Reachability in MDPs: Refining Convergence of Value Iteration

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

Fourth Cassting Meeting
October 2014, Aachen

UNIVERSITÉ LIBRE DE BRUXELLES

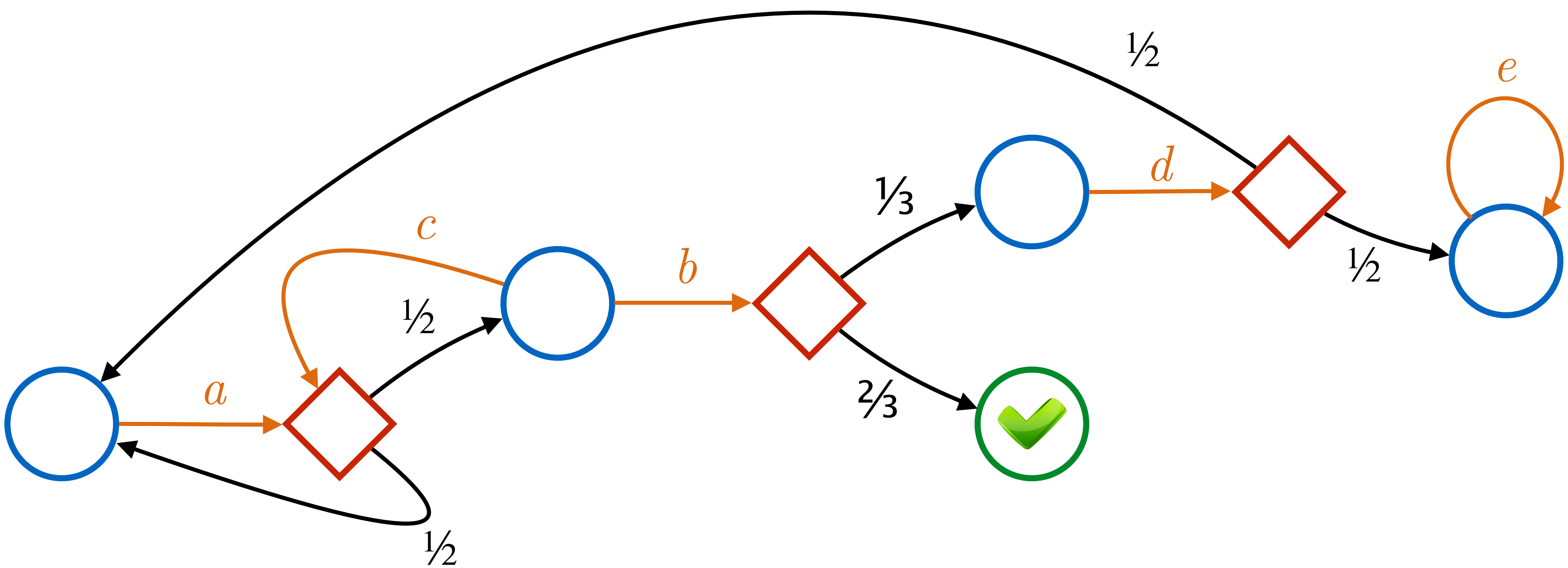
Markov Decision Processes

- What?
  - *Stochastic* process with *non-deterministic* choices
  - Non-determinism solved by *policies*/strategies

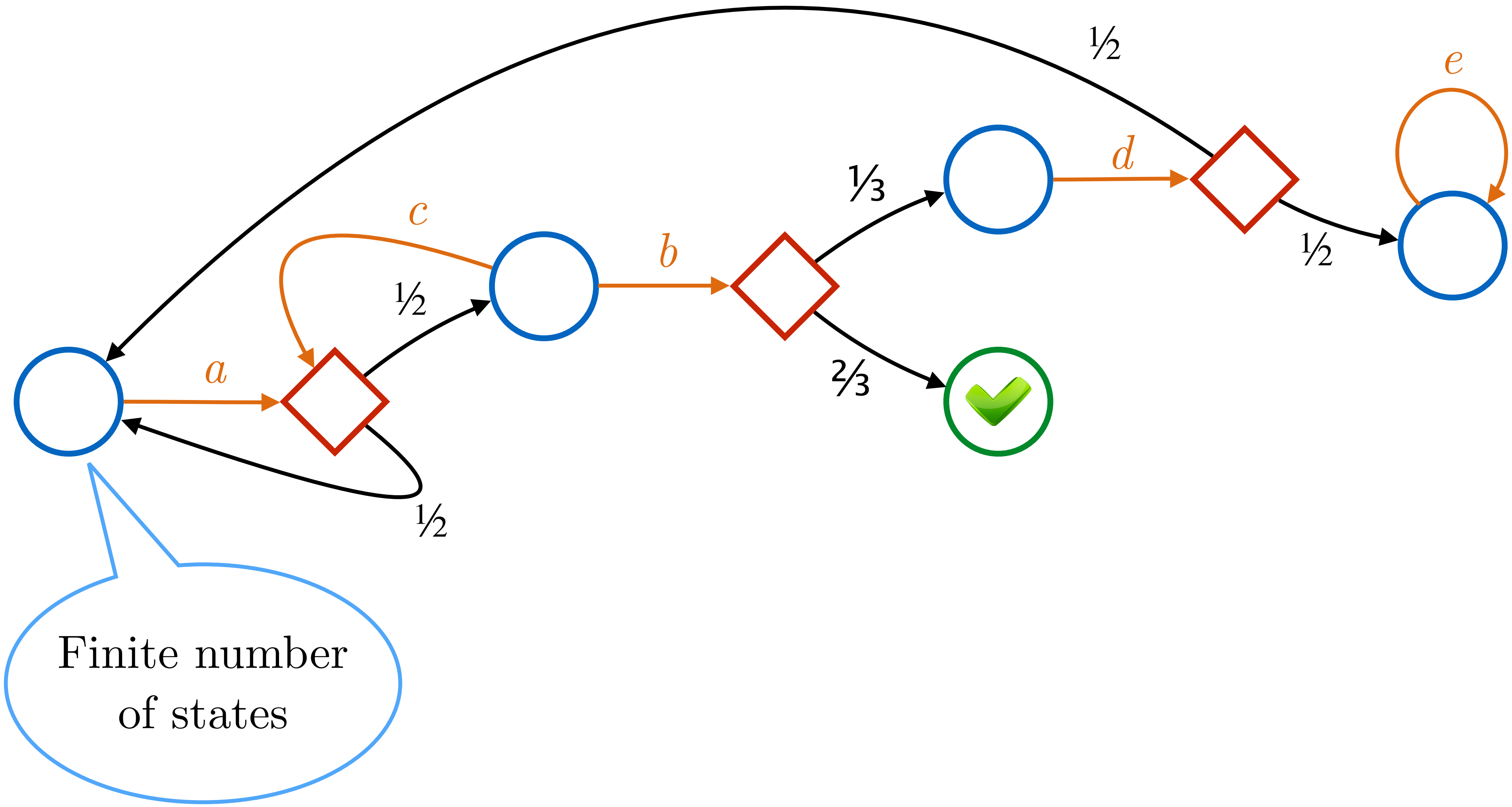
Markov Decision Processes

- What?
  - *Stochastic* process with *non-deterministic* choices
  - Non-determinism solved by *policies*/strategies
- Where?
  - *Optimization*
  - *Program verification*: reachability as the basis of PCTL model-checking
  - *Game theory*: 1+½ players

