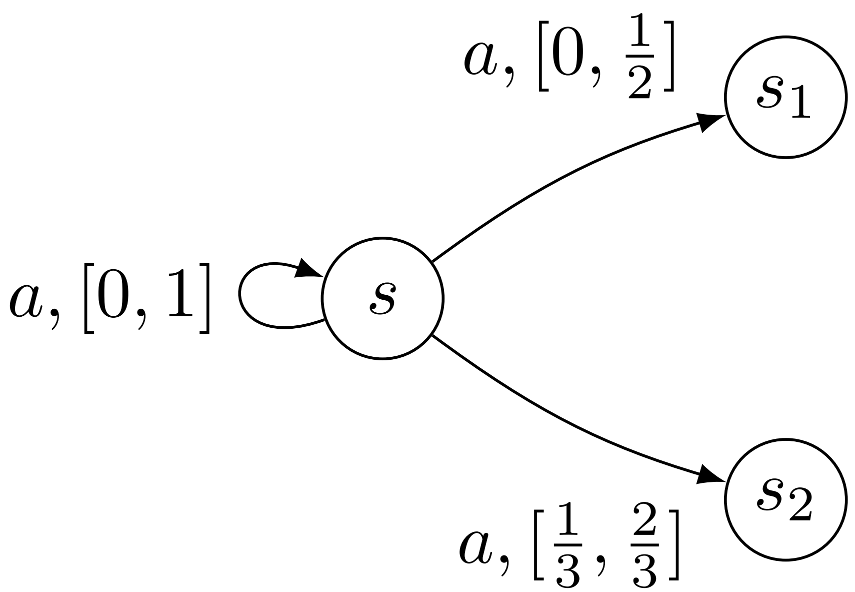
IMDP vs MDP

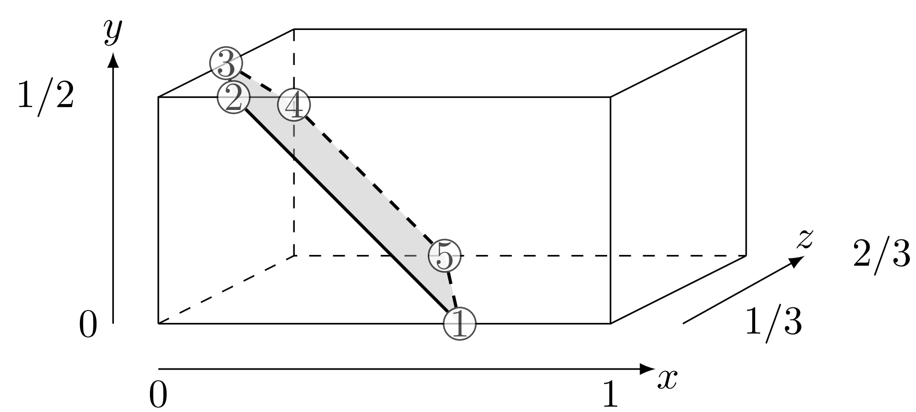
- IMDPs = extension of MDPs with an infinite (uncountable) set of actions
- But, behaviours of IMDPs can be captured by MDPs



IMDP vs MDP

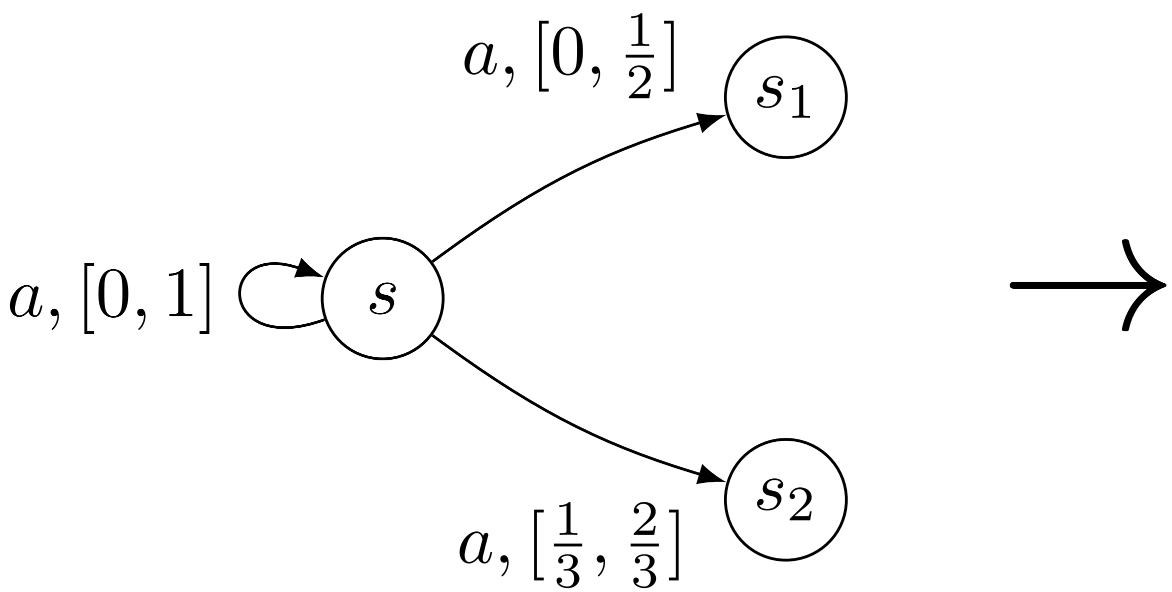
- IMDPs = extension of MDPs with an infinite (uncountable) set of actions
- But, behaviours of IMDPs can be captured by MDPs