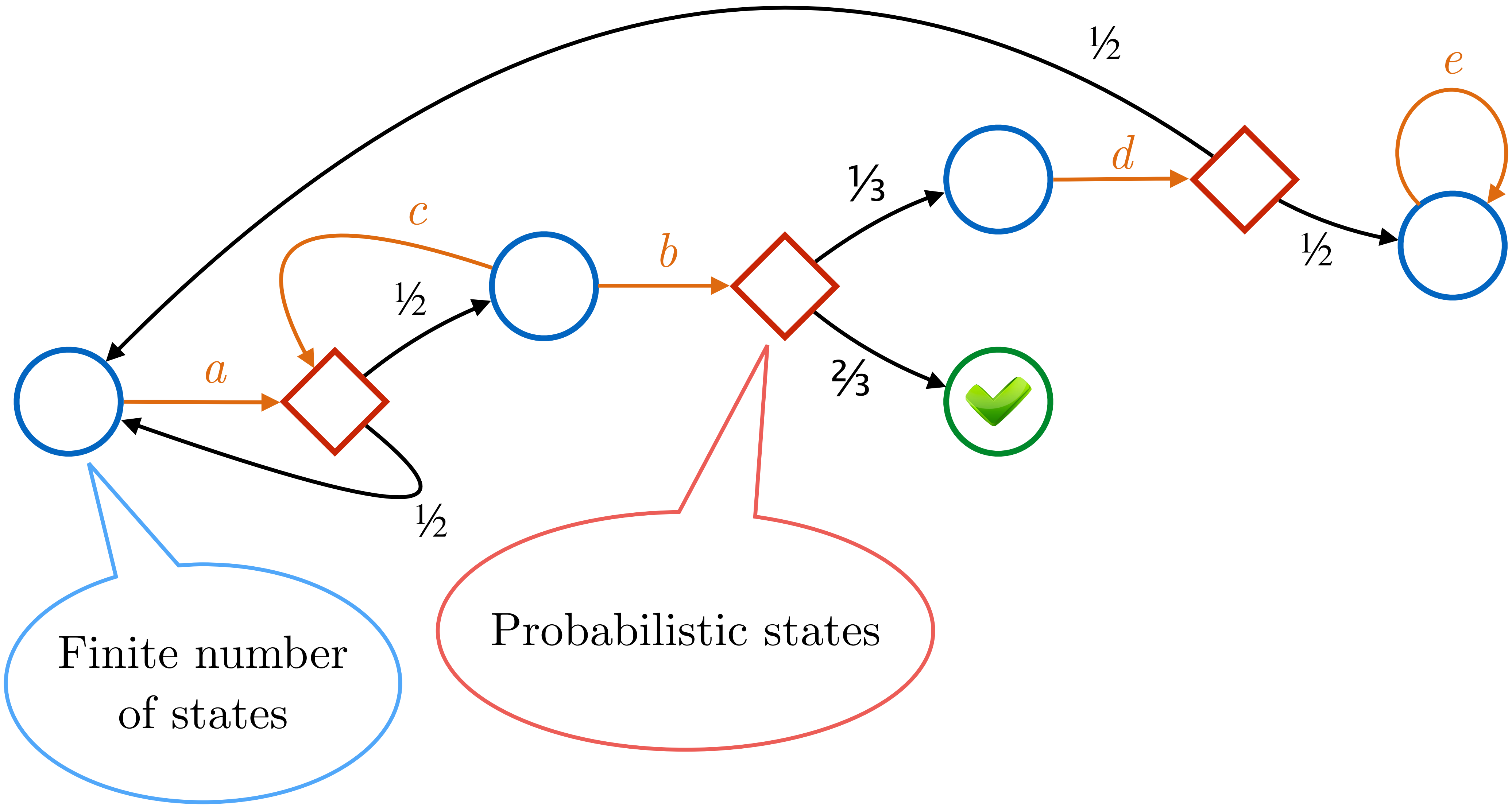
MDPs: definition and objective



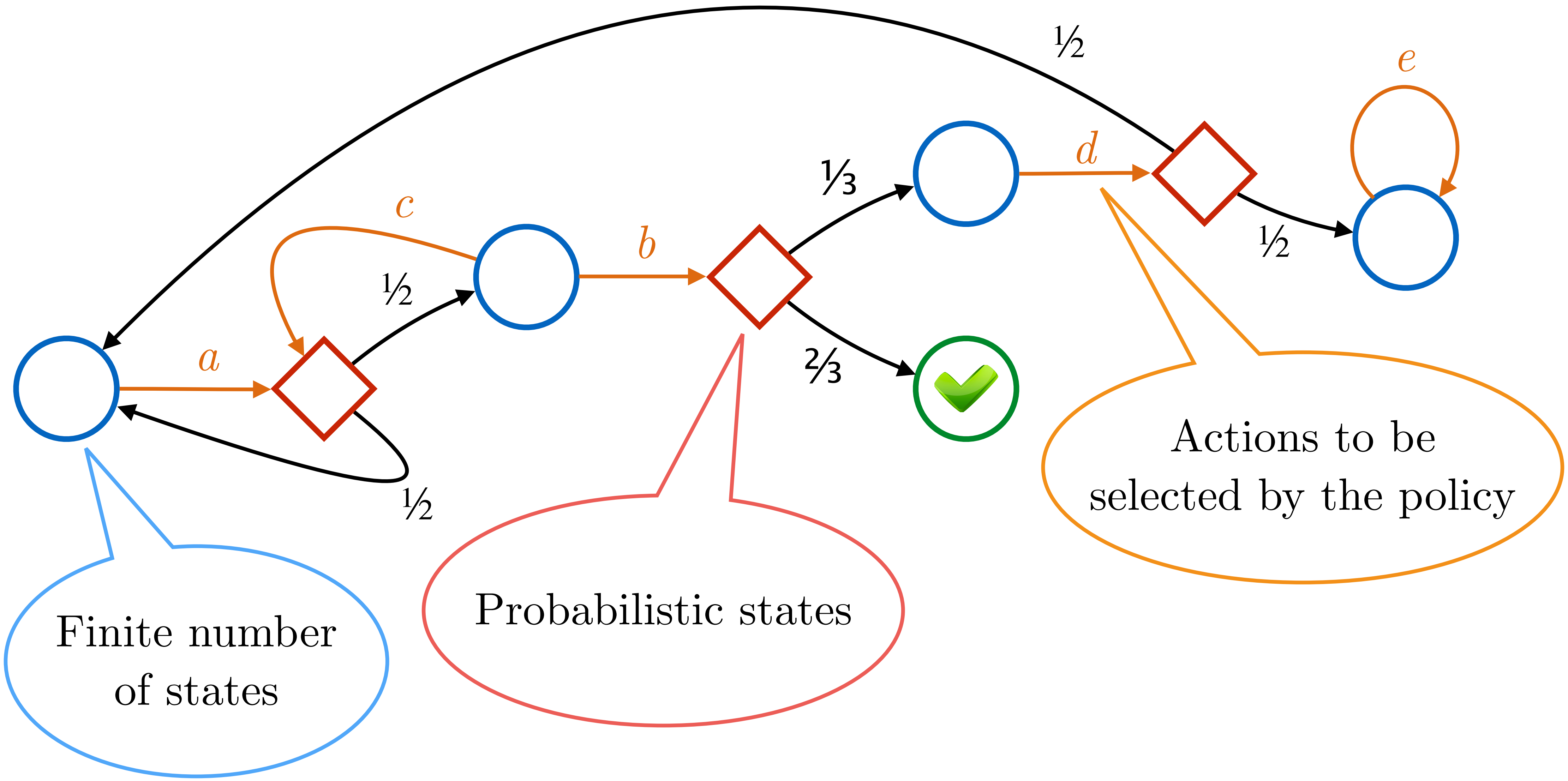
MDPs: definition and objective



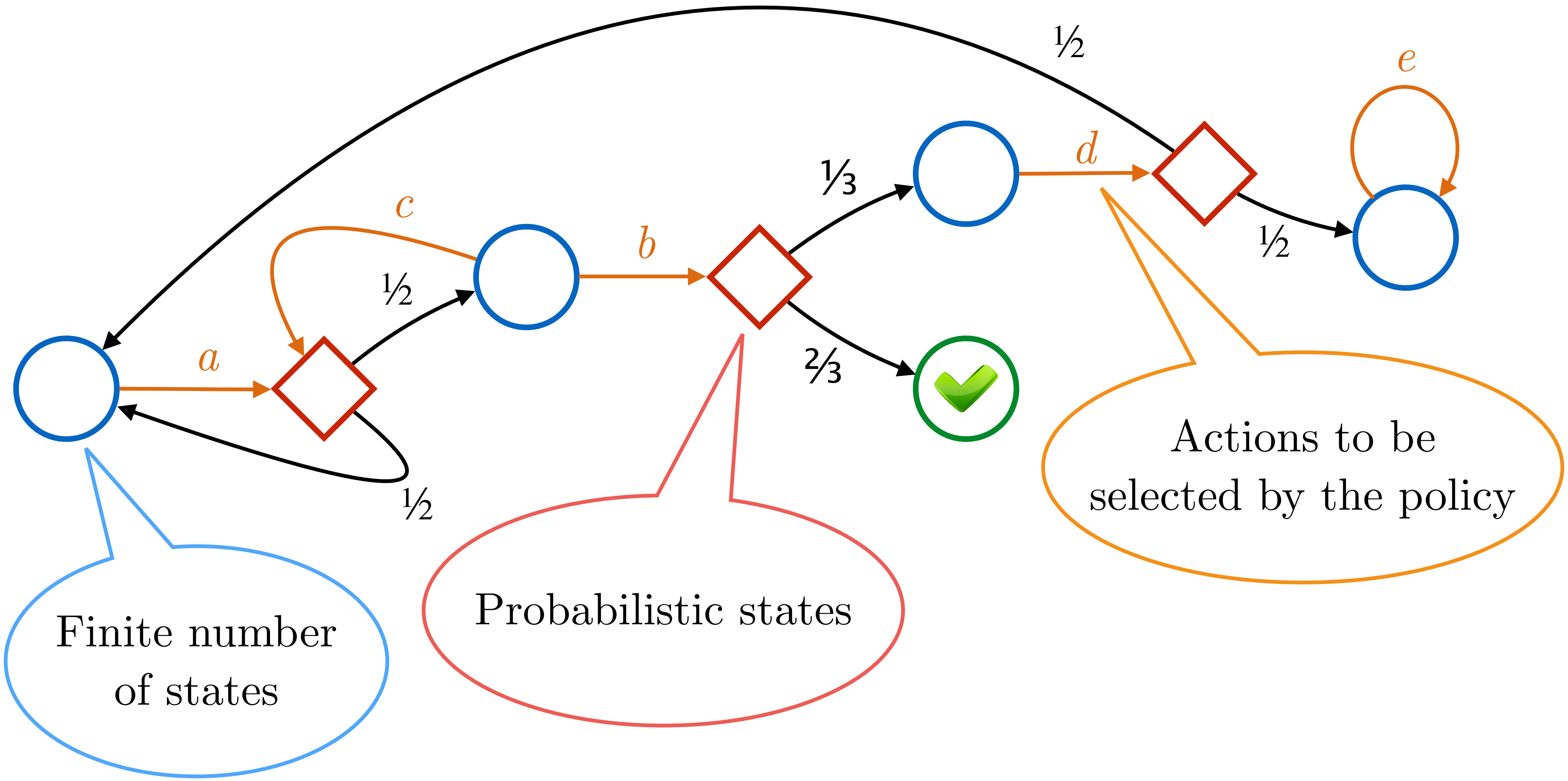
MDPs: definition and objective



MDPs: definition and objective



MDPs: definition and objective

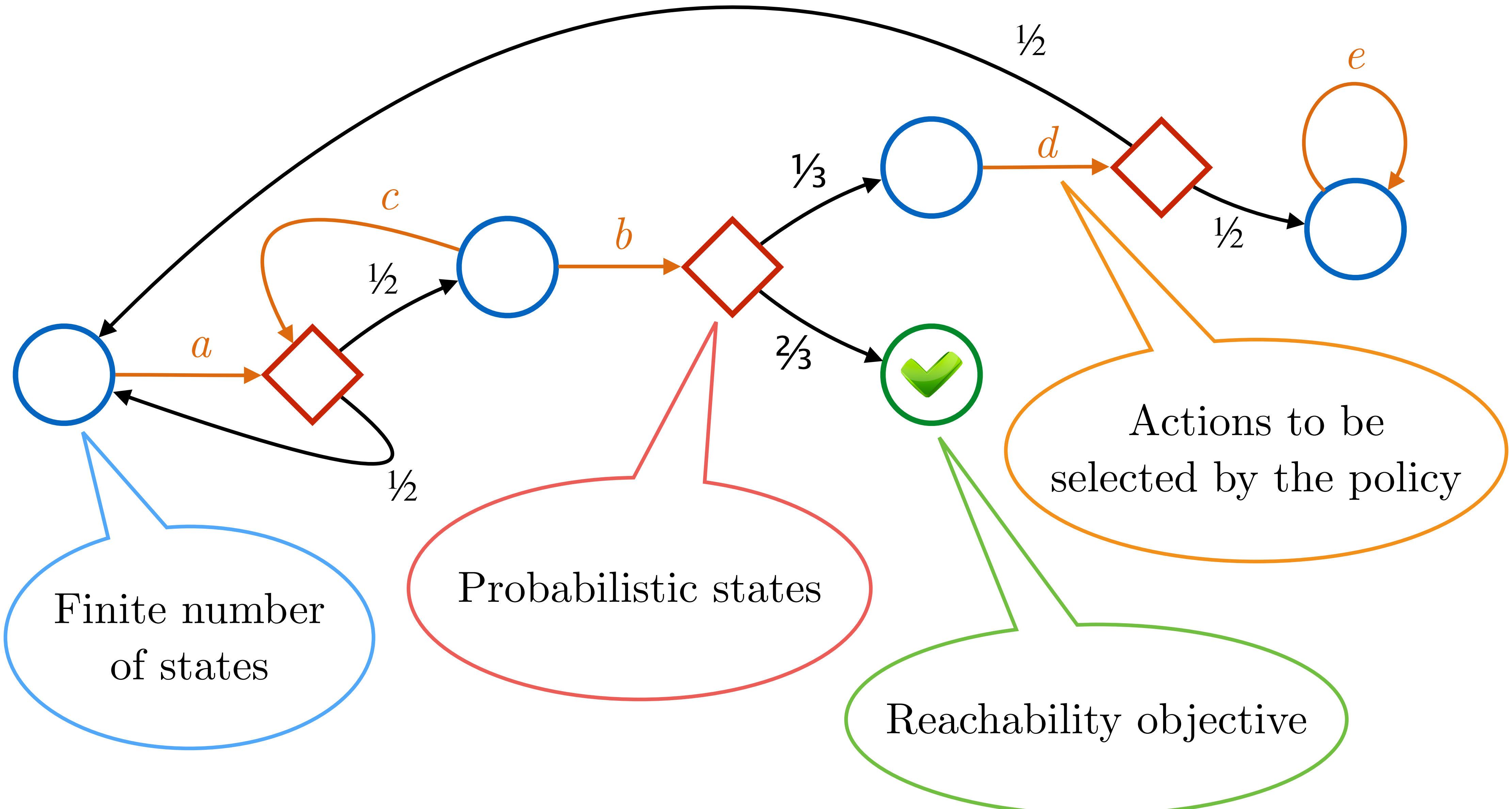


$$\mathcal{M} = (S, \alpha, \delta)$$

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

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

MDPs: definition and objective

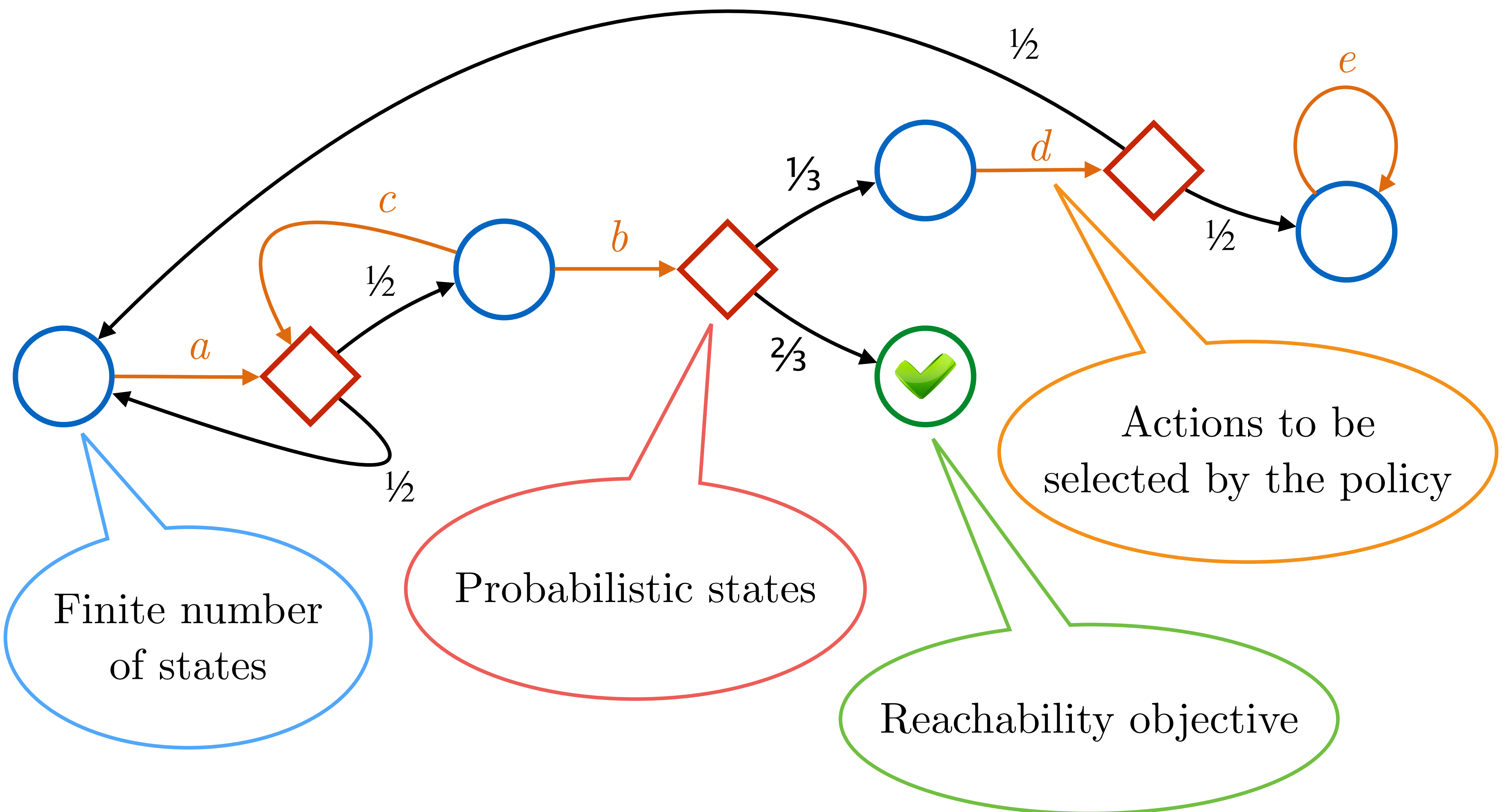


$$\mathcal{M} = (S, \alpha, \delta)$$

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

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

MDPs: definition and objective



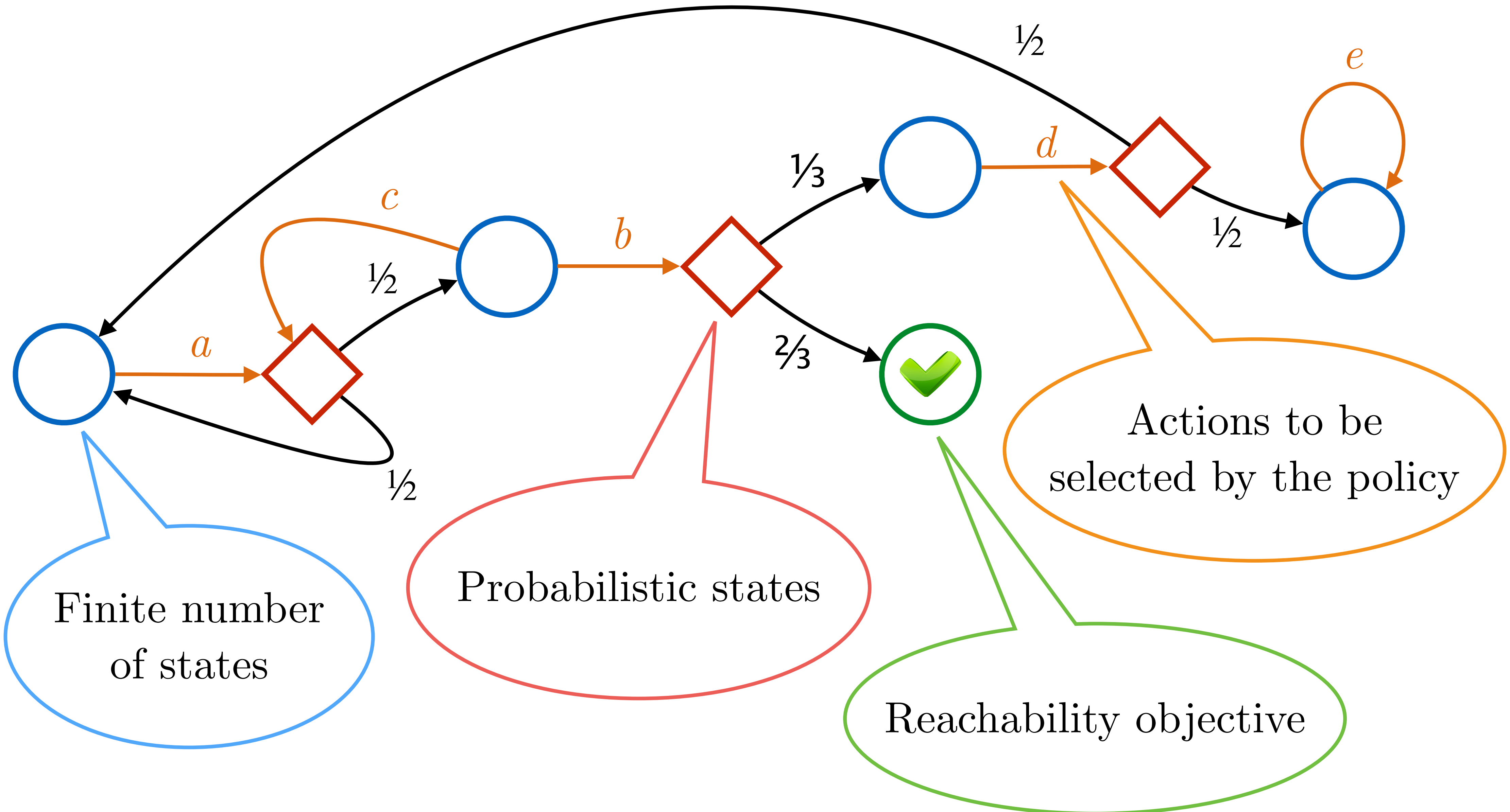
$$\mathcal{M} = (S, \alpha, \delta)$$

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

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

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

MDPs: definition and objective



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

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

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

Probability to reach: $\mathrm{Pr}_s^{\sigma}(\mathsf{F}\ ✔)$

Maximal probability to reach: $\mathrm{Pr}_s^{\max}(\mathsf{F}\ ✔) = \sup_{\sigma} \mathrm{Pr}_s^{\sigma}(\mathsf{F}\ ✔)$

Optimal reachability probabilities of MDPs

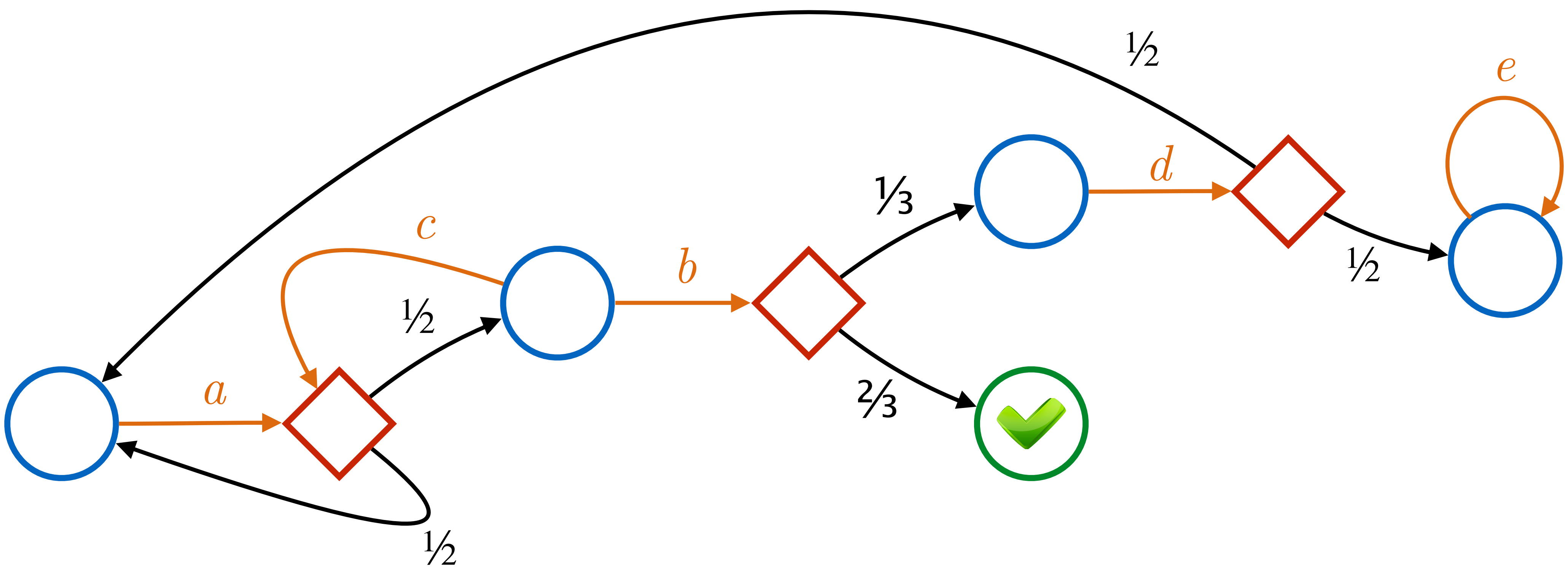
- How?
  - *Linear programming*
  - *Policy iteration*
  - *Value iteration*: numerical scheme that scales well and works in practice

Optimal reachability probabilities of MDPs

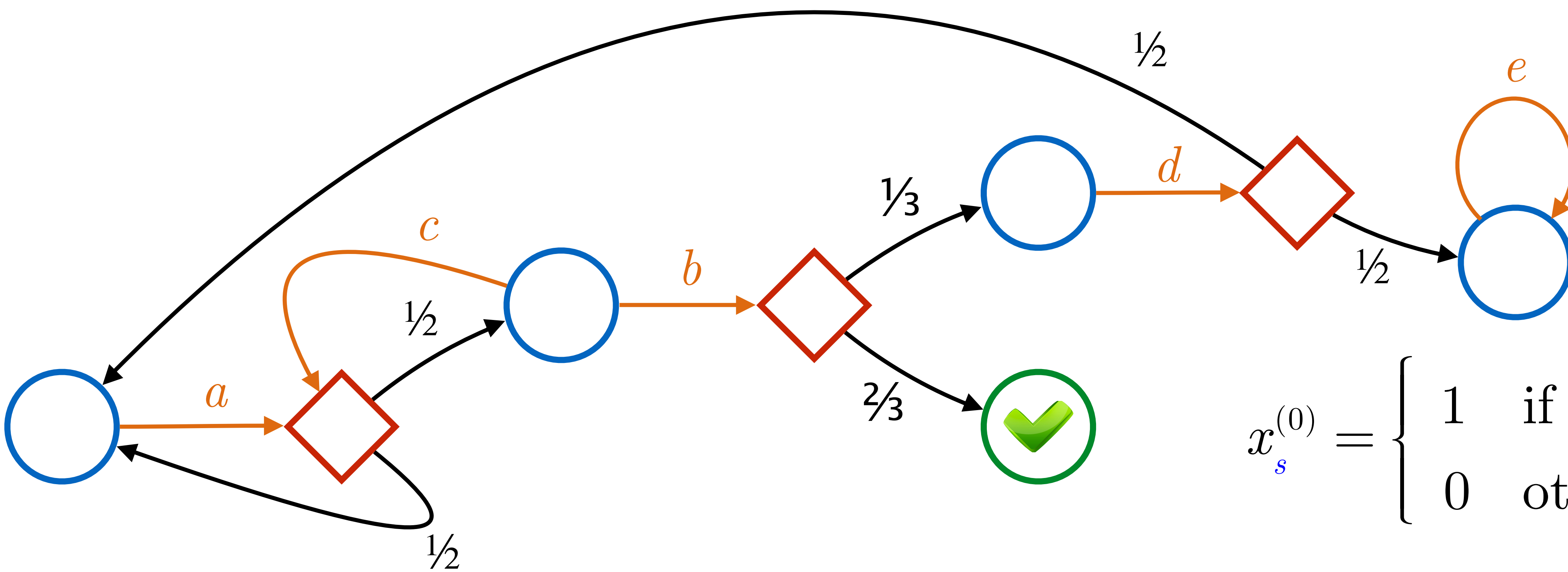
- How?
  - ✦ *Linear programming*
  - ✦ *Policy iteration*
  - ✦ *Value iteration*: numerical scheme that scales well and works in practice

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

Value iteration



Value iteration

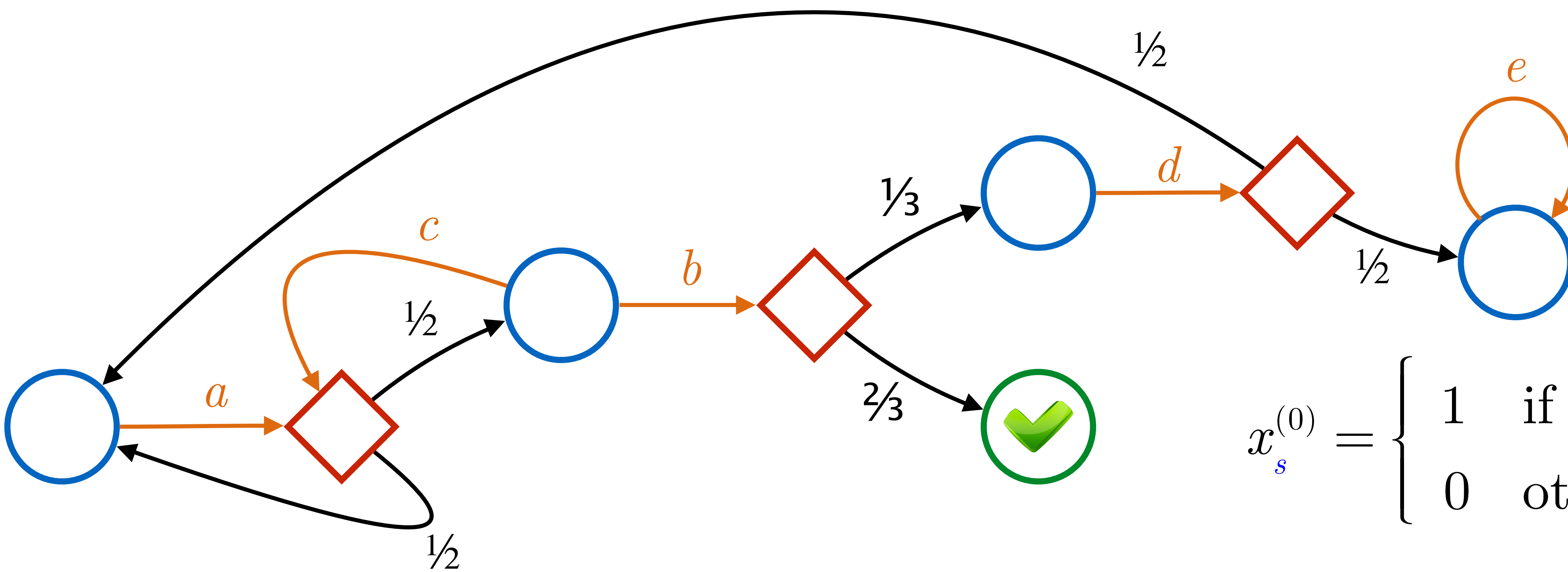


$$x_s^{(0)} = \begin{cases} 1 & \text{if } s = \checkmark \\ 0 & \text{otherwise} \end{cases}$$

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

Value iteration

0	0	0	0

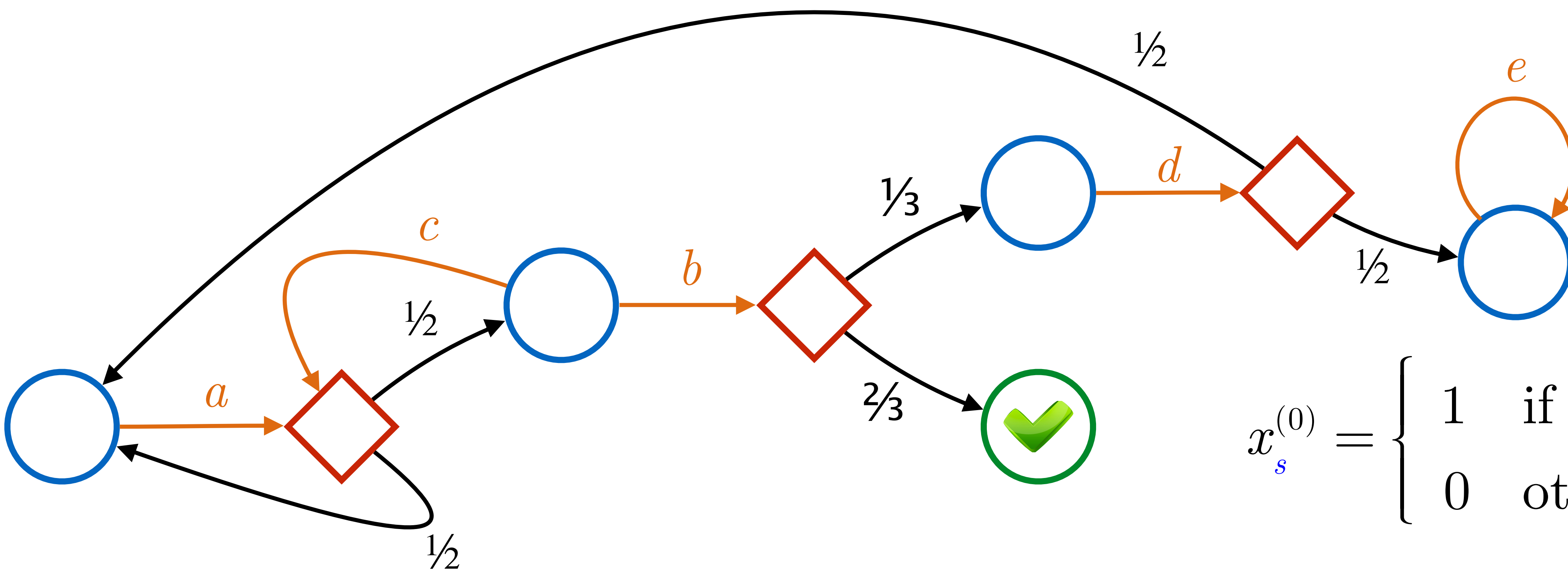


$$x_s^{(0)} = \begin{cases} 1 & \text{if } s = \checkmark \\ 0 & \text{otherwise} \end{cases}$$

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

Value iteration

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

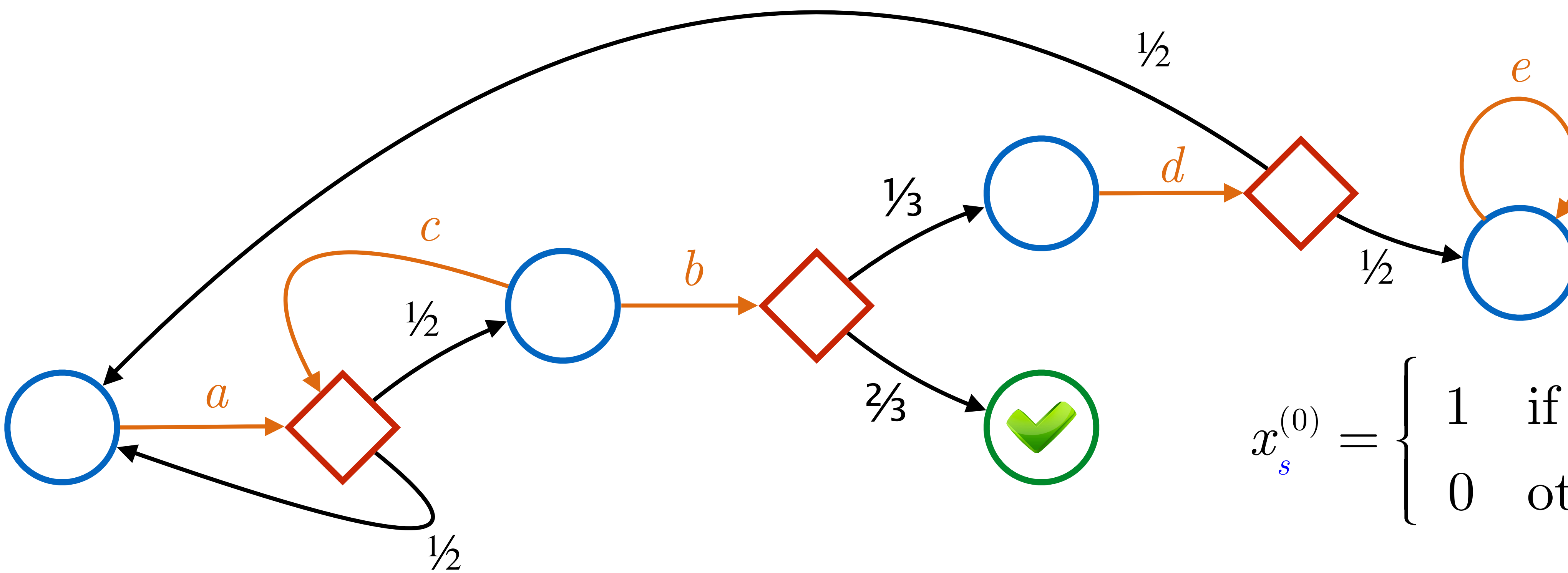


$$x_s^{(0)} = \begin{cases} 1 & \text{if } s = \checkmark \\ 0 & \text{otherwise} \end{cases}$$

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

Value iteration

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

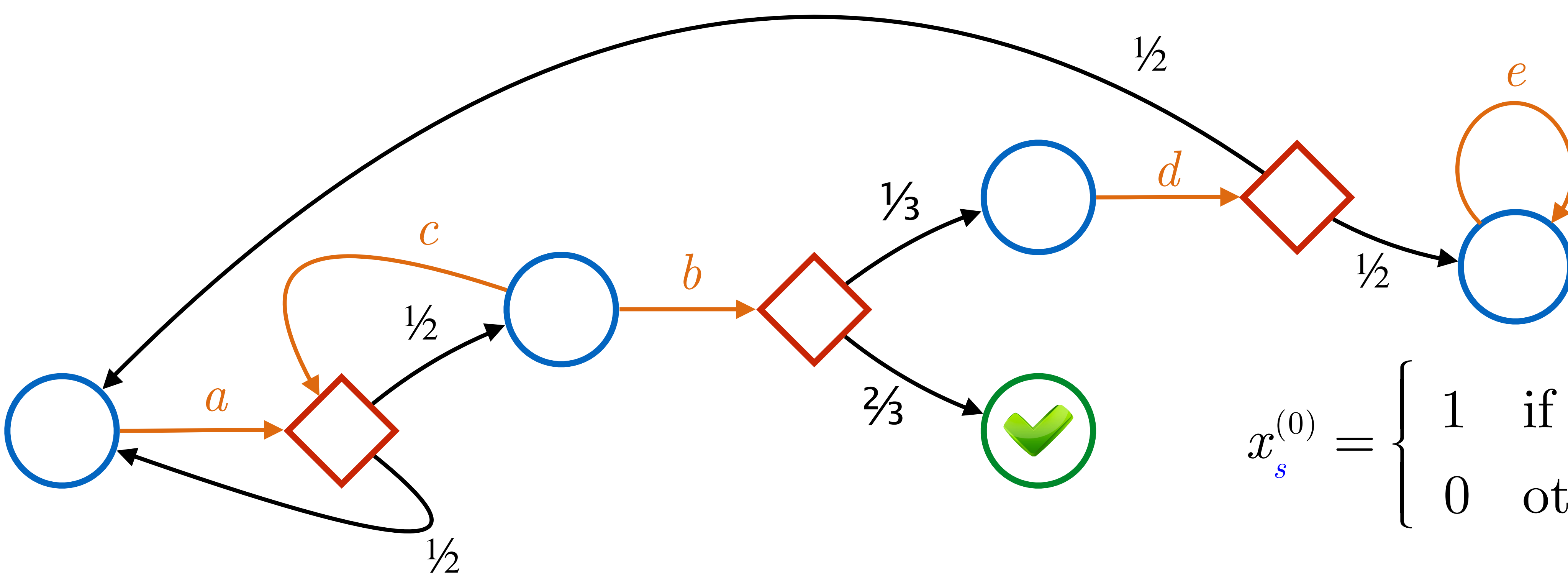


$$x_s^{(0)} = \begin{cases} 1 & \text{if } s = \checkmark \\ 0 & \text{otherwise} \end{cases}$$