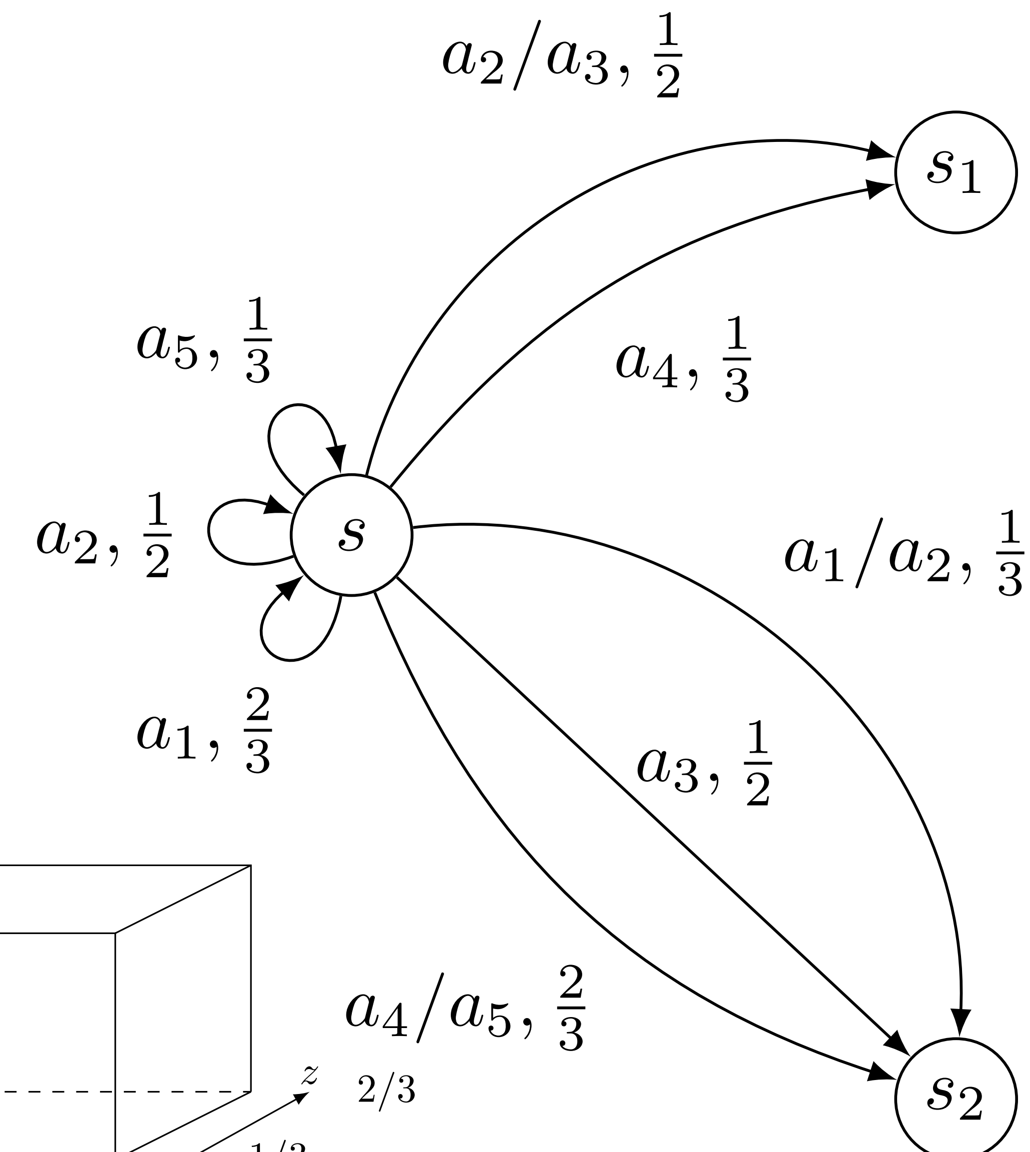


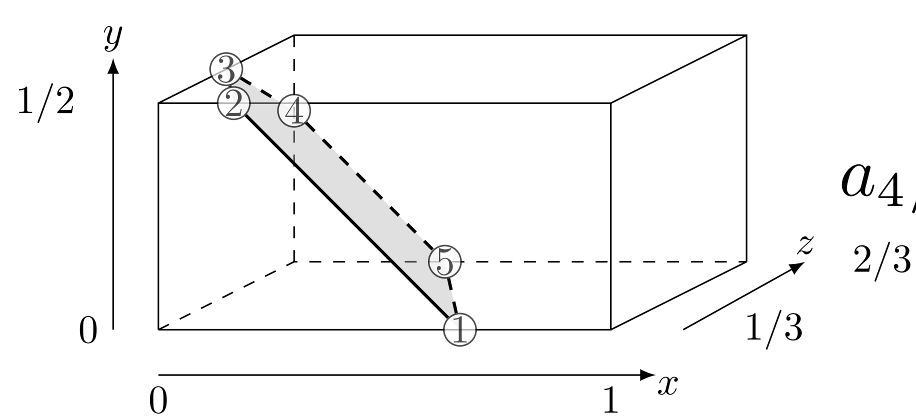


IMDP vs MDP

- IMDPs = extension of MDPs with an infinite (uncountable) set of actions
- But, behaviours of IMDPs can be captured by MDPs







Value iteration for IMDPs

- Simulate on the IMDP the value iteration on its MDP...
- One step is the application of
- Achievable in polynomial time by sorting x...

[Sen, Viswanathan, Agha, 2006]

Value iteration for IMDPs

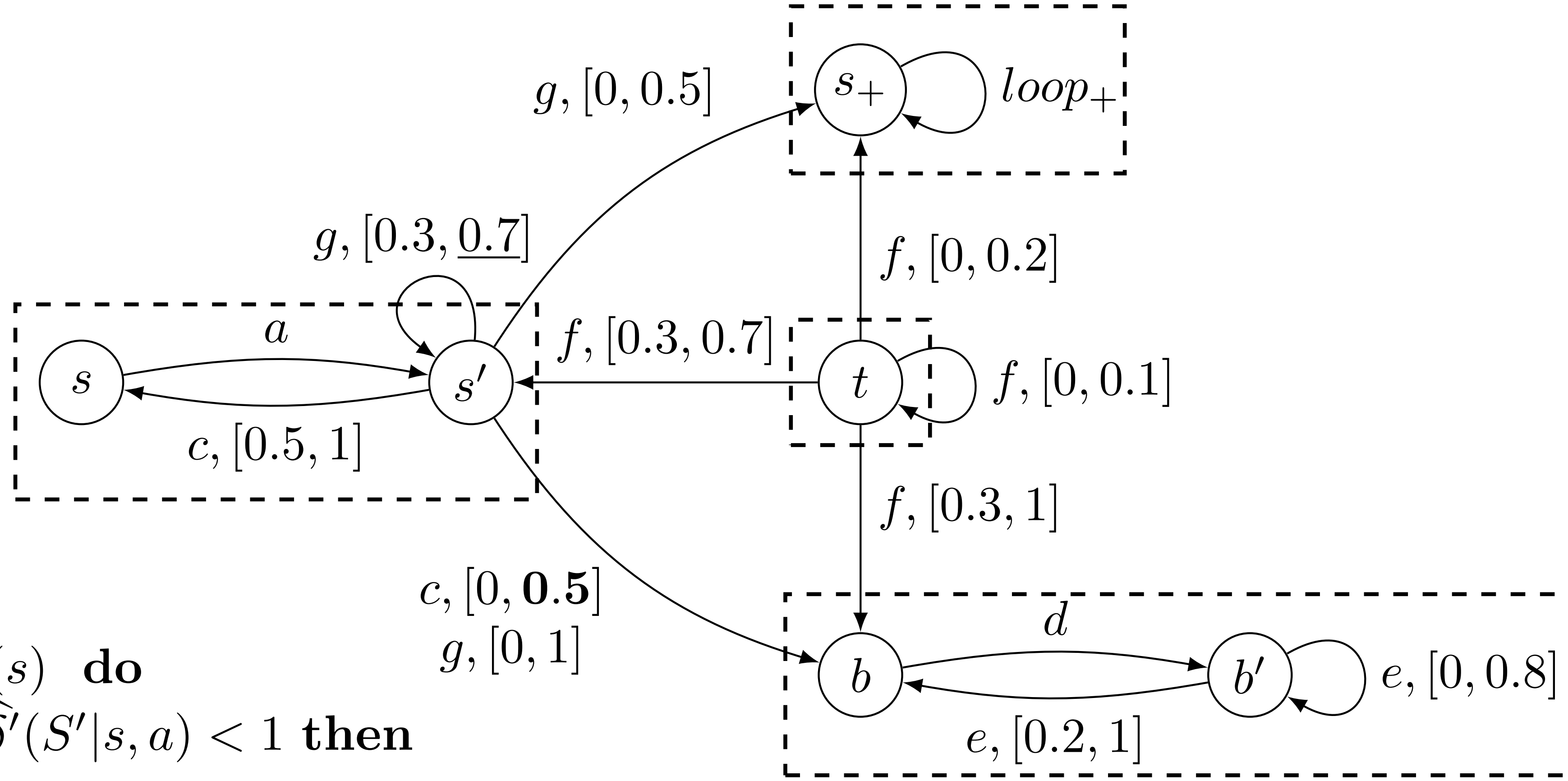
- Simulate on the IMDP the value iteration on its MDP…
- One step is the application of