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

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

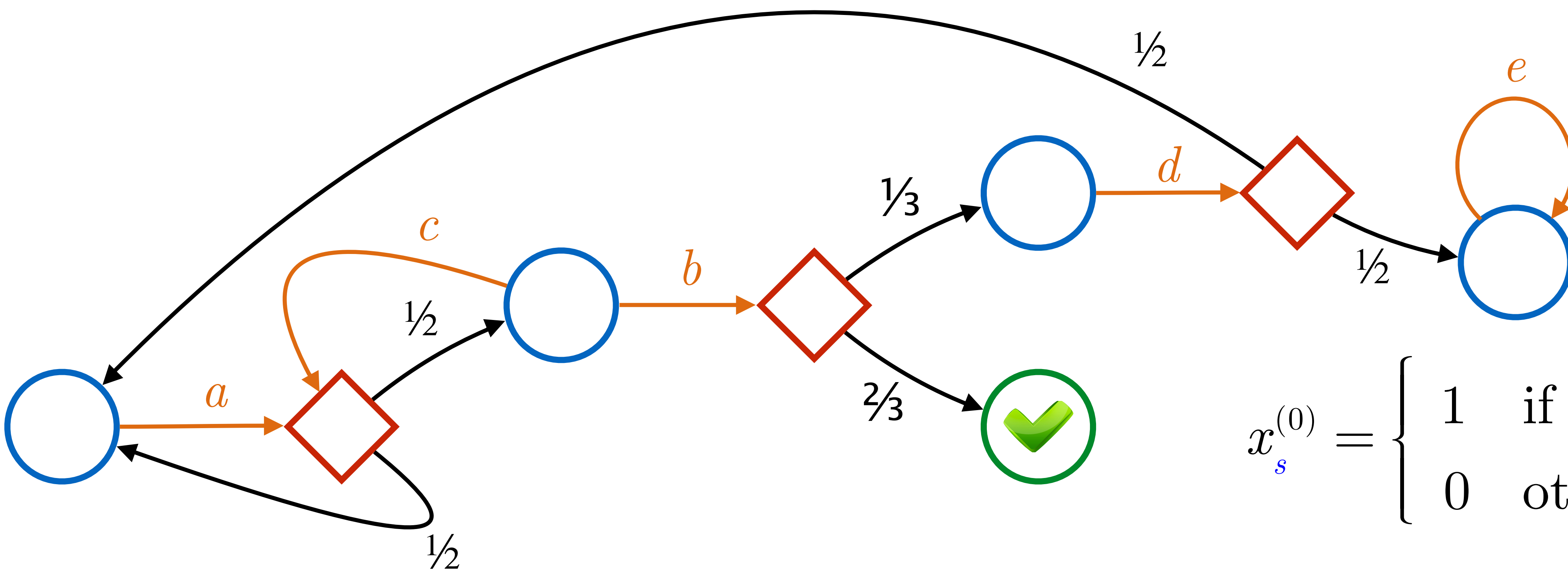


$$x_s^{(0)} = \begin{cases} 1 & \text{if } s = ✔ \\ 0 & \text{otherwise} \end{cases}$$

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

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

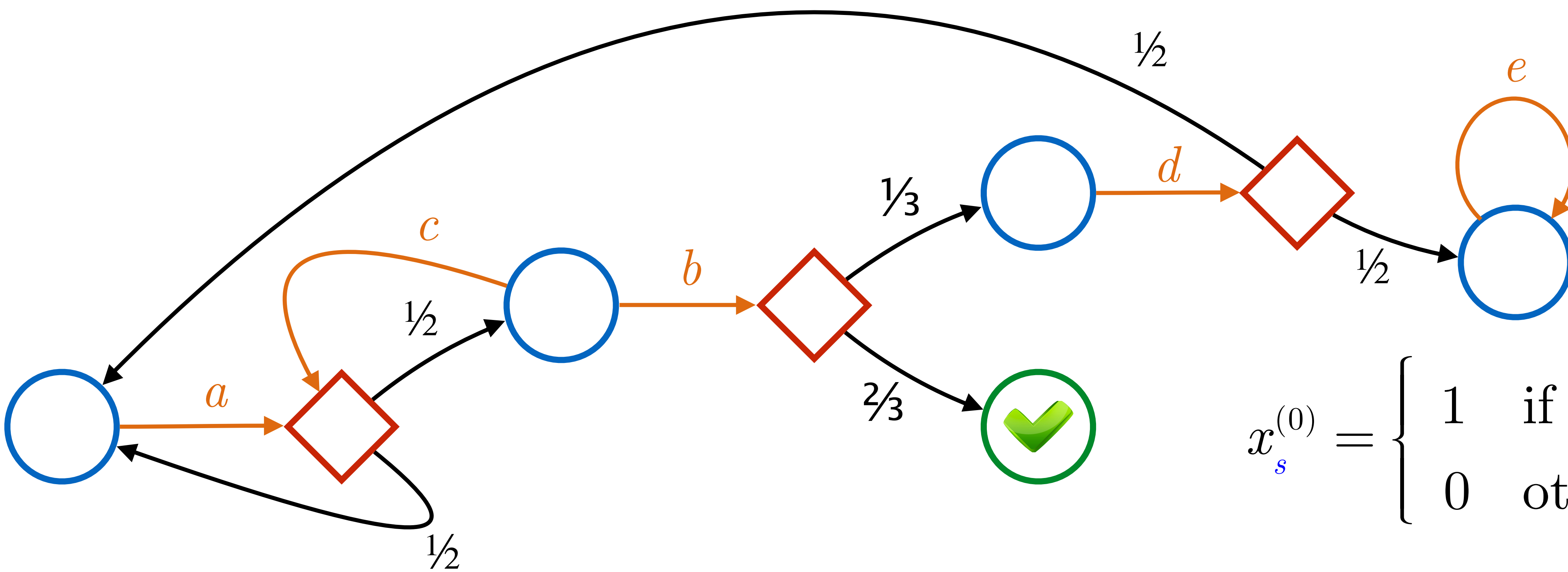


$$x_s^{(0)} = \begin{cases} 1 & \text{if } s = \checkmark \\ 0 & \text{otherwise} \end{cases}$$

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

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
$\dots$	$\dots$	$\dots$	$\dots$

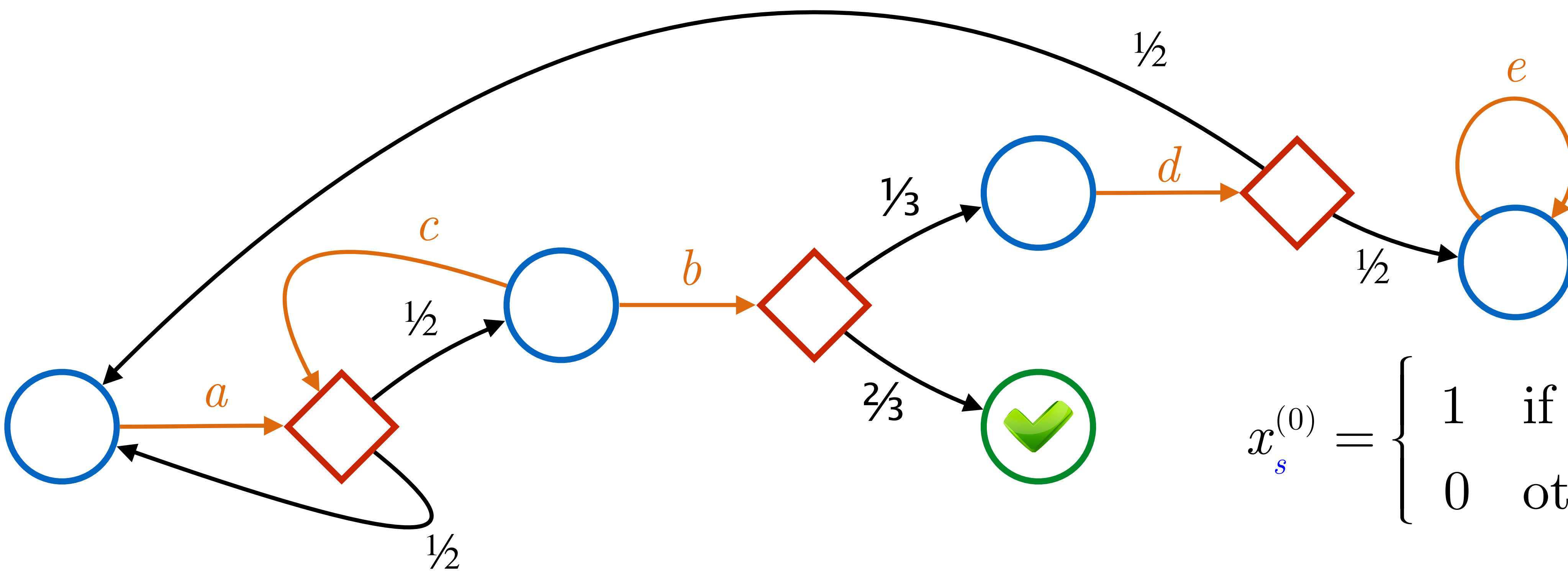


$$x_s^{(0)} = \begin{cases} 1 & \text{if } s = \checkmark \\ 0 & \text{otherwise} \end{cases}$$

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

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

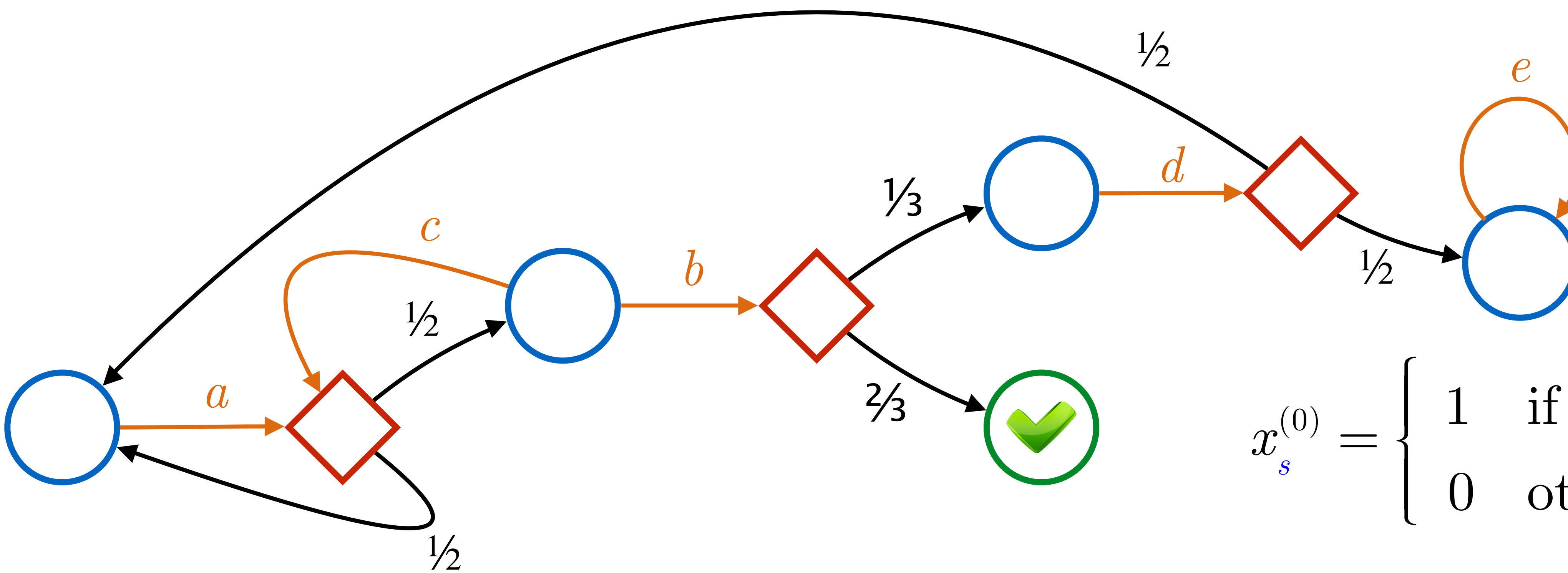


$$x_s^{(0)} = \begin{cases} 1 & \text{if } s = \checkmark \\ 0 & \text{otherwise} \end{cases}$$

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

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



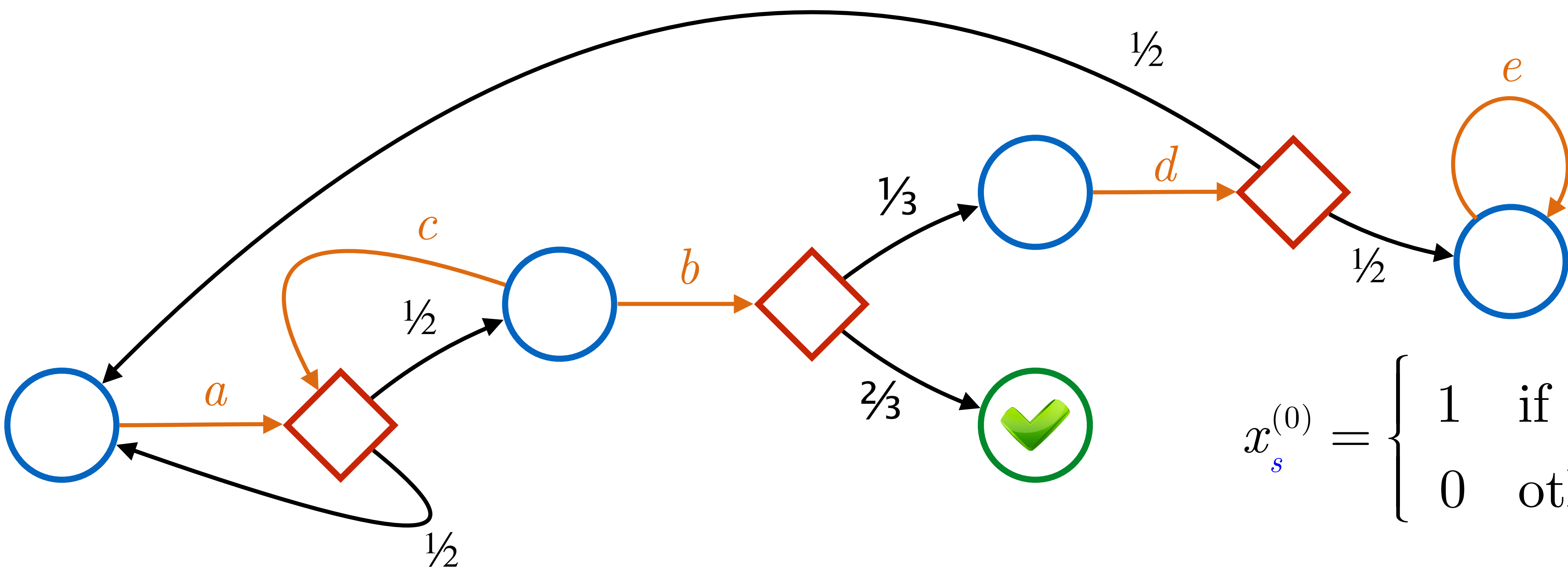
$$x_s^{(0)} = \begin{cases} 1 & \text{if } s = ✓ \\ 0 & \text{otherwise} \end{cases}$$

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

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

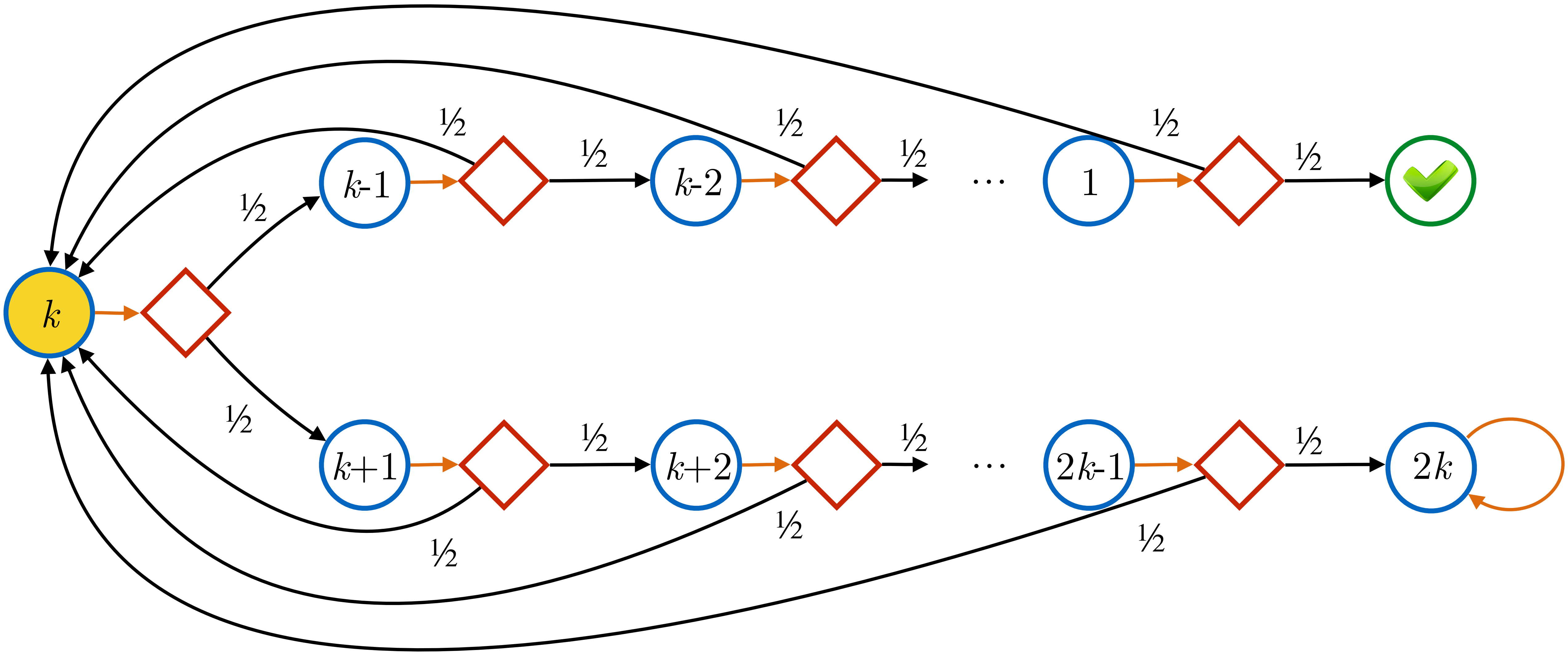
≤ 0.001



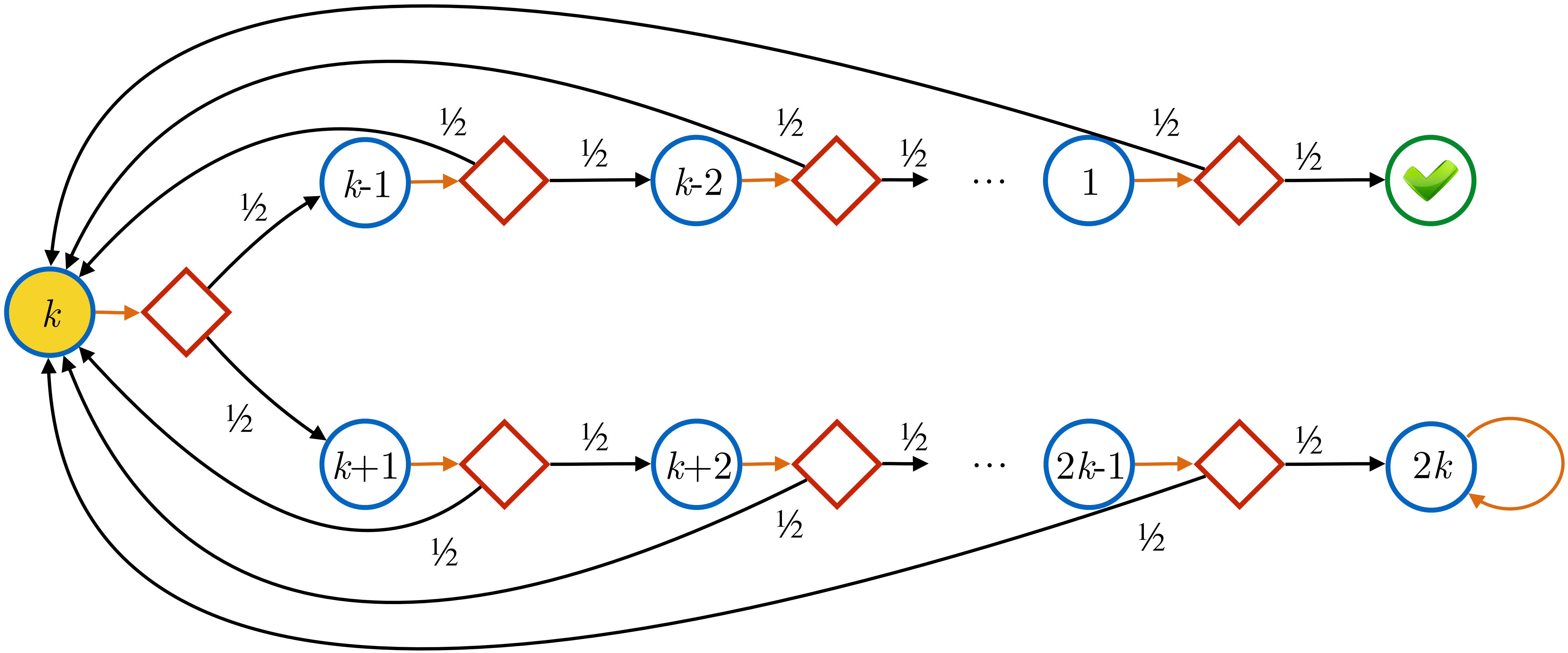
$$x_s^{(0)} = \begin{cases} 1 & \text{if } s = \checkmark \\ 0 & \text{otherwise} \end{cases}$$

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

Value iteration: which guarantees?

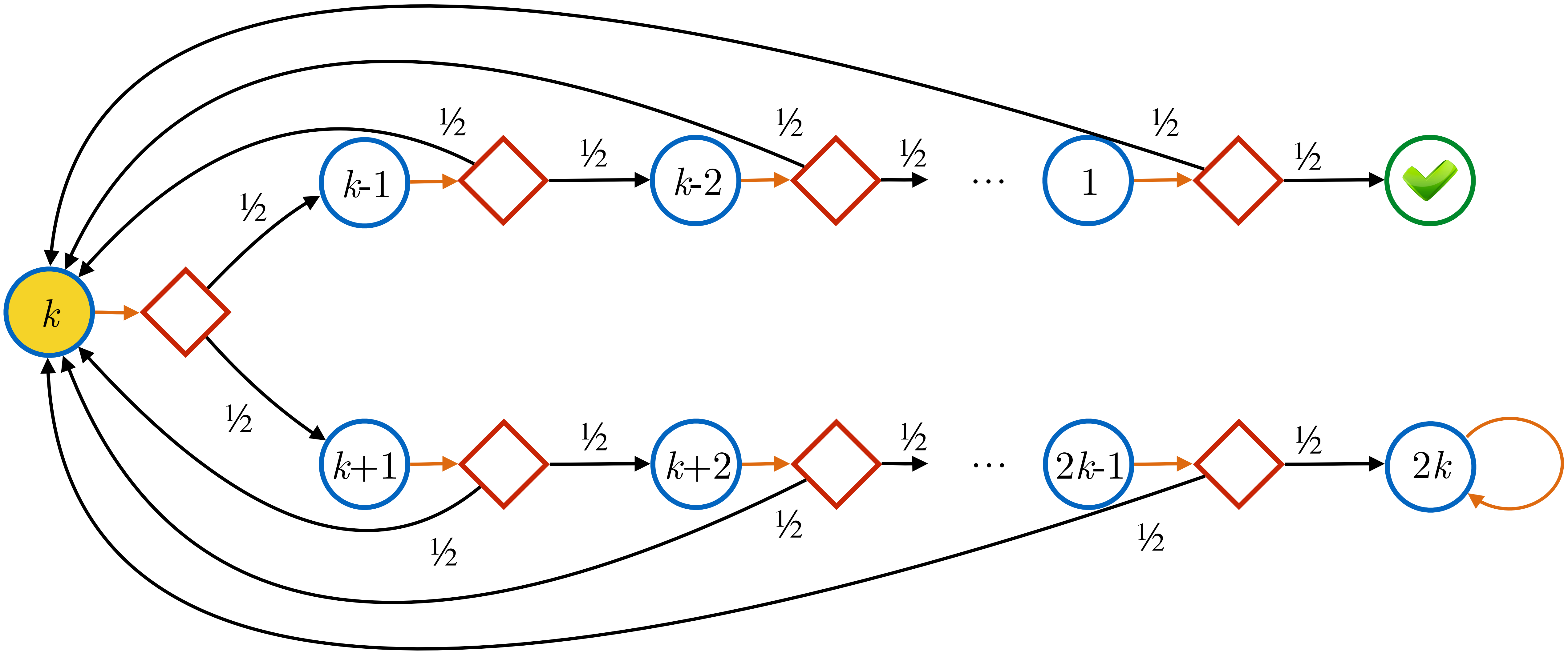


Value iteration: which guarantees?