$$f_{\max}(x)_s = \max_{a \in A(s)} \max_{p \in \mathrm{BFS}(a)} \sum_{s' \in S} p(s') \times x_{s'}$$

- Achievable in polynomial time by sorting x…

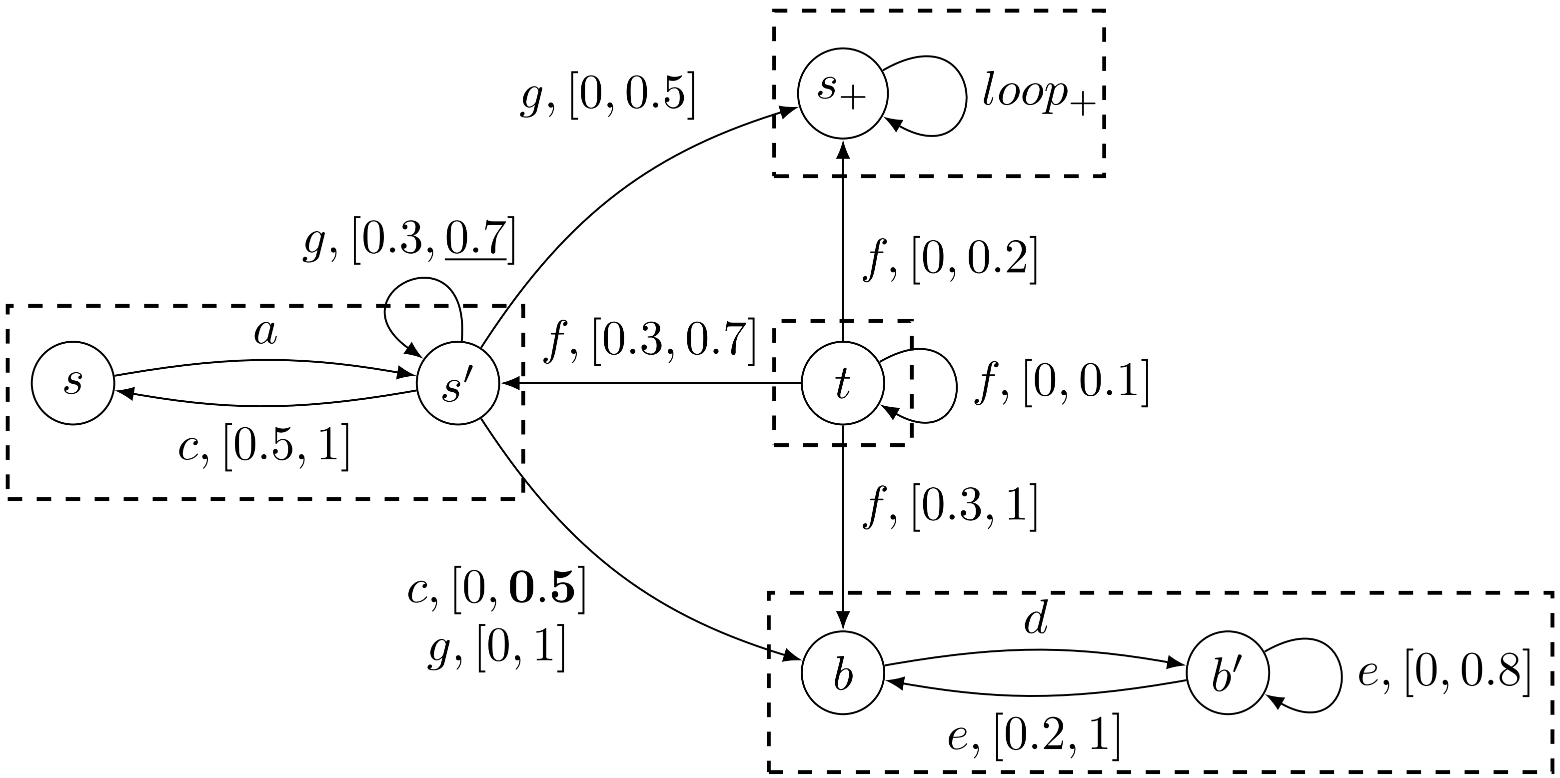
[Sen, Viswanathan, Agha, 2006]