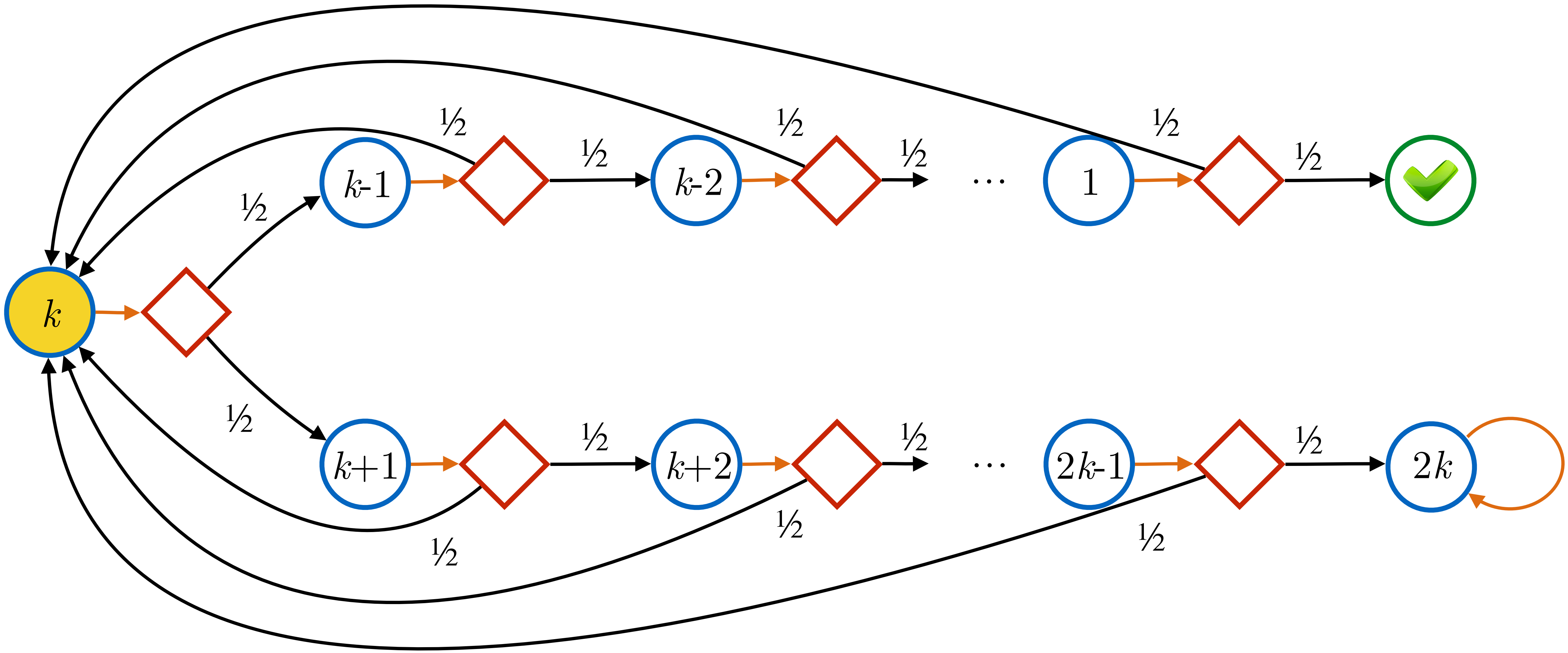
State	0	1	2	3	…	k-1	k	k+1	…	$2k$

Value iteration: which guarantees?



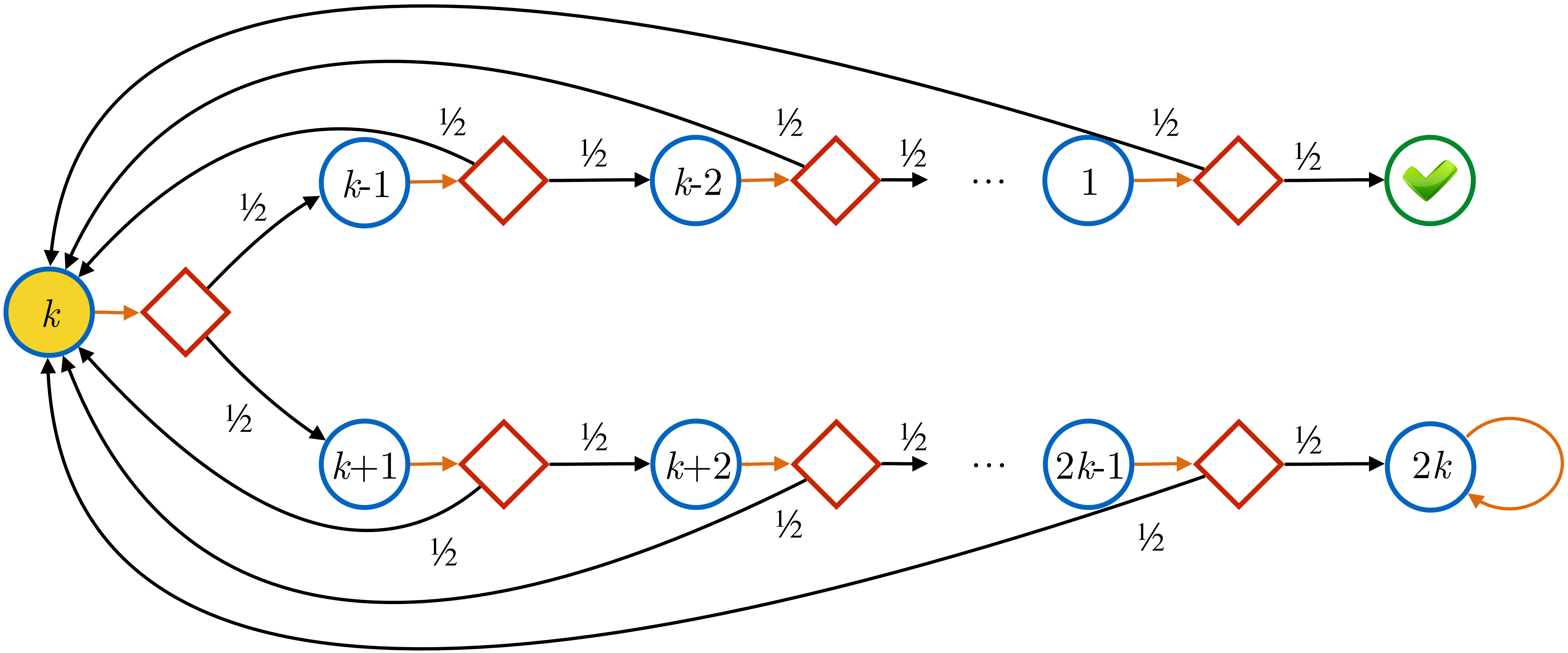
State	0	1	2	3	…	k-1	k	k+1	…	2k
Step 1	1	0	0	0	…	0	0	0	…	0

Value iteration: which guarantees?



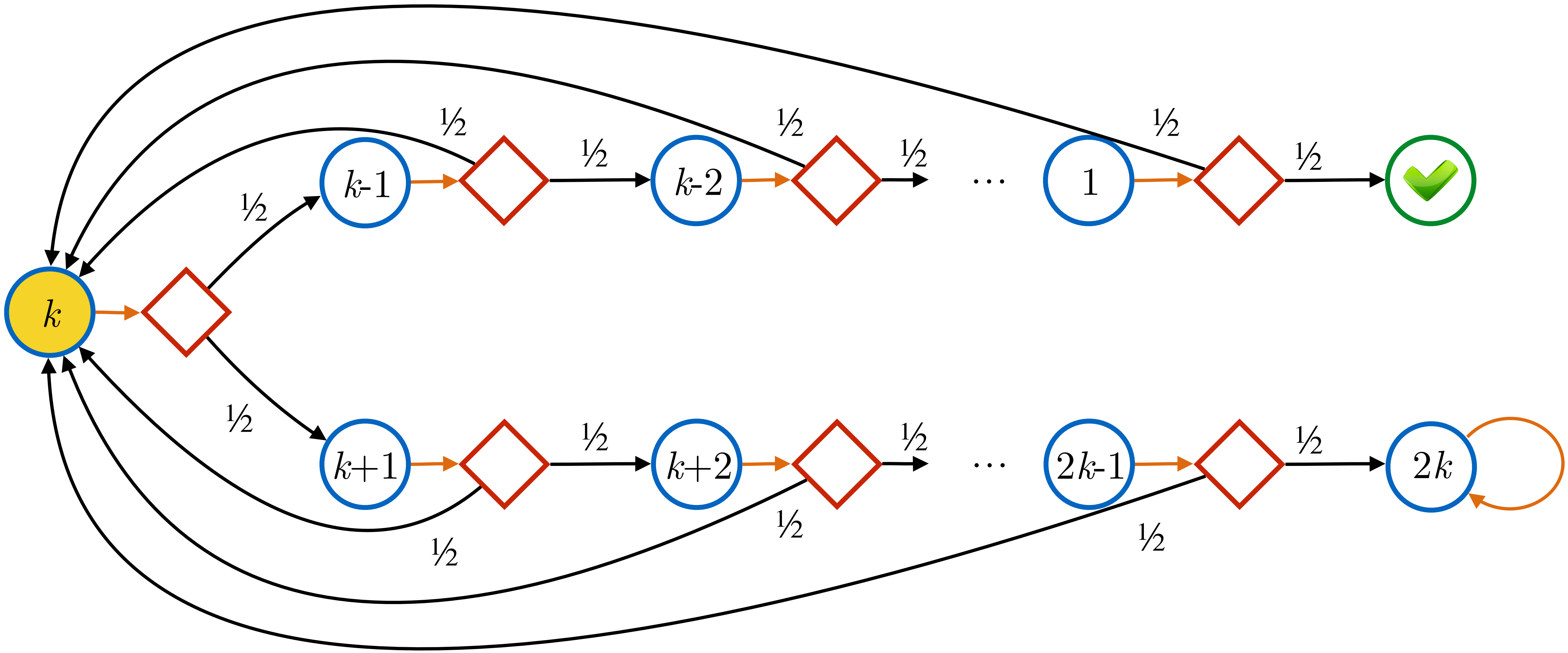
State	0	1	2	3	…	k-1	k	k+1	…	2k
Step 1	1	0	0	0	…	0	0	0	…	0
Step 2	1	1/2	0	0	…	0	0	0	…	0

Value iteration: which guarantees?



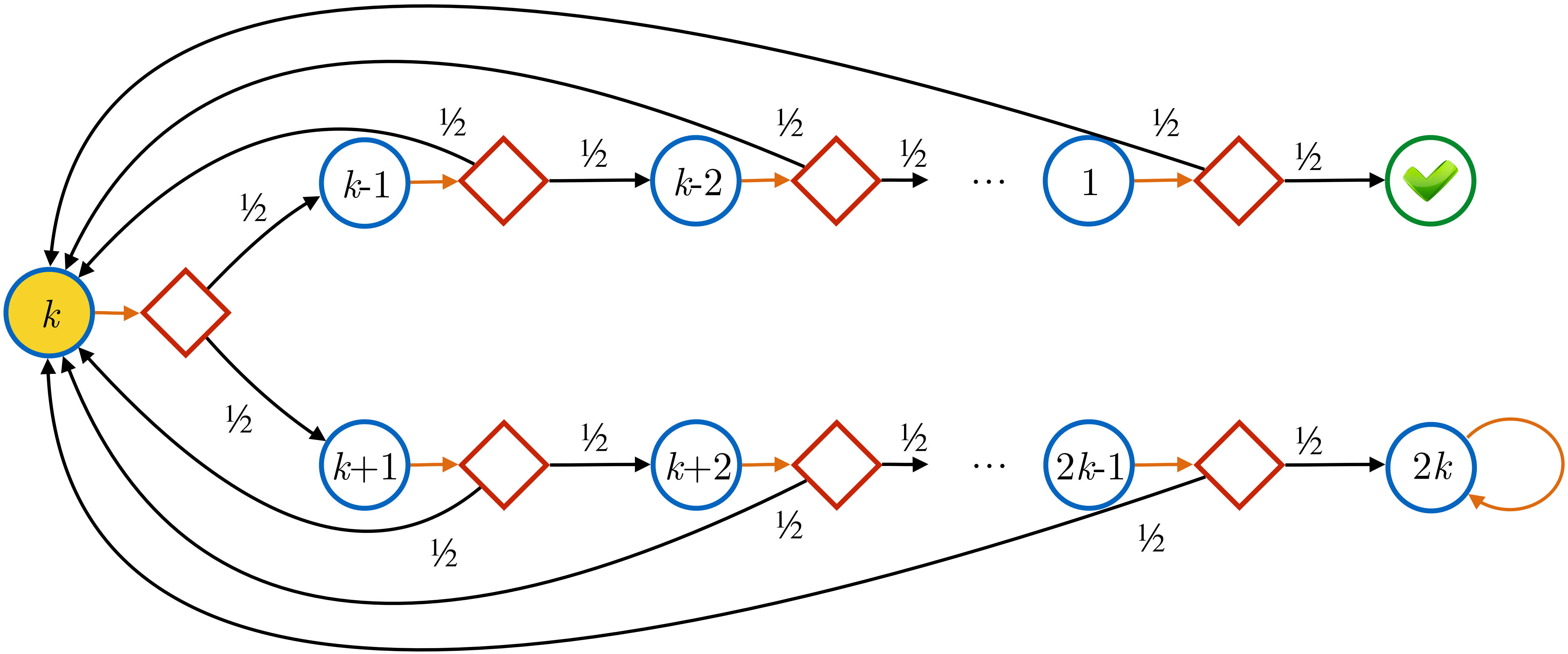
State	0	1	2	3	…	k-1	k	k+1	…	$2k$
Step 1	1	0	0	0	…	0	0	0	…	0
Step 2	1	1/2	0	0	…	0	0	0	…	0
Step 3	1	1/2	1/4	0	…	0	0	0	…	0

Value iteration: which guarantees?



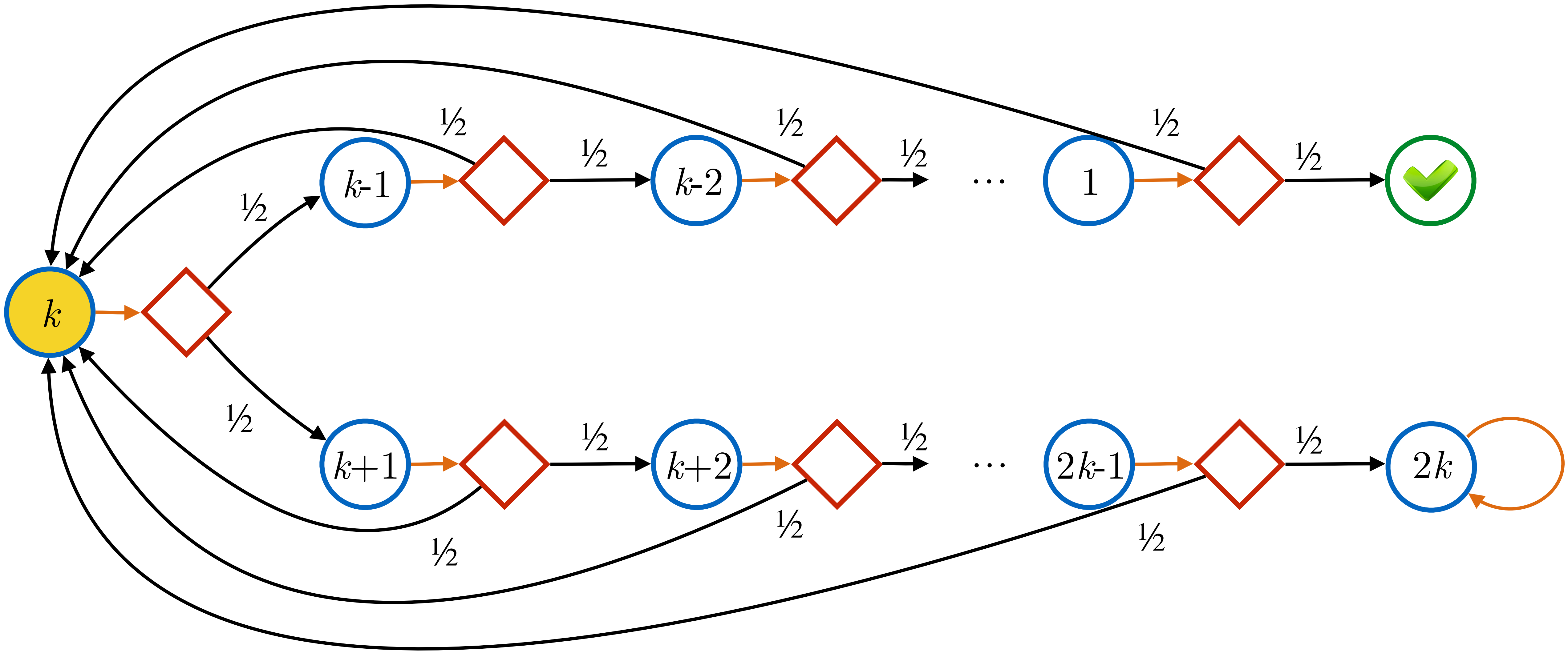
State	0	1	2	3	…	k-1	k	k+1	…	2k
Step 1	1	0	0	0	…	0	0	0	…	0
Step 2	1	1/2	0	0	…	0	0	0	…	0
Step 3	1	1/2	1/4	0	…	0	0	0	…	0
Step 4	1	1/2	1/4	1/8	…	0	0	0	…	0

Value iteration: which guarantees?



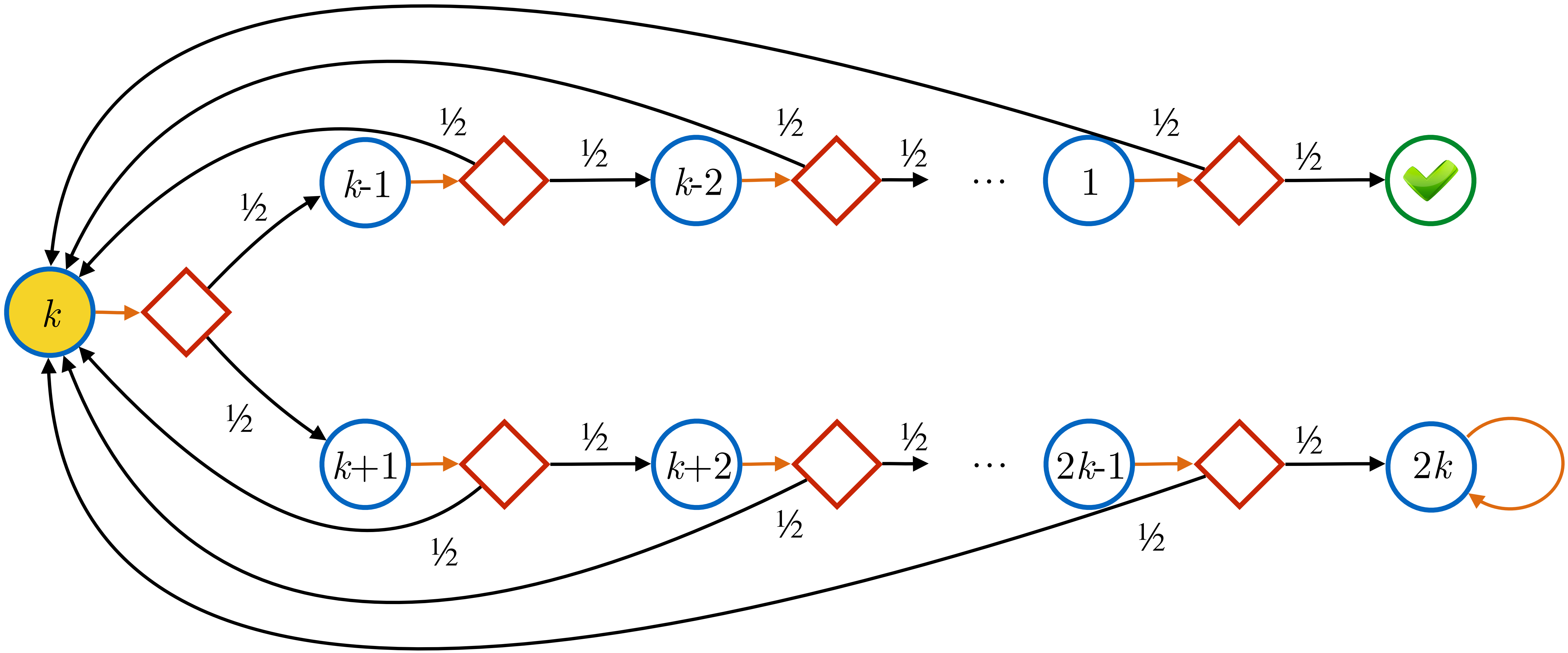
State	0	1	2	3	…	k-1	k	k+1	…	2k
Step 1	1	0	0	0	…	0	0	0	…	0
Step 2	1	1/2	0	0	…	0	0	0	…	0
Step 3	1	1/2	1/4	0	…	0	0	0	…	0
Step 4	1	1/2	1/4	1/8	…	0	0	0	…	0
…	…	…	…	…	…	…	…	…	…	…
Step k	1	1/2	1/4	1/8	…	$1/2^{k-1}$	0	0	…	0

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

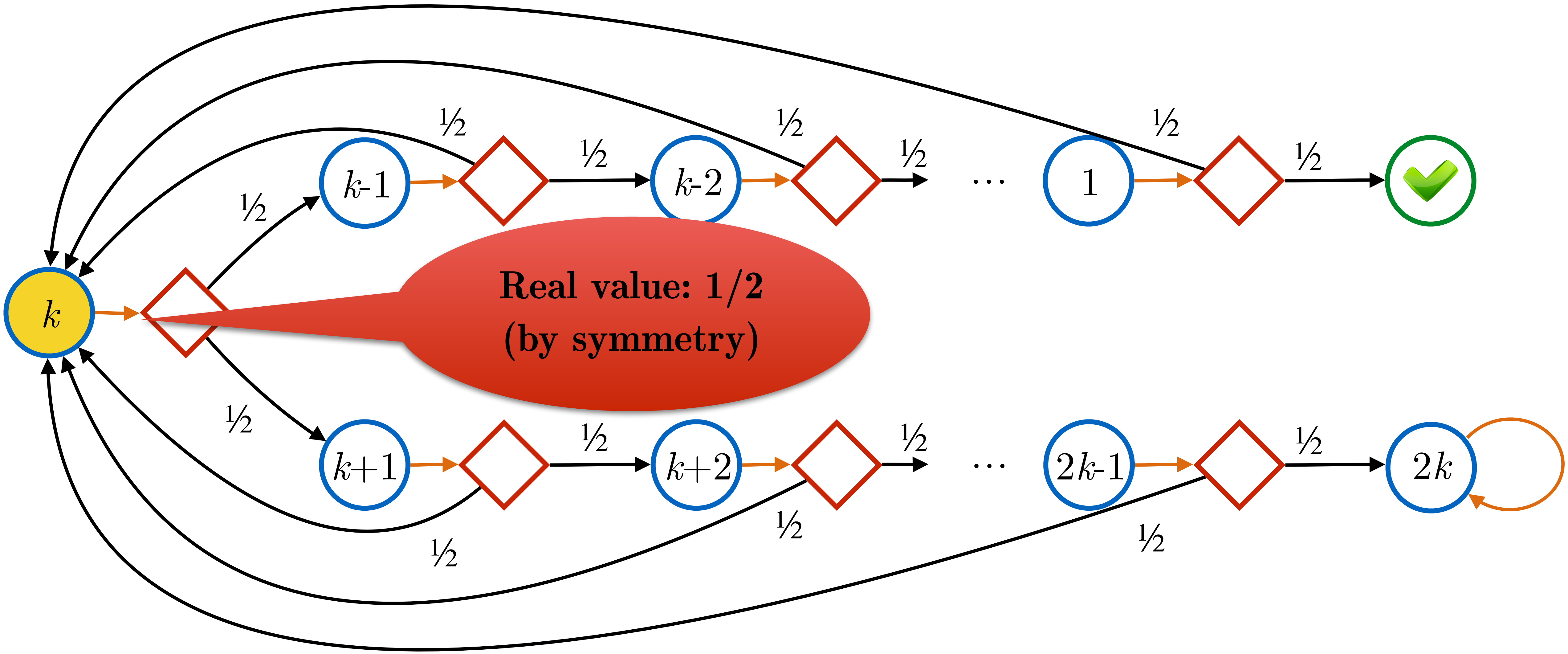
Value iteration: which guarantees?



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

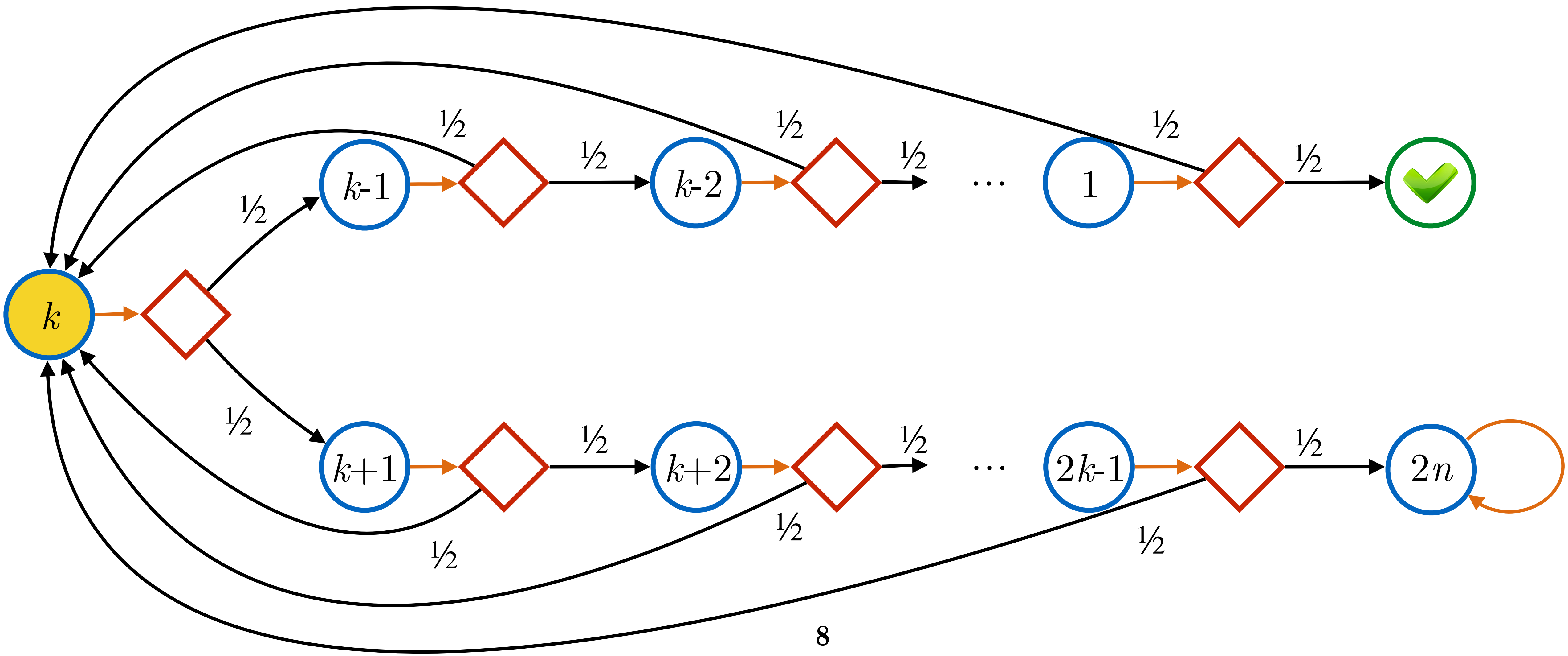
Contributions

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

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

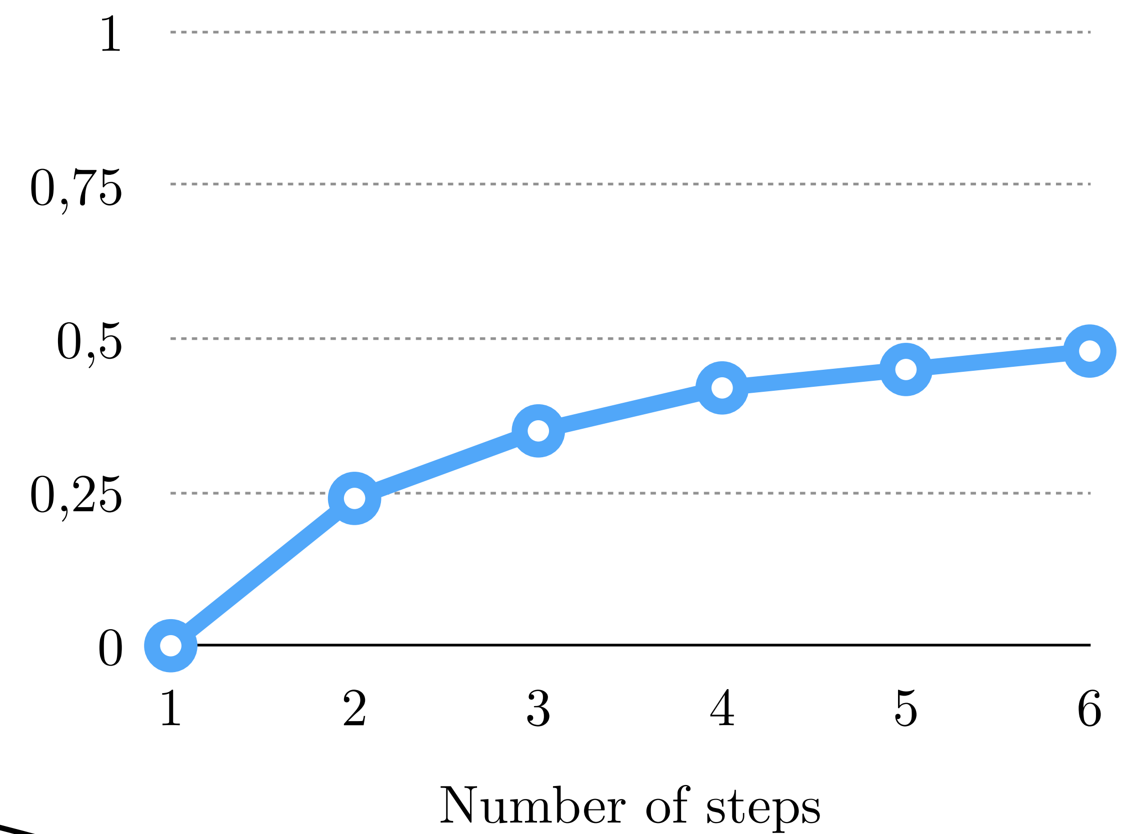
$$x_s^{(n+1)} = \max_{a \in \alpha} \sum_{s' \in S} \delta(s,a)(s') \times x_{s'}^{(n)}$$