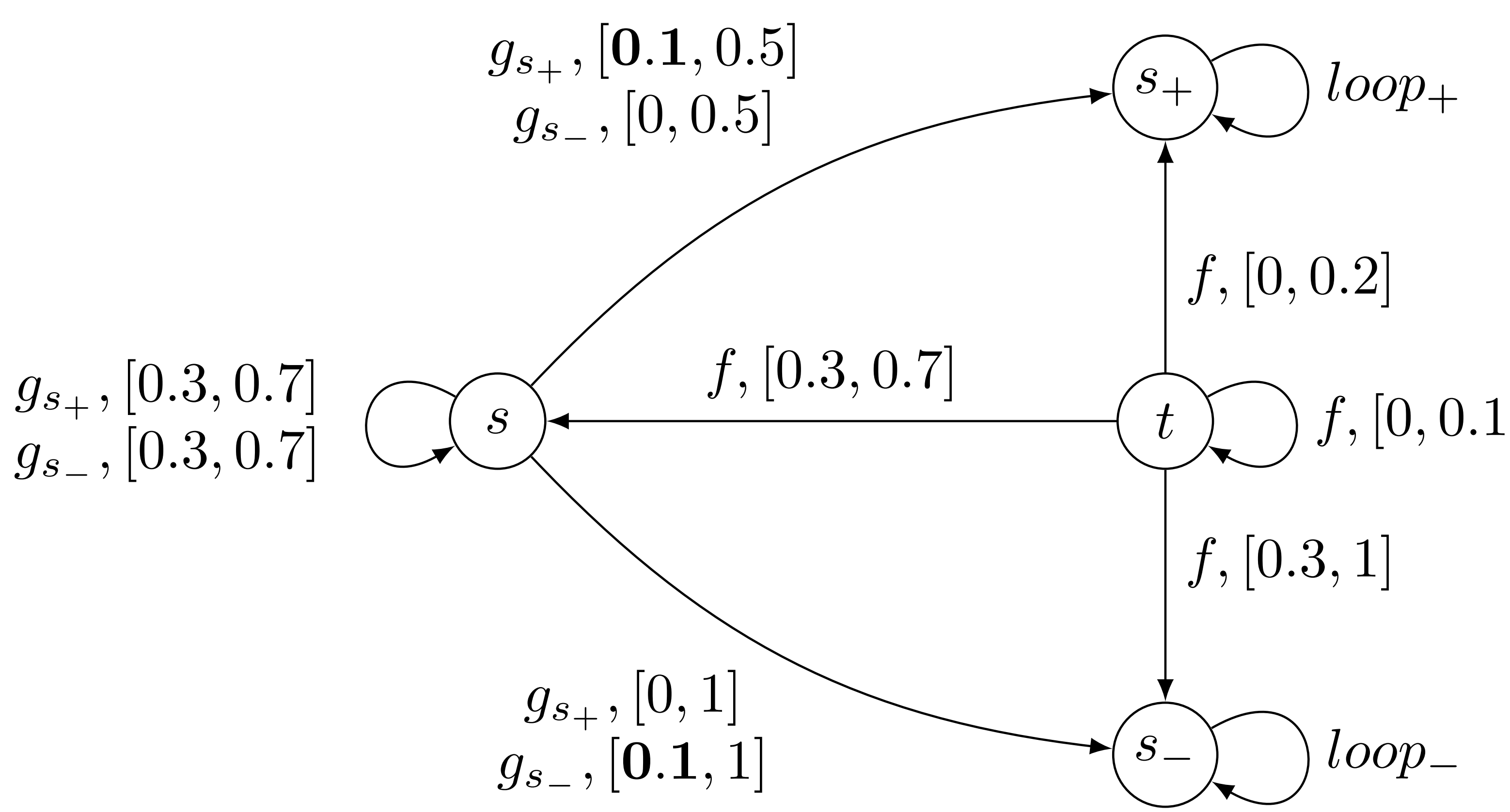
MEC decomposition



Push$(stack, \mathcal{M})$; $\mathcal{SM} \leftarrow \emptyset$
while not Empty$(stack)$ **do**
  $(S', \alpha', \check{\delta}', \widehat{\delta}') \leftarrow$ Pop$(stack)$
  for $s \in S'$ **and** $a \in \alpha' \cap A(s)$ **do**
    if $\check{\delta}'(S \setminus S'|s,a) > 0 \vee \widehat{\delta}'(S'|s,a) < 1$ **then**
      $\alpha' \leftarrow \alpha' \setminus \{a\}$
    else
      for $s' \notin S'$ **do** $\widehat{\delta}'(s'|s,a) \leftarrow 0$
  $E \leftarrow \emptyset$
  for $s, s' \in S'$ **and** $a \in \alpha' \cap A(s)$ **do**
    if $\widehat{\delta}'(s'|s,a) > 0 \wedge \check{\delta}'(S \setminus \{s'\}|s,a) < 1$ **then** $E \leftarrow E \cup \{(s,s')\}$
  compute the strongly connected components of (S', E): $S_1, \ldots, S_K$
  if $K > 1$ **then**
    for $i = 1$ **to** K **do** Push$(stack, (S_i, \alpha' \cap \bigcup_{s \in S_i} A(s), \check{\delta}'|_{S_i}, \widehat{\delta}'|_{S_i}))$
  else $\mathcal{SM} \leftarrow \mathcal{SM} \cup \{(S', \alpha', \check{\delta}', \widehat{\delta}')\}$
return $\mathcal{SM}$

Max-reduction





Conclusion and related work

- Framework allowing **guarantees** for **value iteration algorithm**
- General results on **convergence rate**
- Criterion for computation of **exact value**
- Future work: test of our preliminary implementation over real instances

Conclusion and related work

- Framework allowing **guarantees** for **value iteration algorithm**