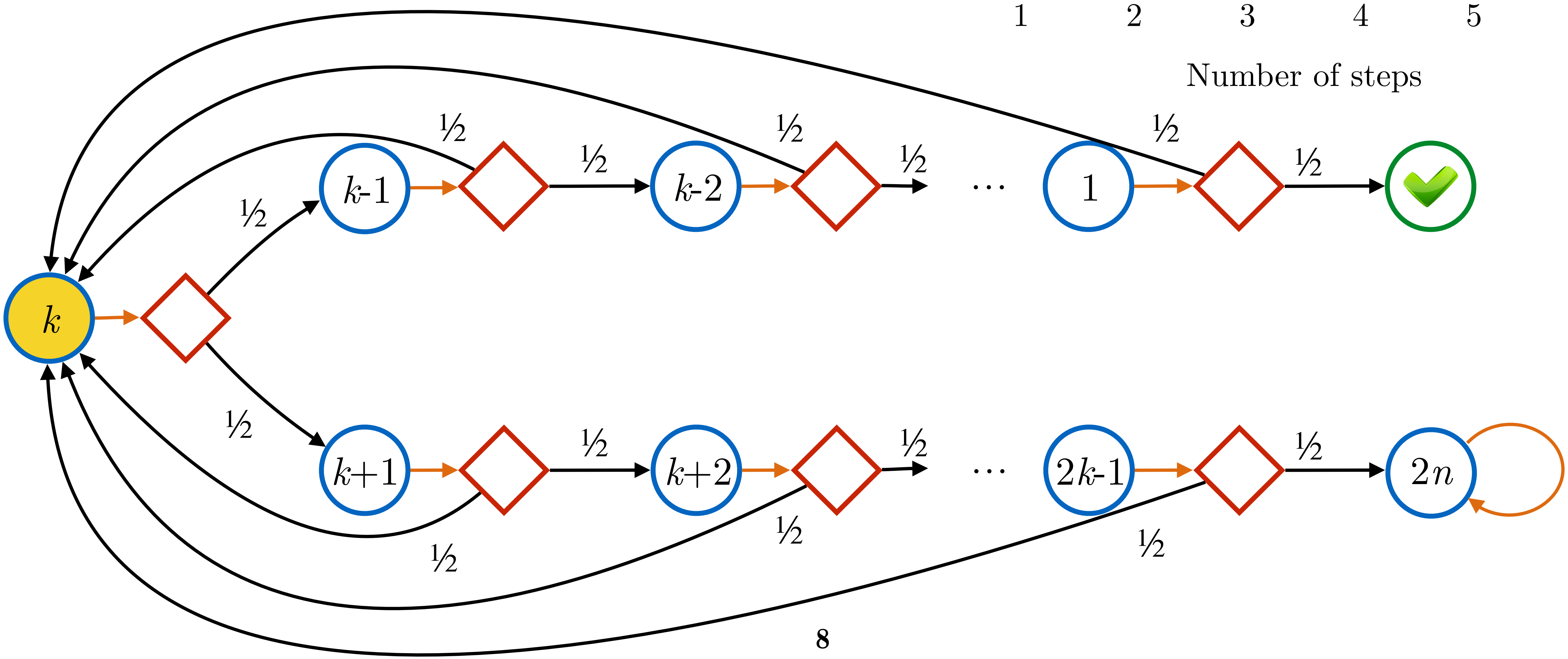


Interval iteration

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

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



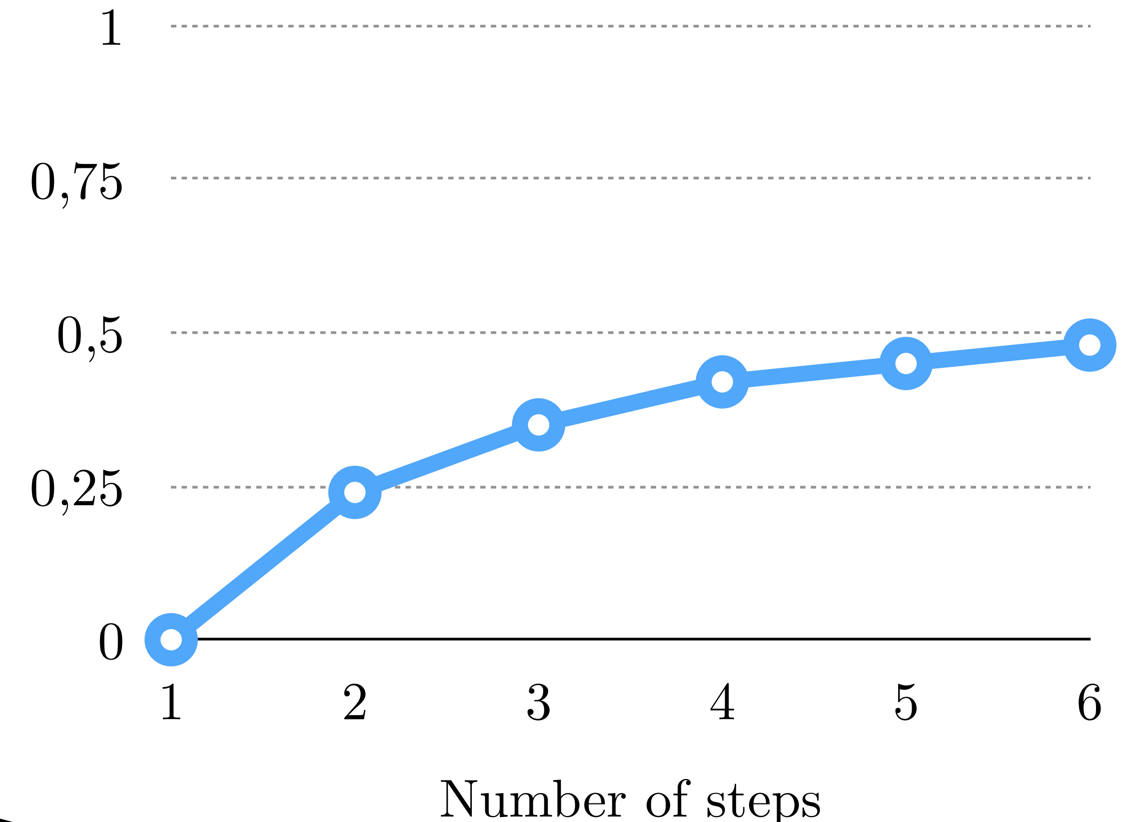


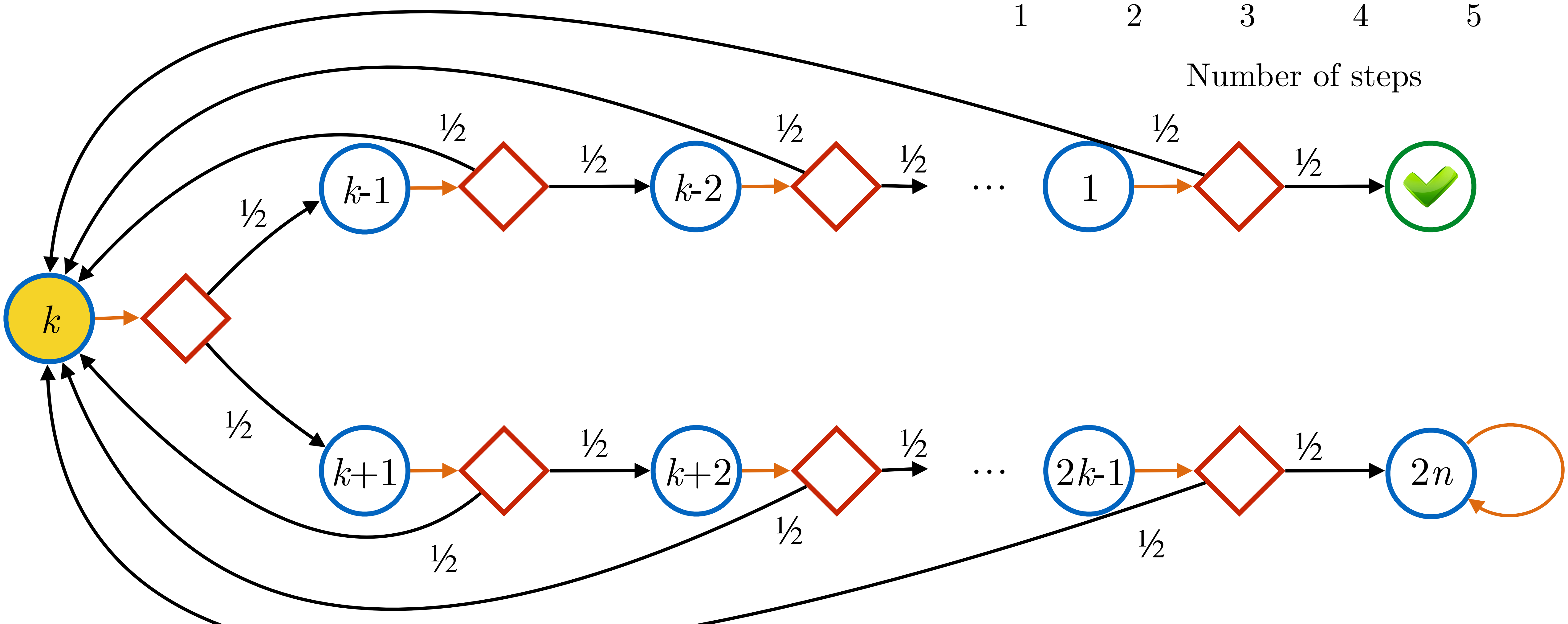
Interval iteration

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

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

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



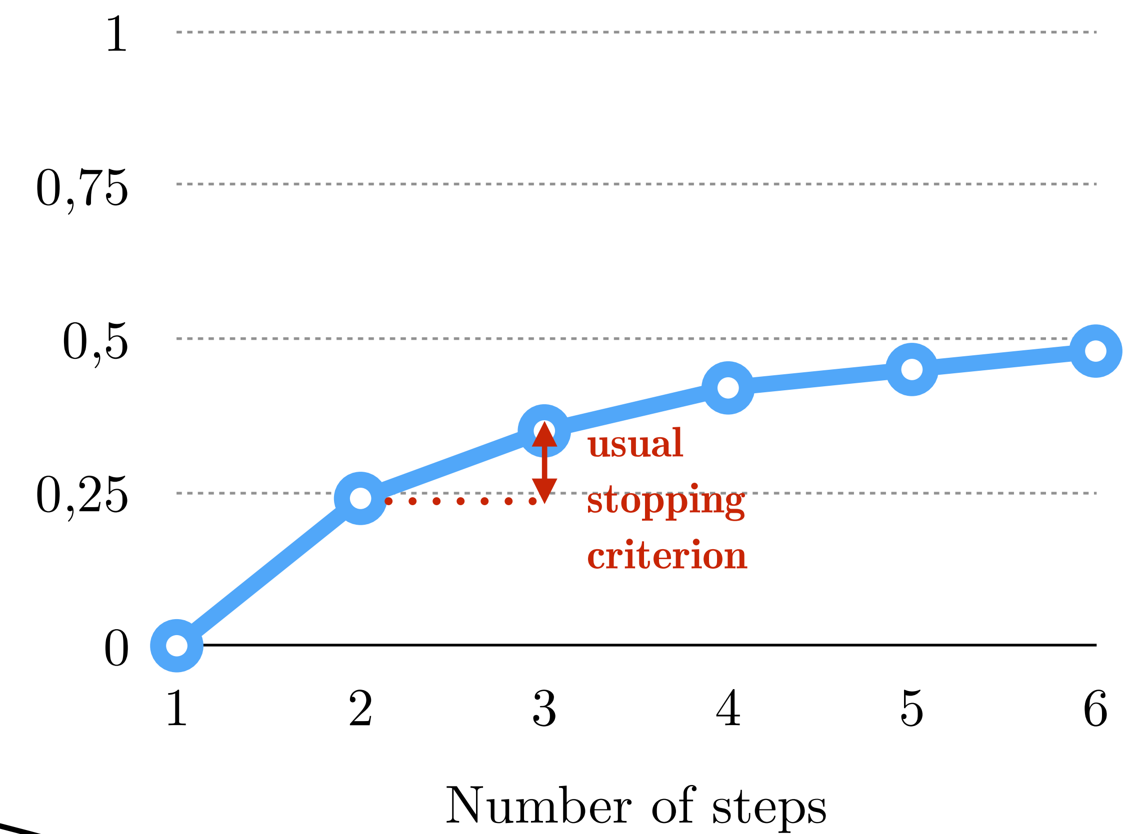


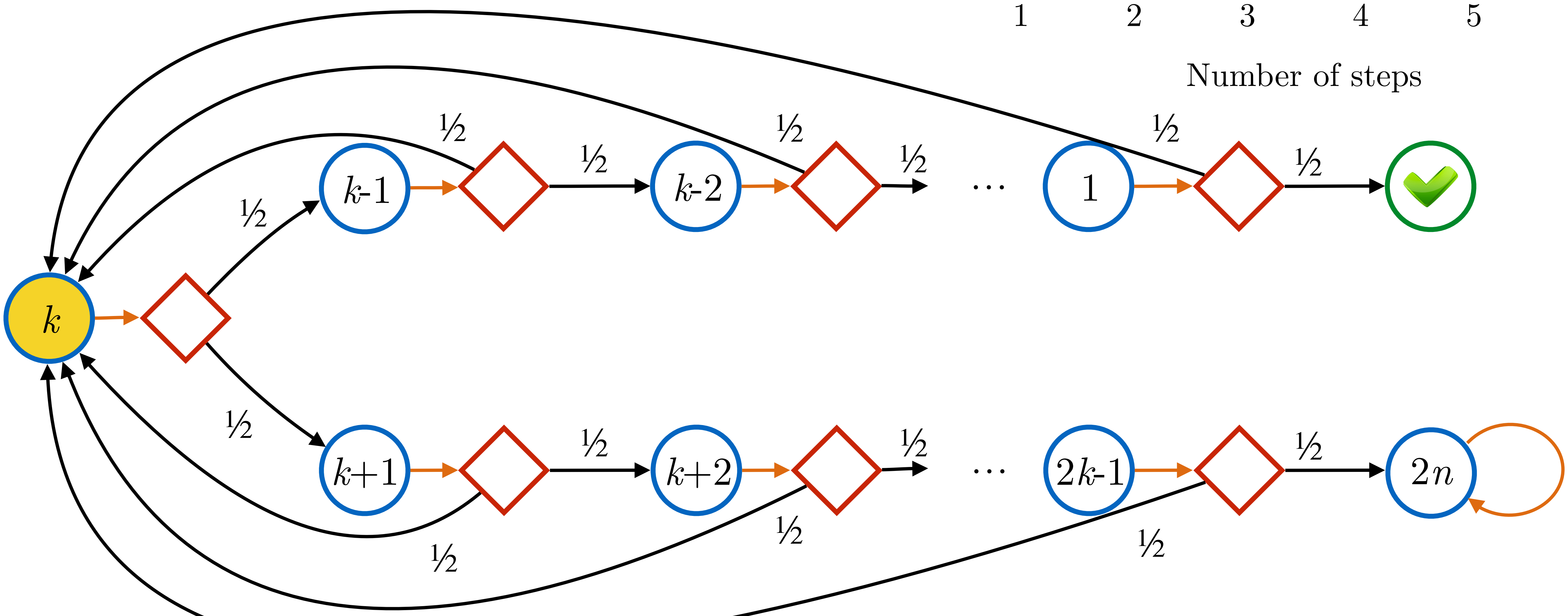
Interval iteration

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

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

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



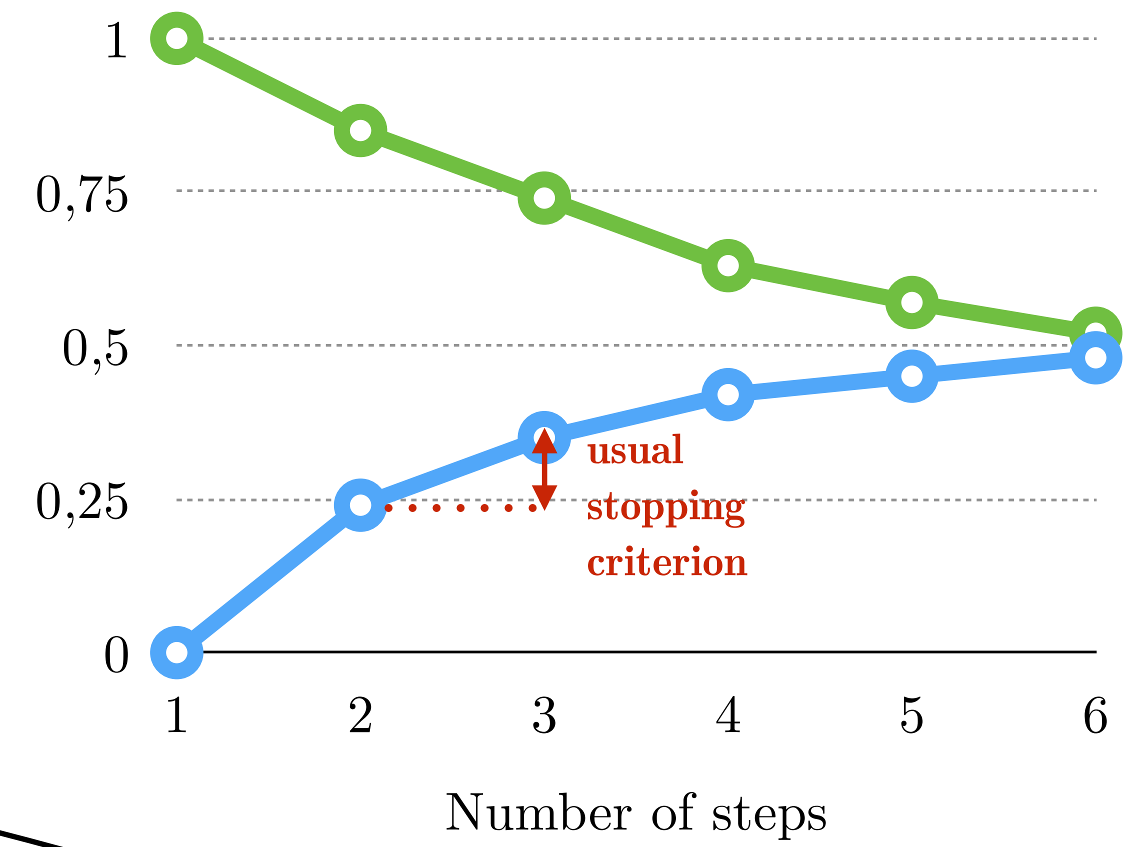


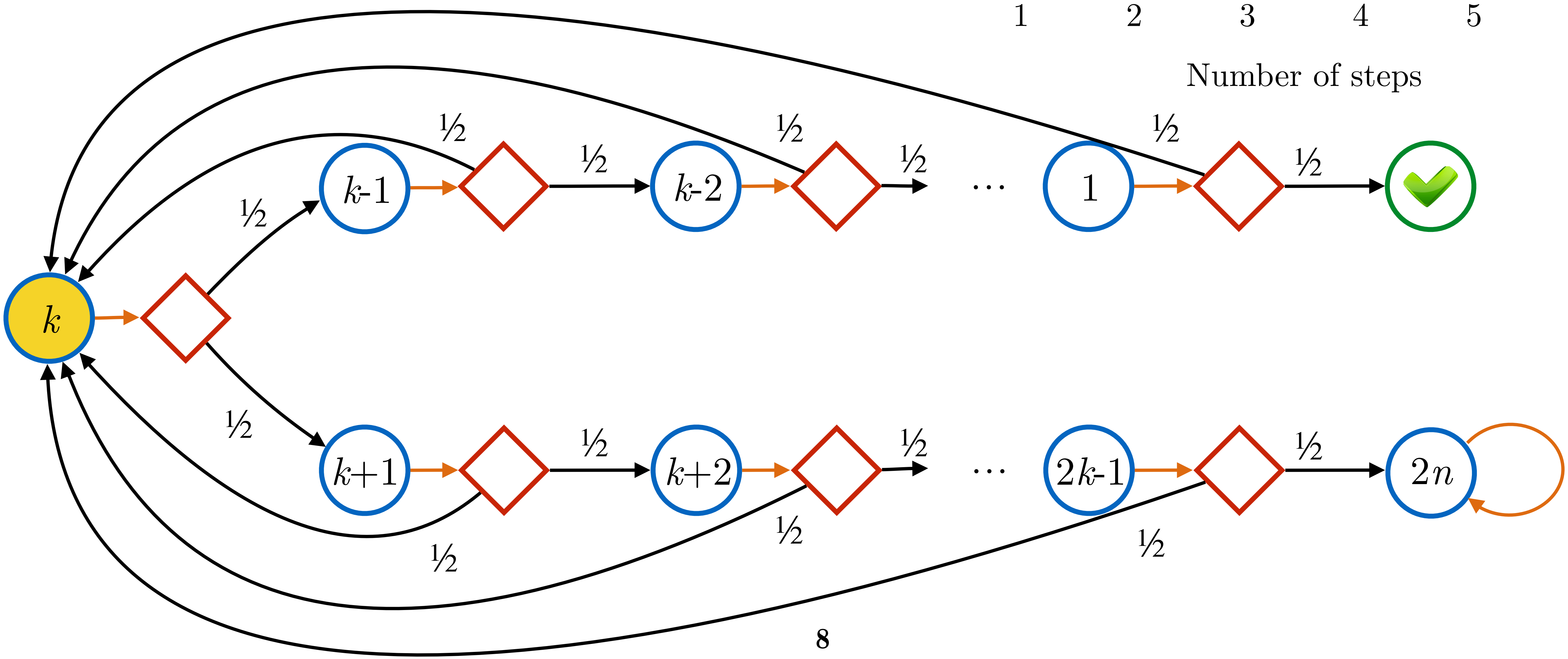
Interval iteration

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

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

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



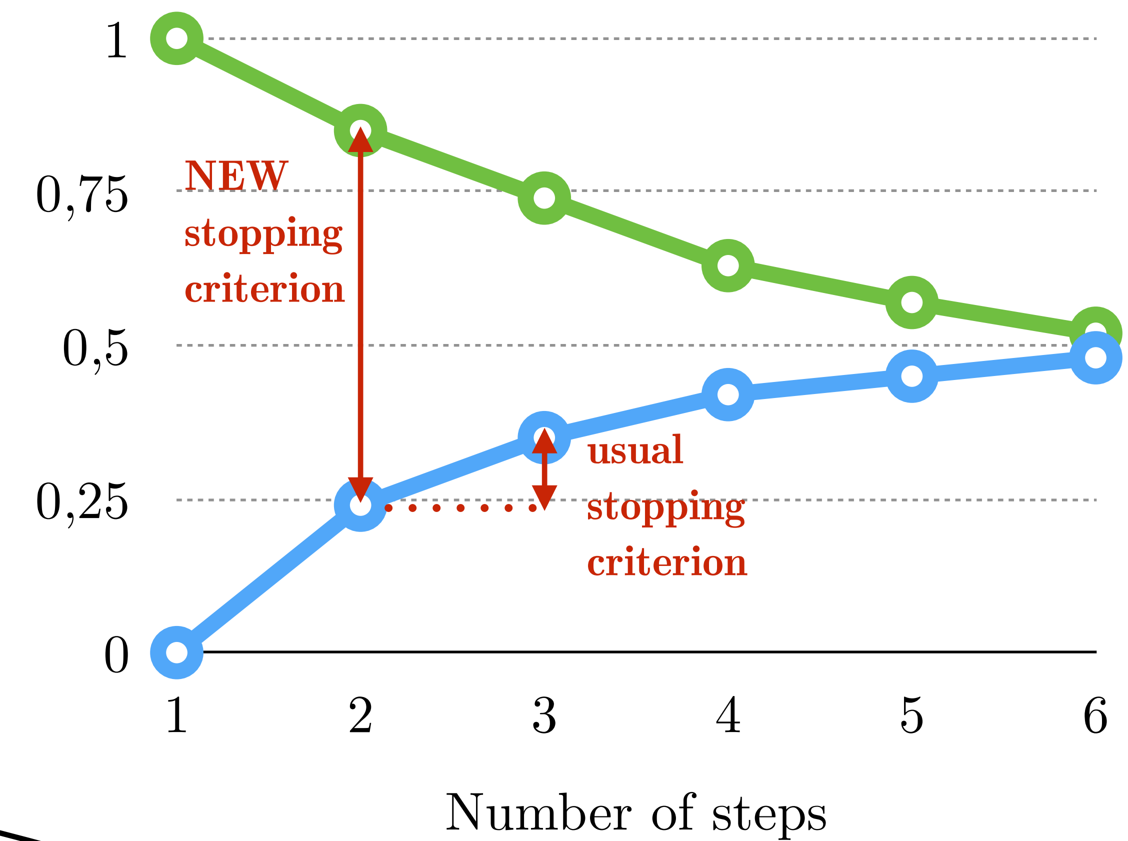


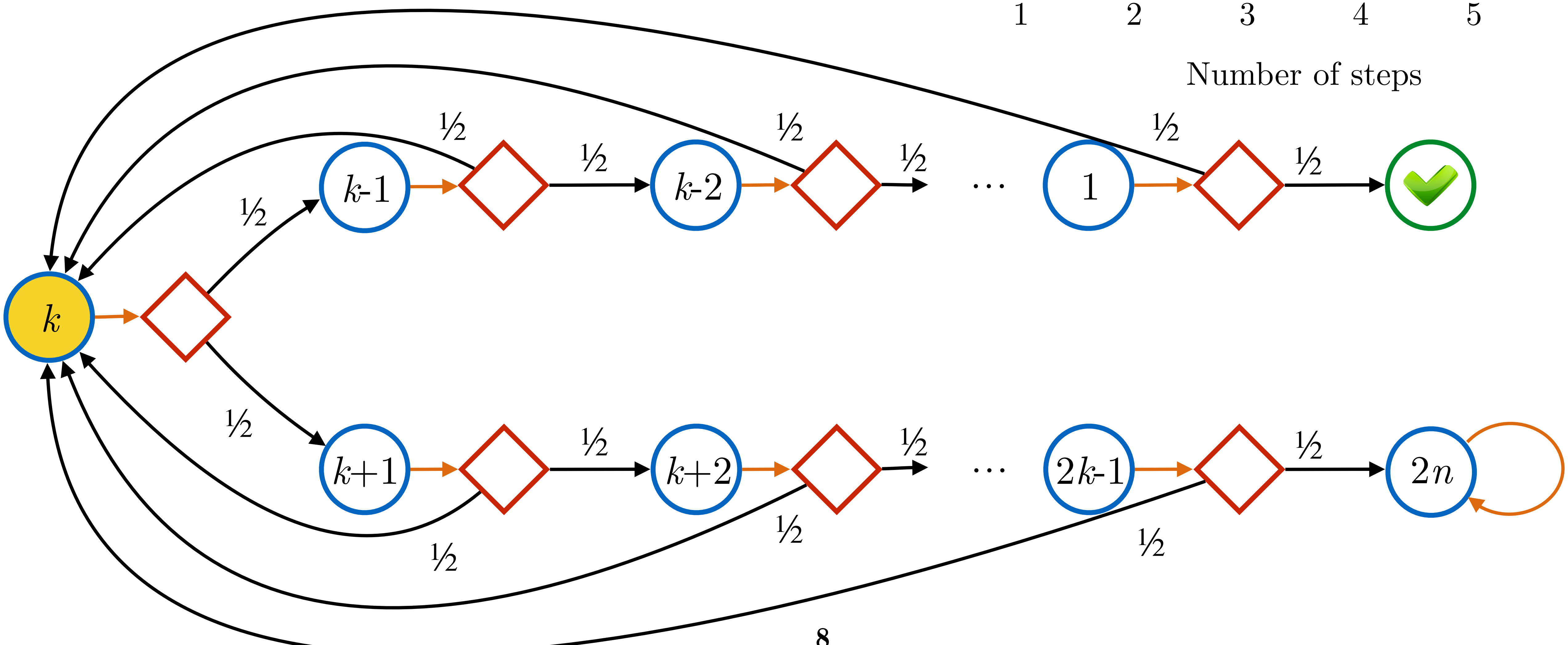
Interval iteration

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

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

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



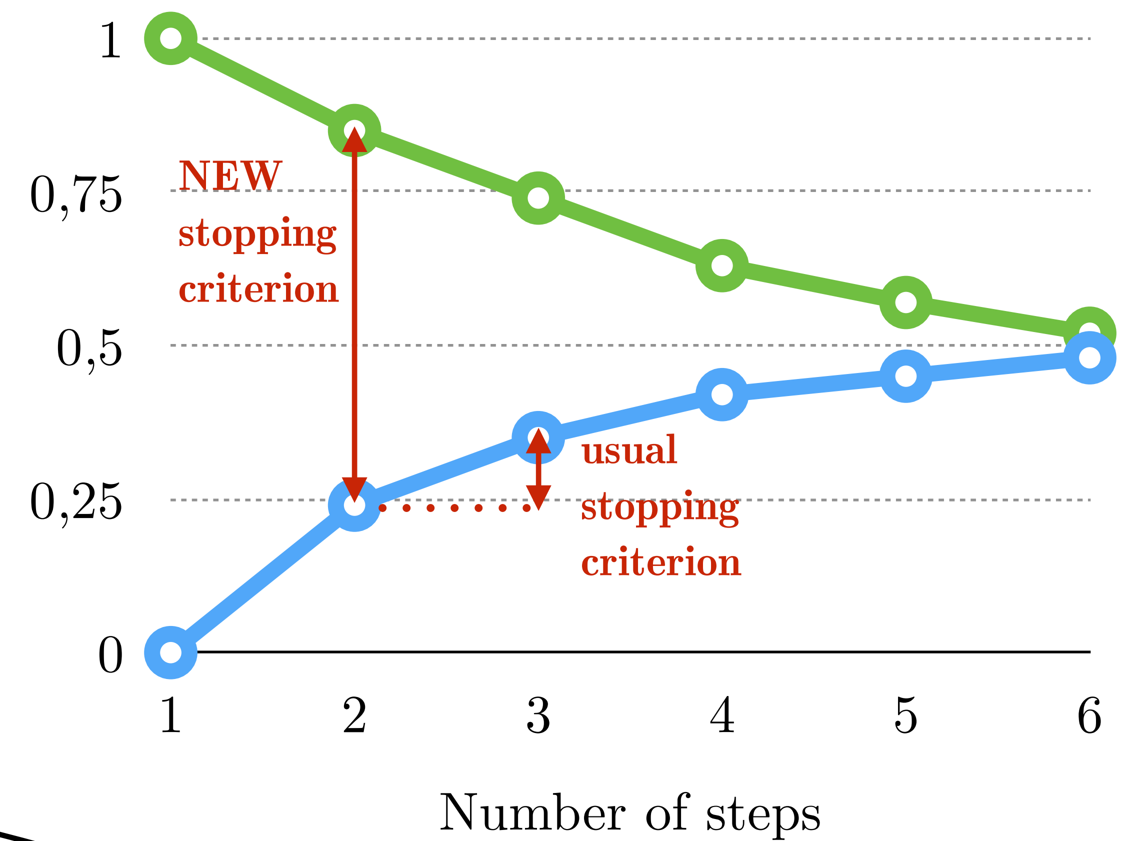


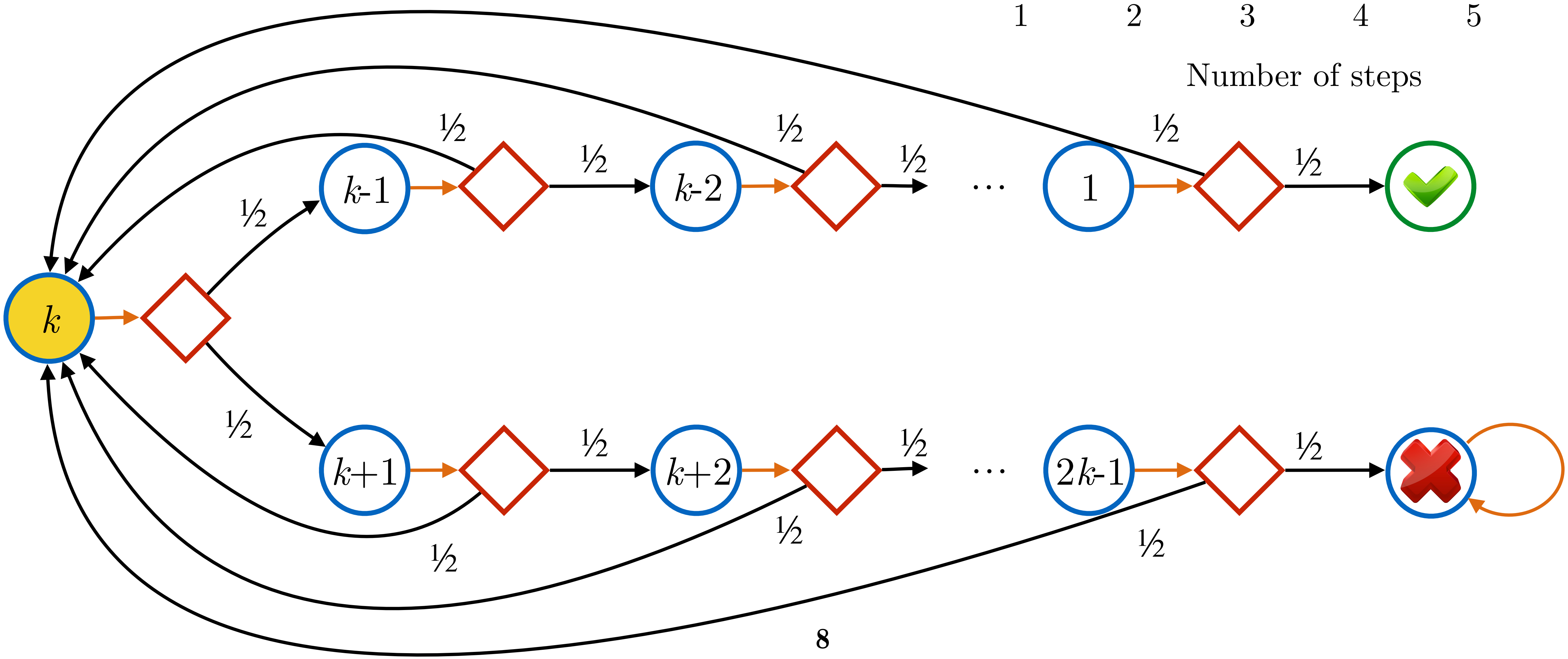
Interval iteration

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