- General results on **convergence rate**
- Criterion for computation of **exact value**
- Future work: test of our preliminary implementation over real instances

$\sim$

- **[Katoen, Zapreev, 2006]** On-the-fly detection of steady-state in the transient analysis of continuous-time Markov chains

Conclusion and related work

- Framework allowing **guarantees** for **value iteration algorithm**
- General results on **convergence rate**
- Criterion for computation of **exact value**
- Future work: test of our preliminary implementation over real instances

$\sim$

- **[Katoen, Zapreev, 2006]** On-the-fly detection of steady-state in the transient analysis of continuous-time Markov chains
- **[Kattenbelt, Kwiatkowska, Norman, Parker, 2010]** CEGAR-based approach for stochastic games

Conclusion and related work

- Framework allowing **guarantees** for **value iteration algorithm**
- General results on **convergence rate**
- Criterion for computation of **exact value**
- Future work: test of our preliminary implementation over real instances

$\sim$

- **[Katoen, Zapreev, 2006]** On-the-fly detection of steady-state in the transient analysis of continuous-time Markov chains
- **[Kattenbelt, Kwiatkowska, Norman, Parker, 2010]** CEGAR-based approach for stochastic games
- **[Brázdil, Chatterjee, Chmelík, Forejt, Křetínský, Kwiatkowska, Parker, Ujma, ATVA 2014]** same techniques in a machine learning framework with almost sure convergence and computation of non-trivial end components on-the-fly