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

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



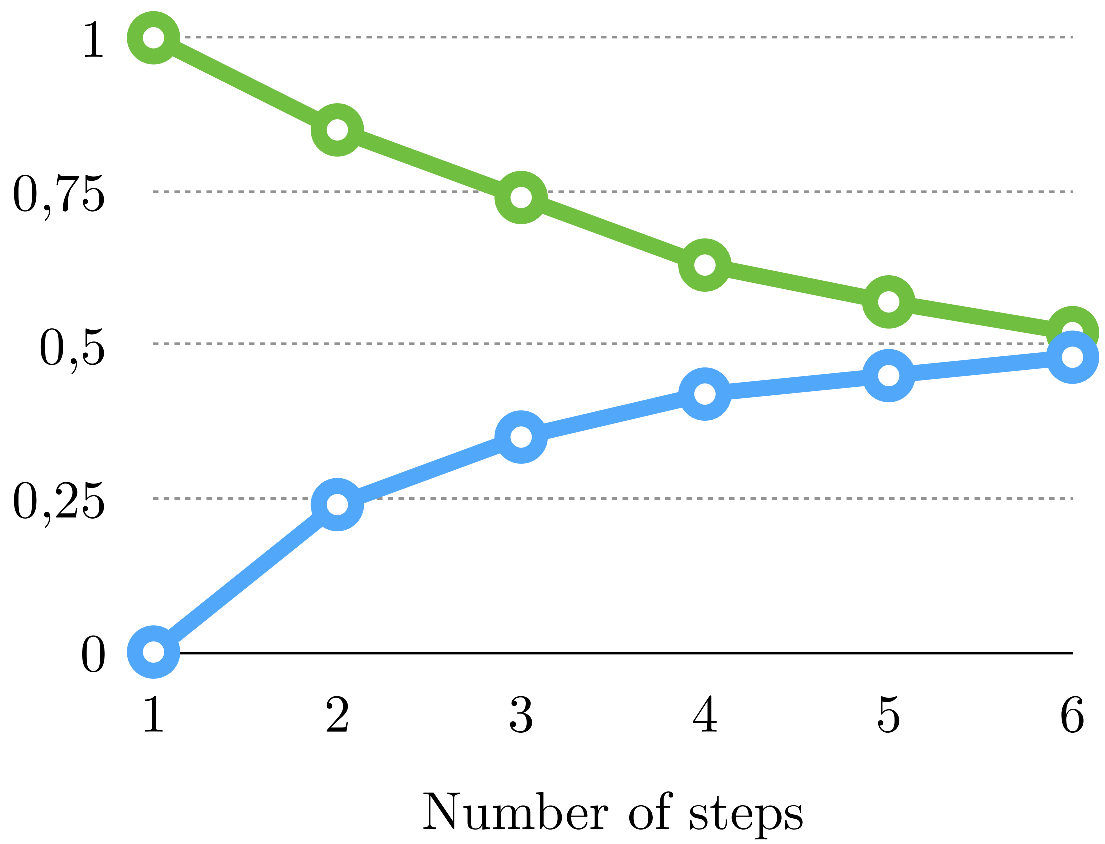


Fixed point characterization

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

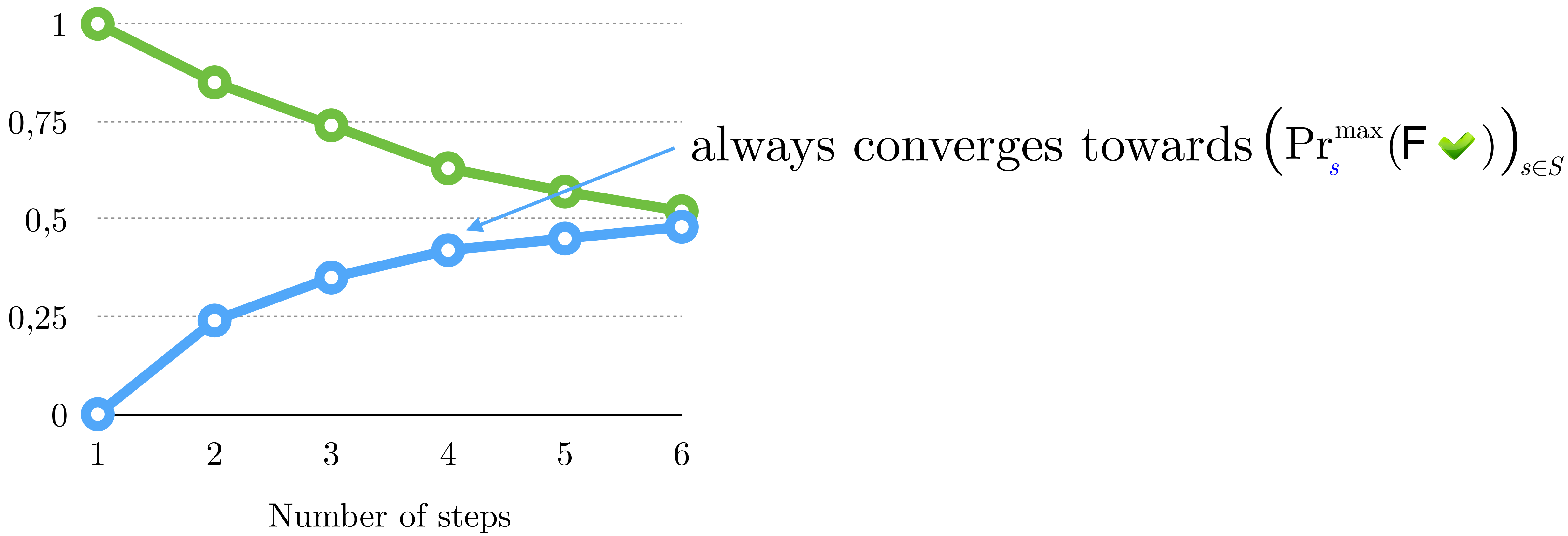
Fixed point characterization

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



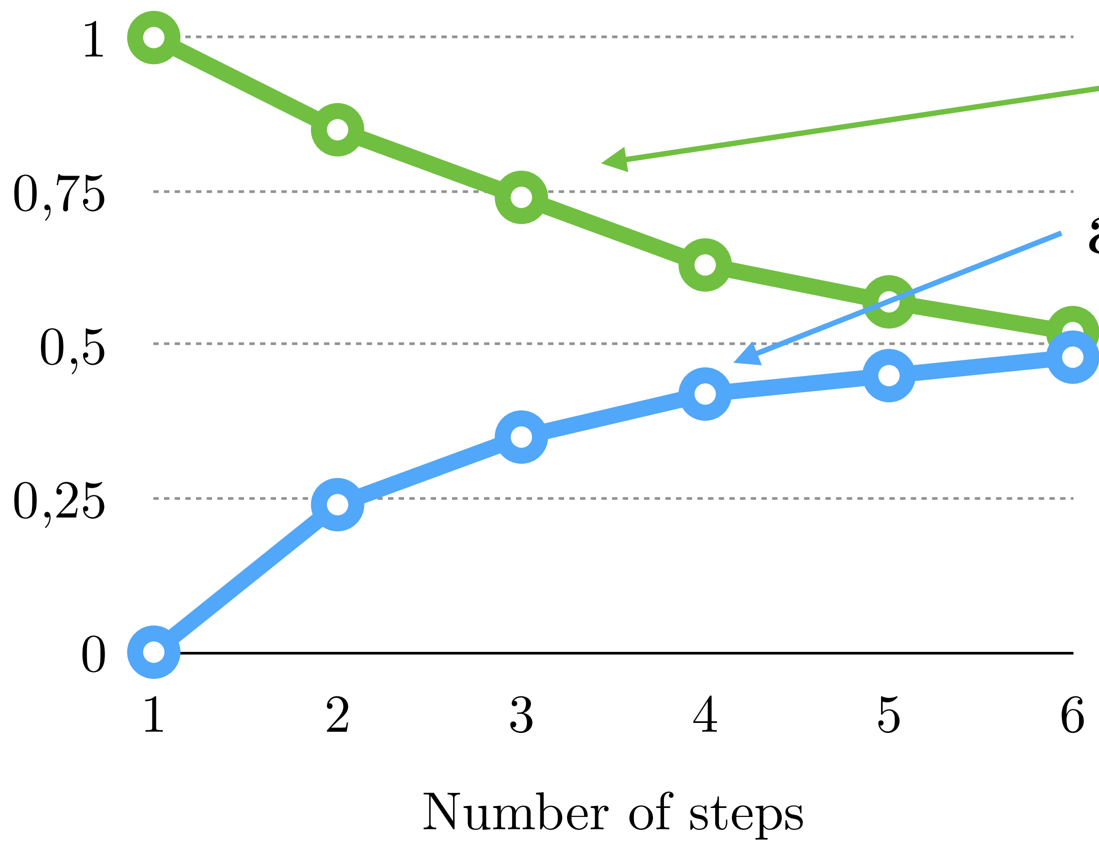
Fixed point characterization

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



Fixed point characterization

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

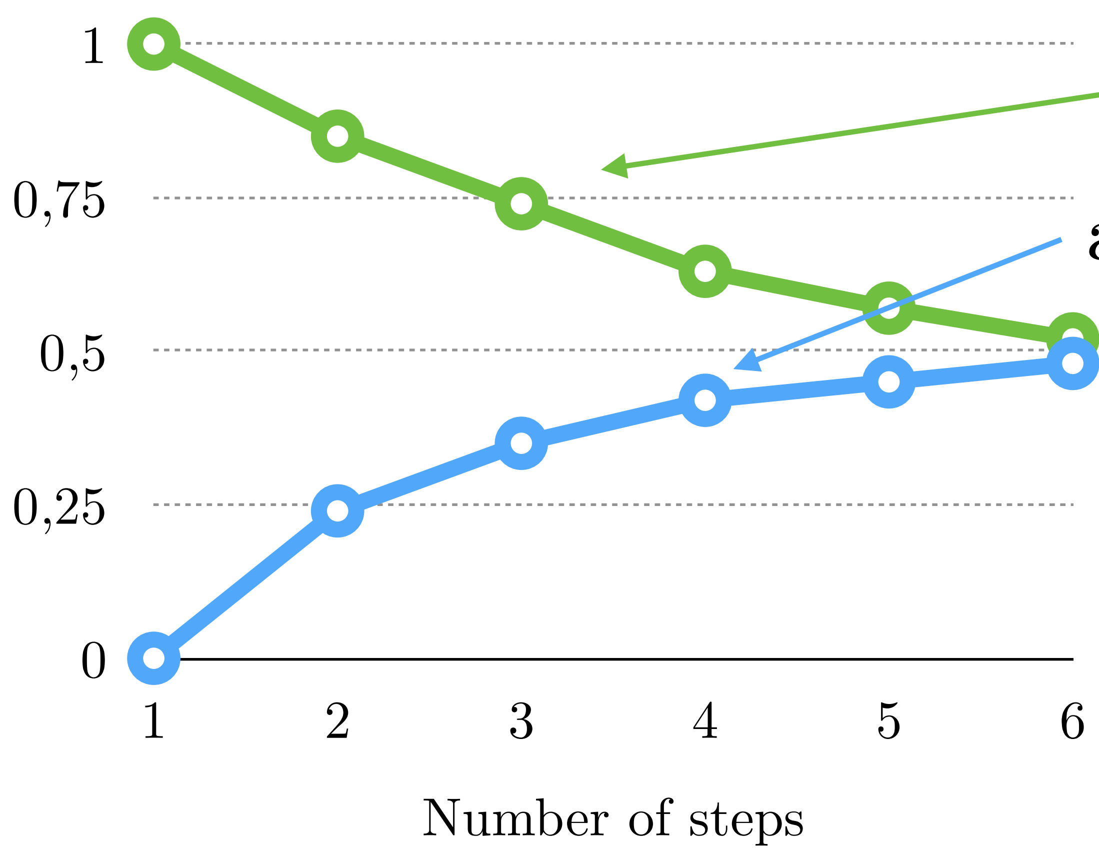


not always...!

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

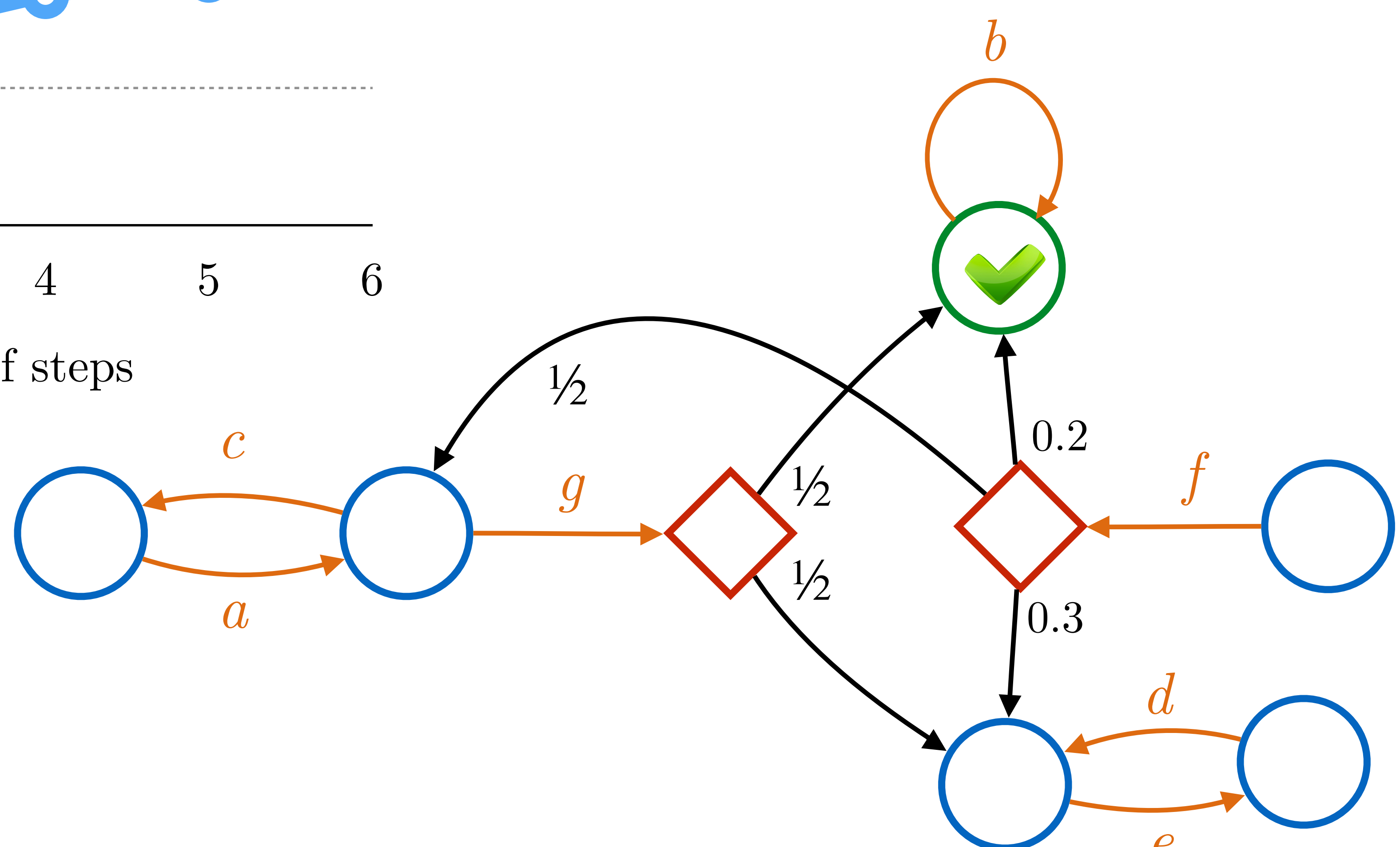
Fixed point characterization

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



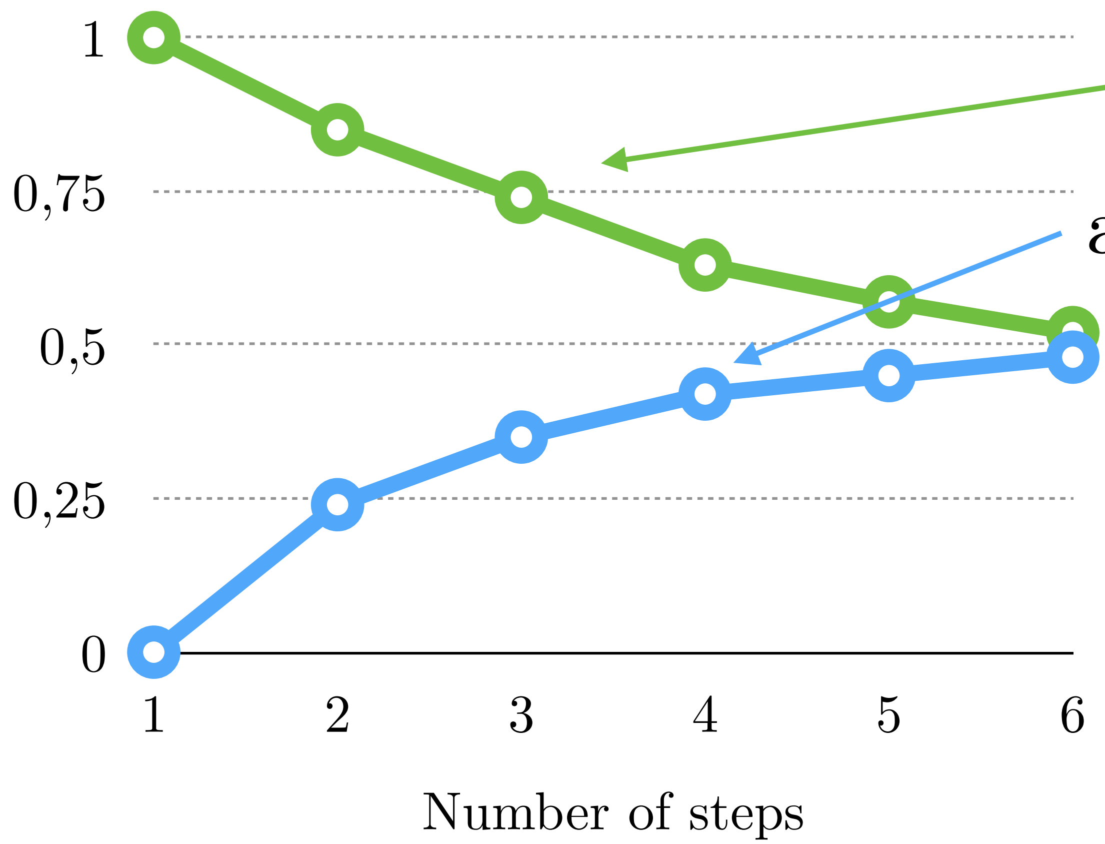
not always...!

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



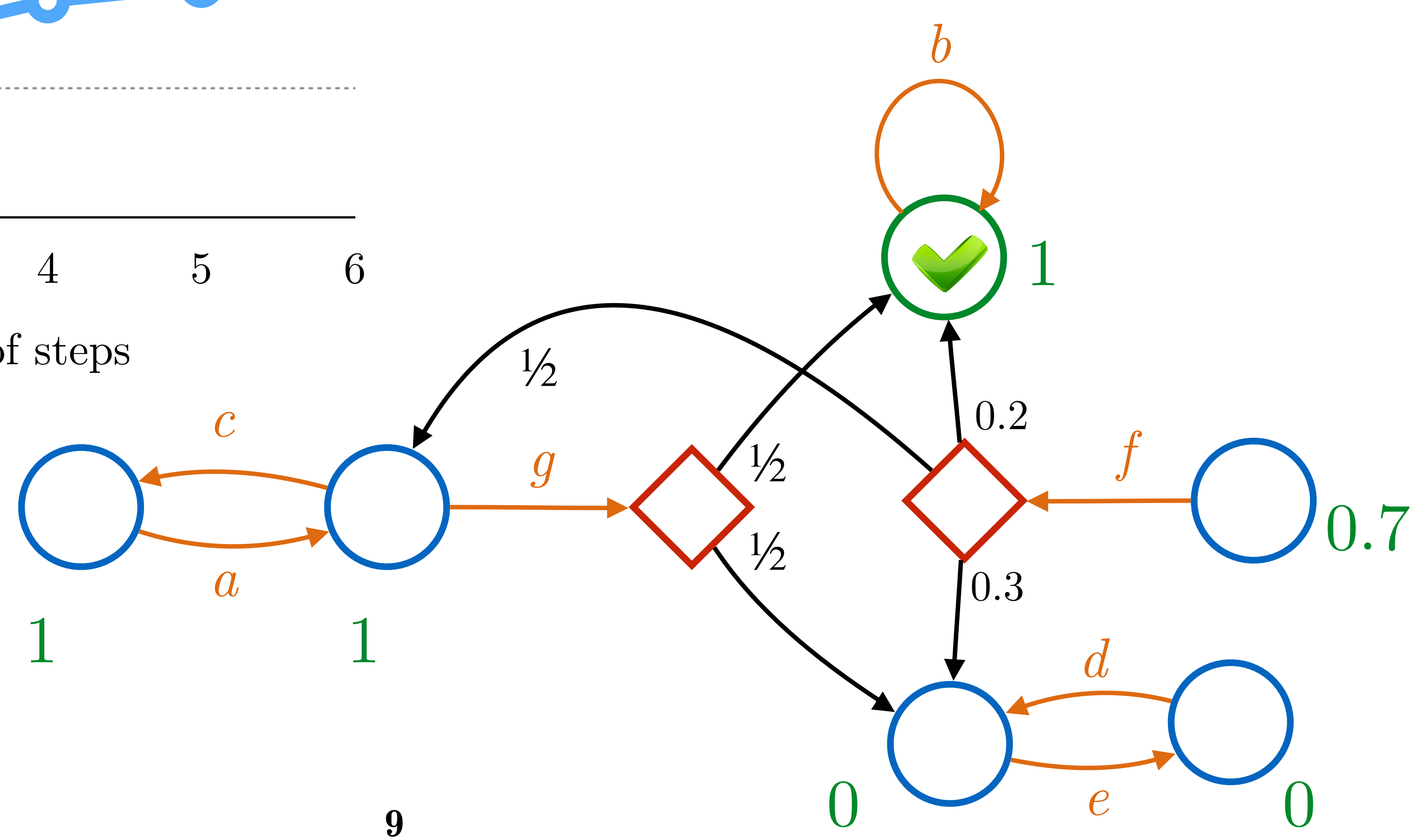
Fixed point characterization

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



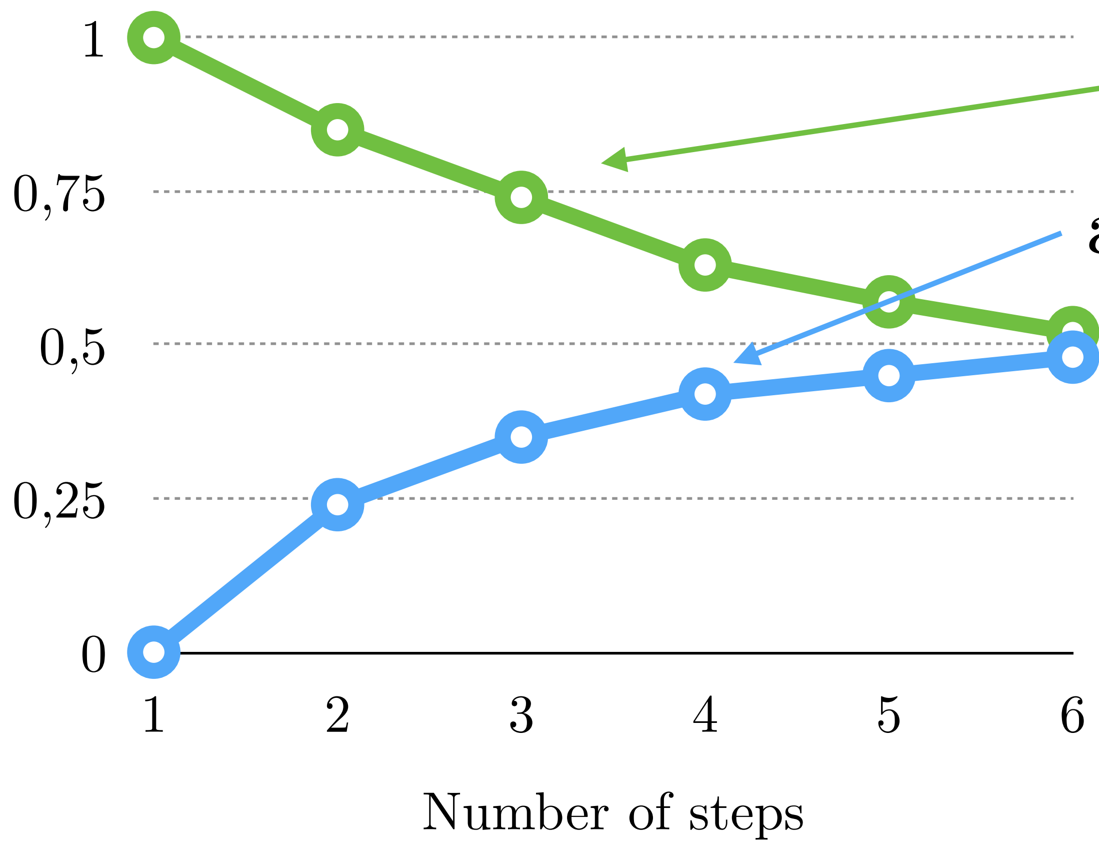
not always...!

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



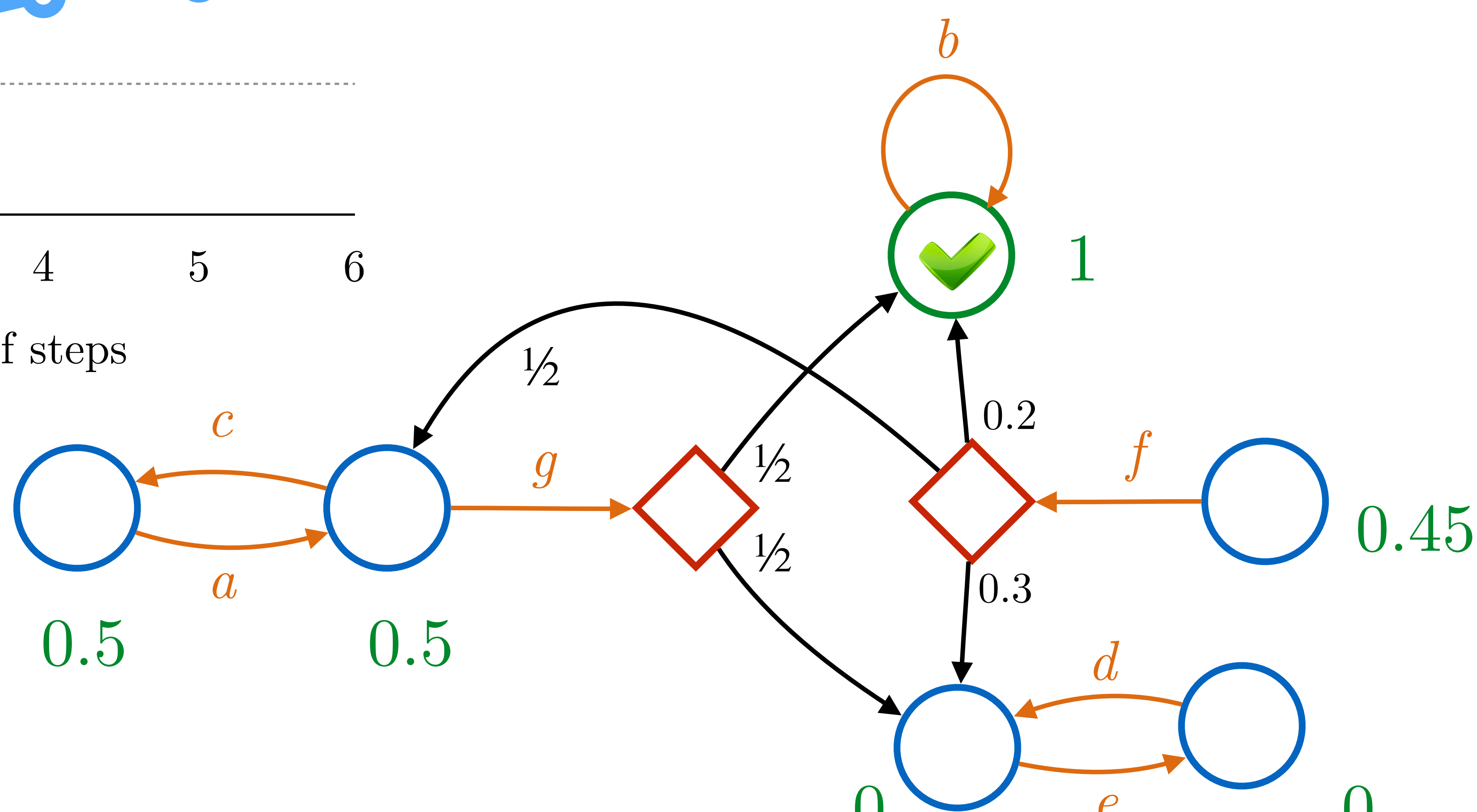
Fixed point characterization

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



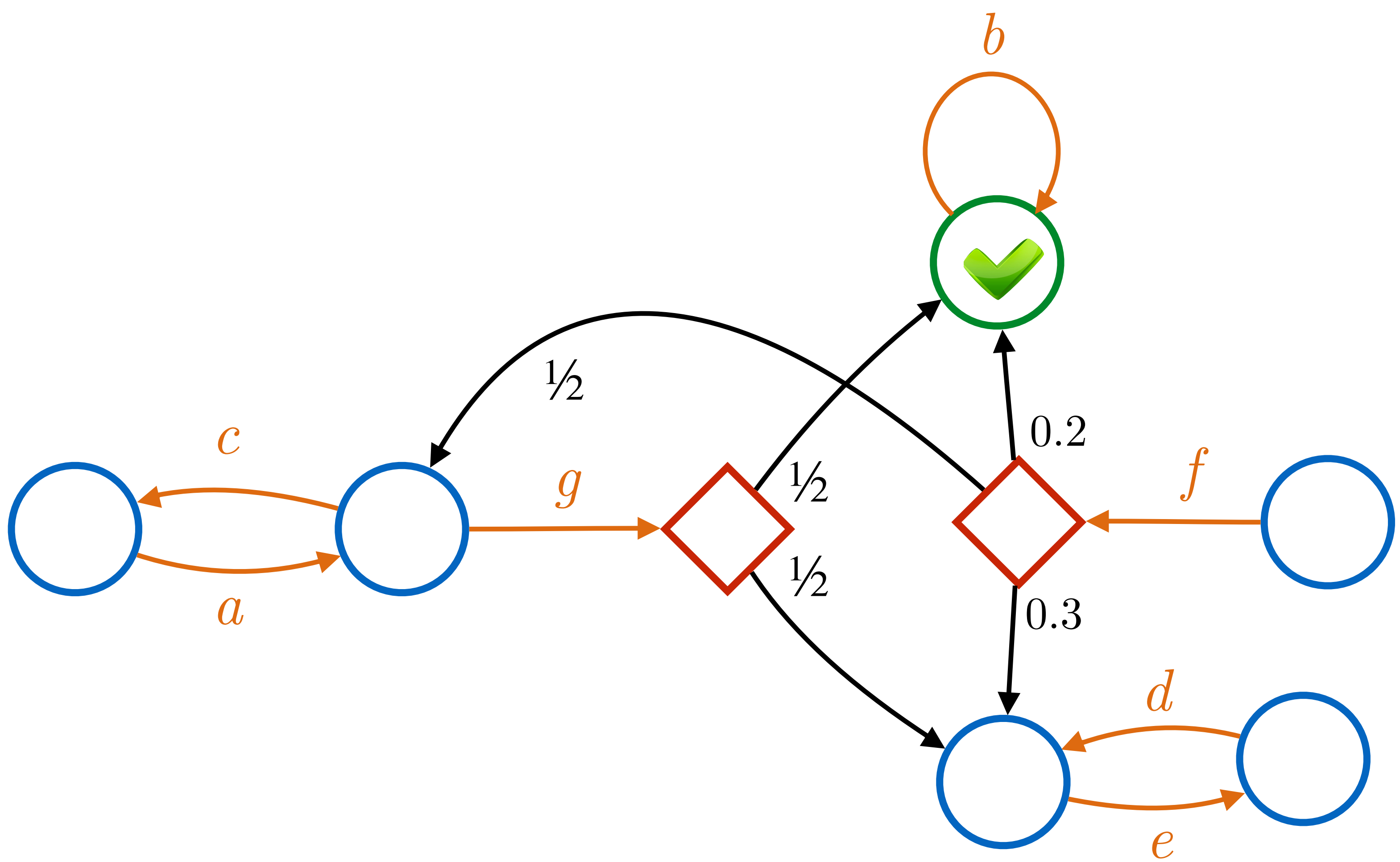
not always...!

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



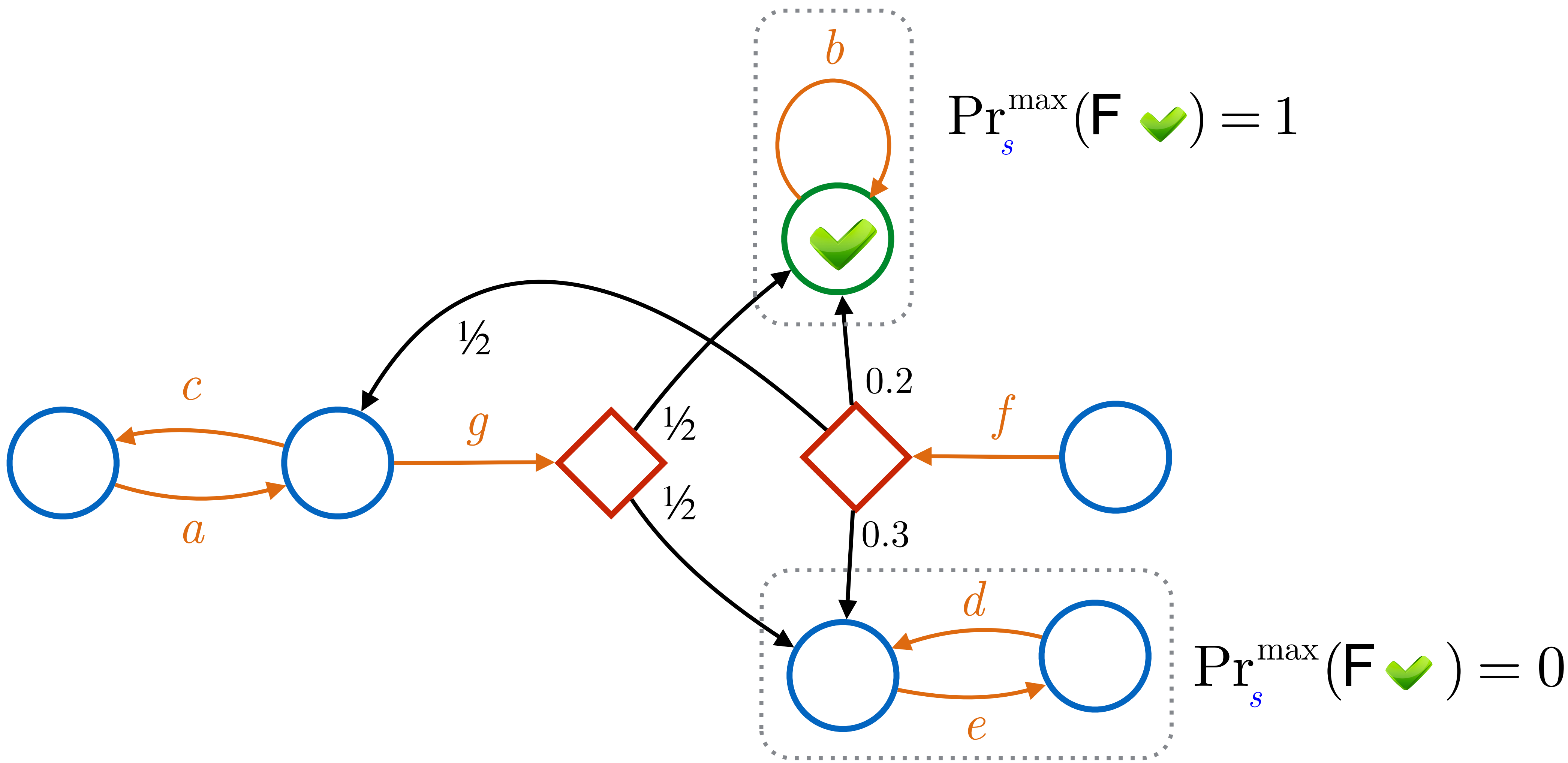
Solution: ensure uniqueness!

Usual techniques applied for MDPs do not apply…



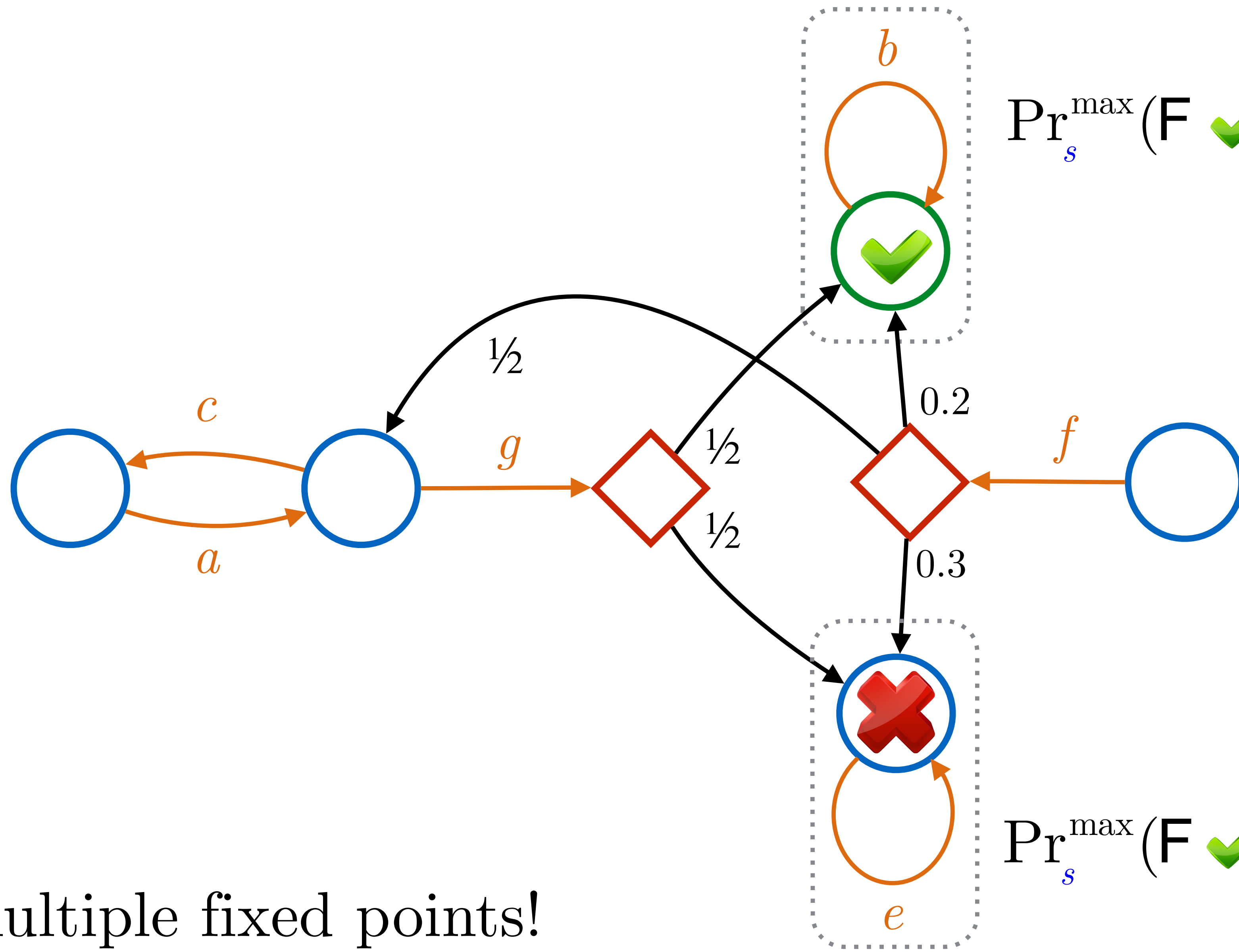
Solution: ensure uniqueness!

Usual techniques applied for MDPs do not apply…



Solution: ensure uniqueness!

Usual techniques applied for MDPs do not apply…



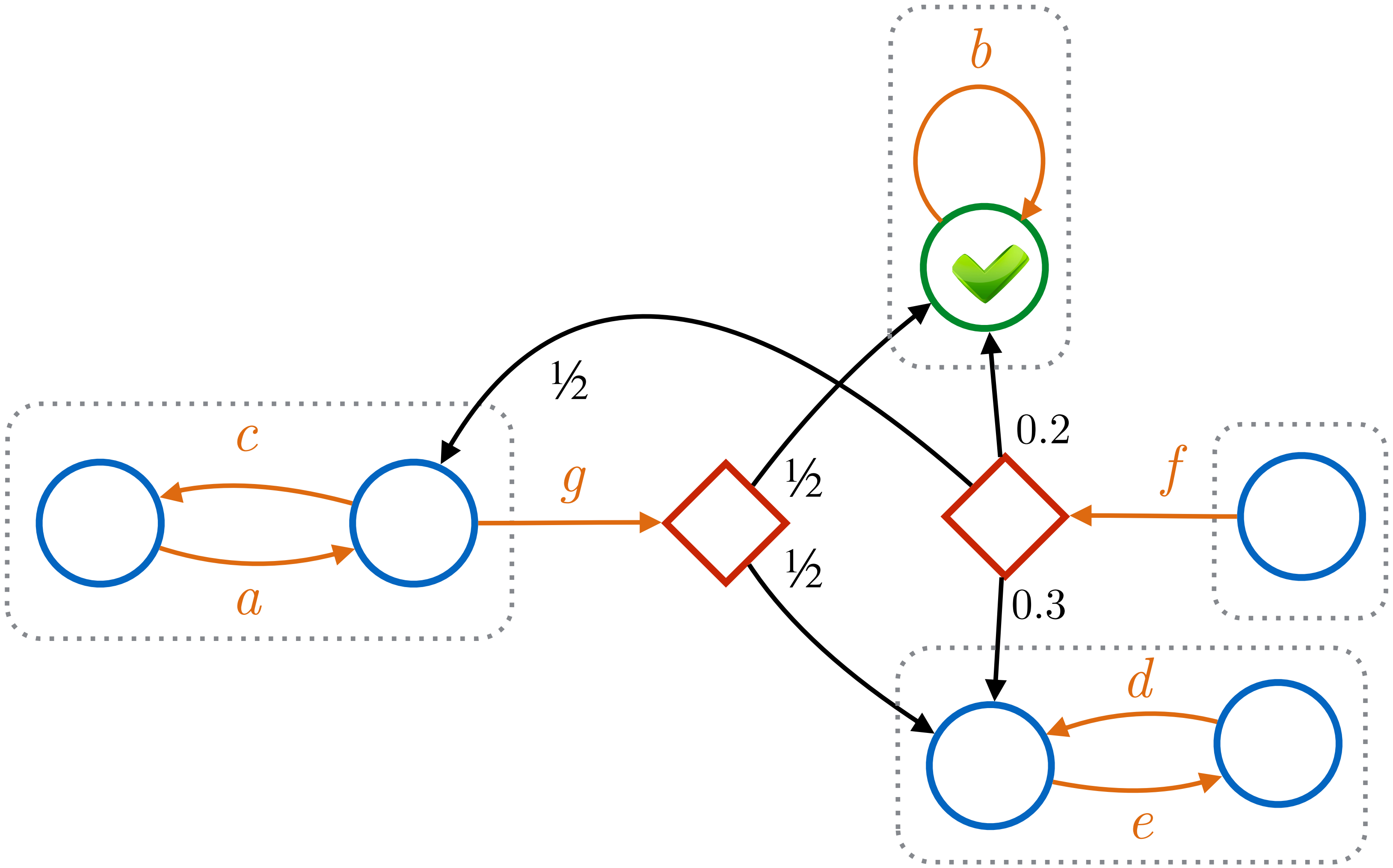
$\Pr_s^{\max}(\mathsf{F}\ ✓) = 1$

$\Pr_s^{\max}(\mathsf{F}\ ✓) = 0$

Still multiple fixed points!

Solution: ensure uniqueness!

Usual techniques applied for MDPs do not apply…

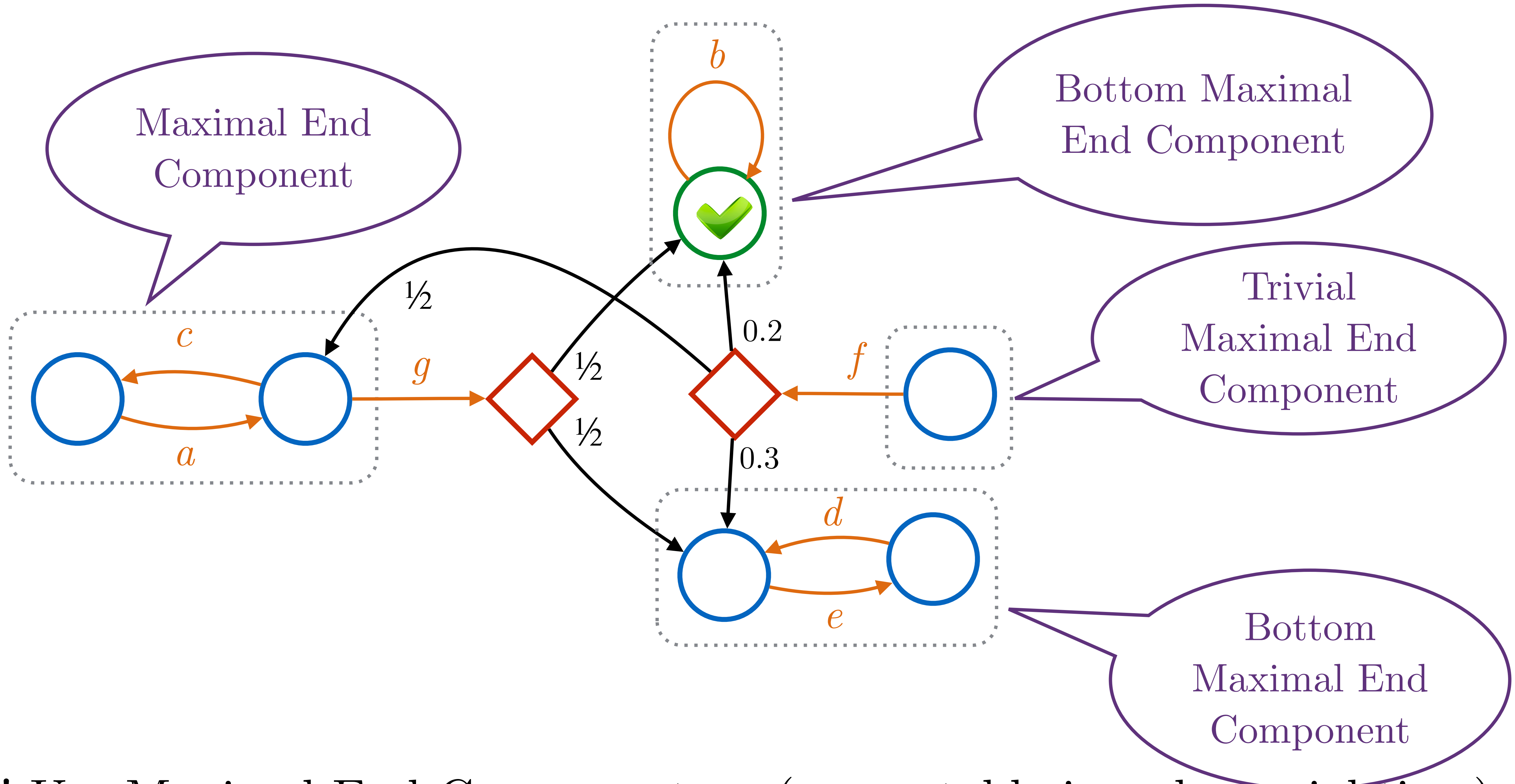


NEW! Use Maximal End Components… (computable in polynomial time)

[de Alfaro, 1997]

Solution: ensure uniqueness!

Usual techniques applied for MDPs do not apply…

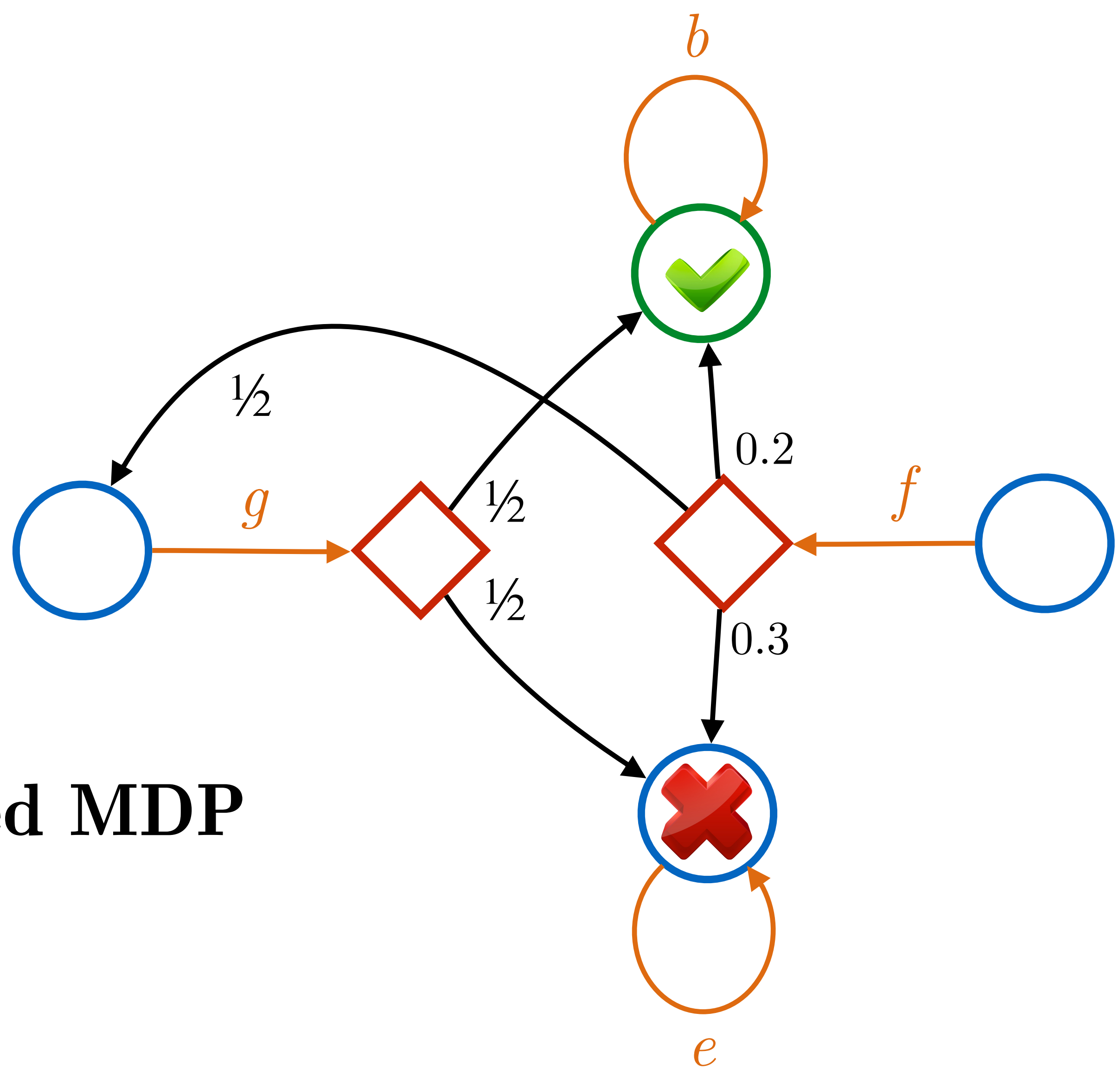


NEW! Use Maximal End Components… (computable in polynomial time)

[de Alfaro, 1997]

Solution: ensure uniqueness!

Usual techniques applied for MDPs do not apply…

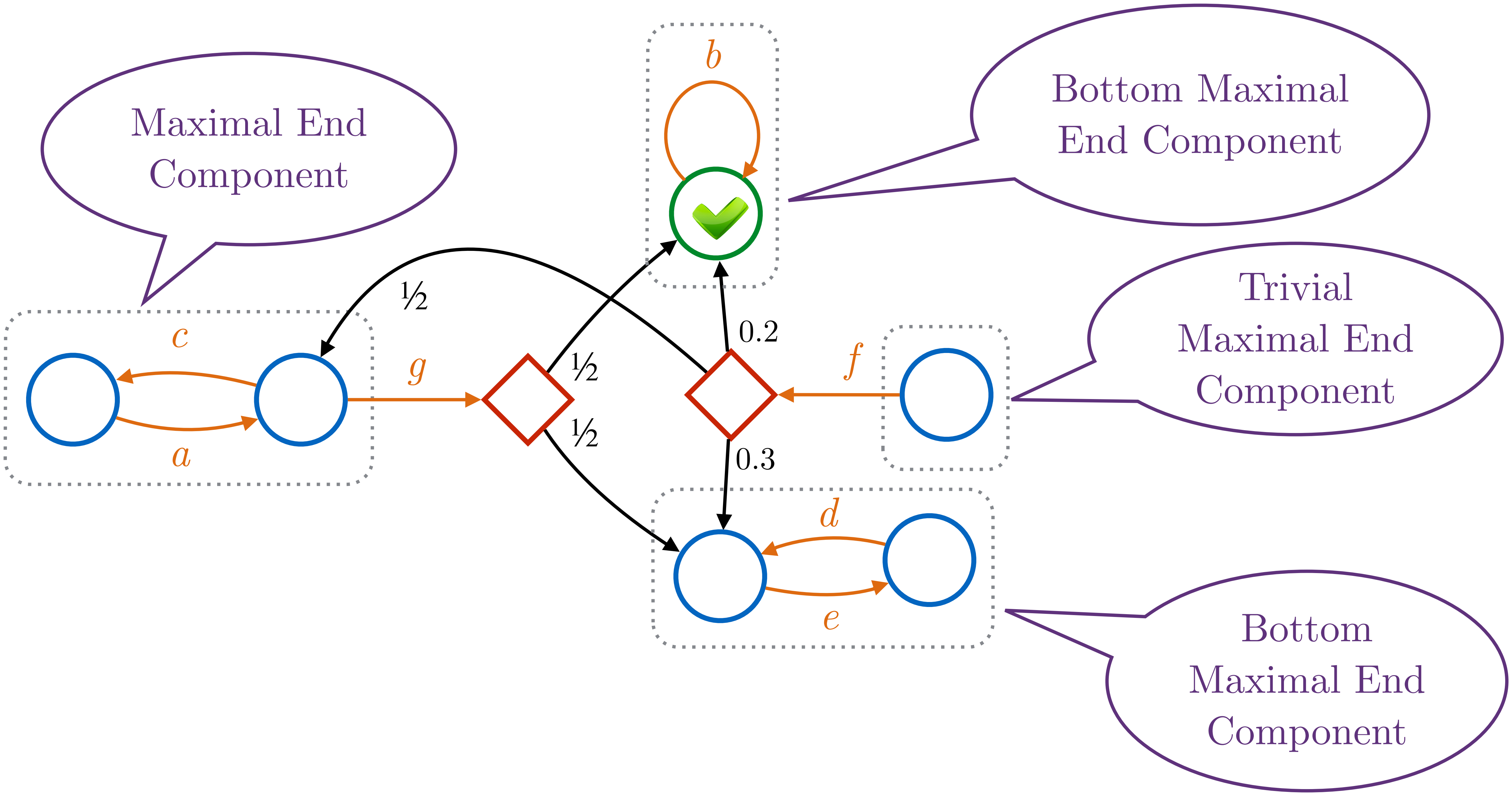


Max-reduced MDP

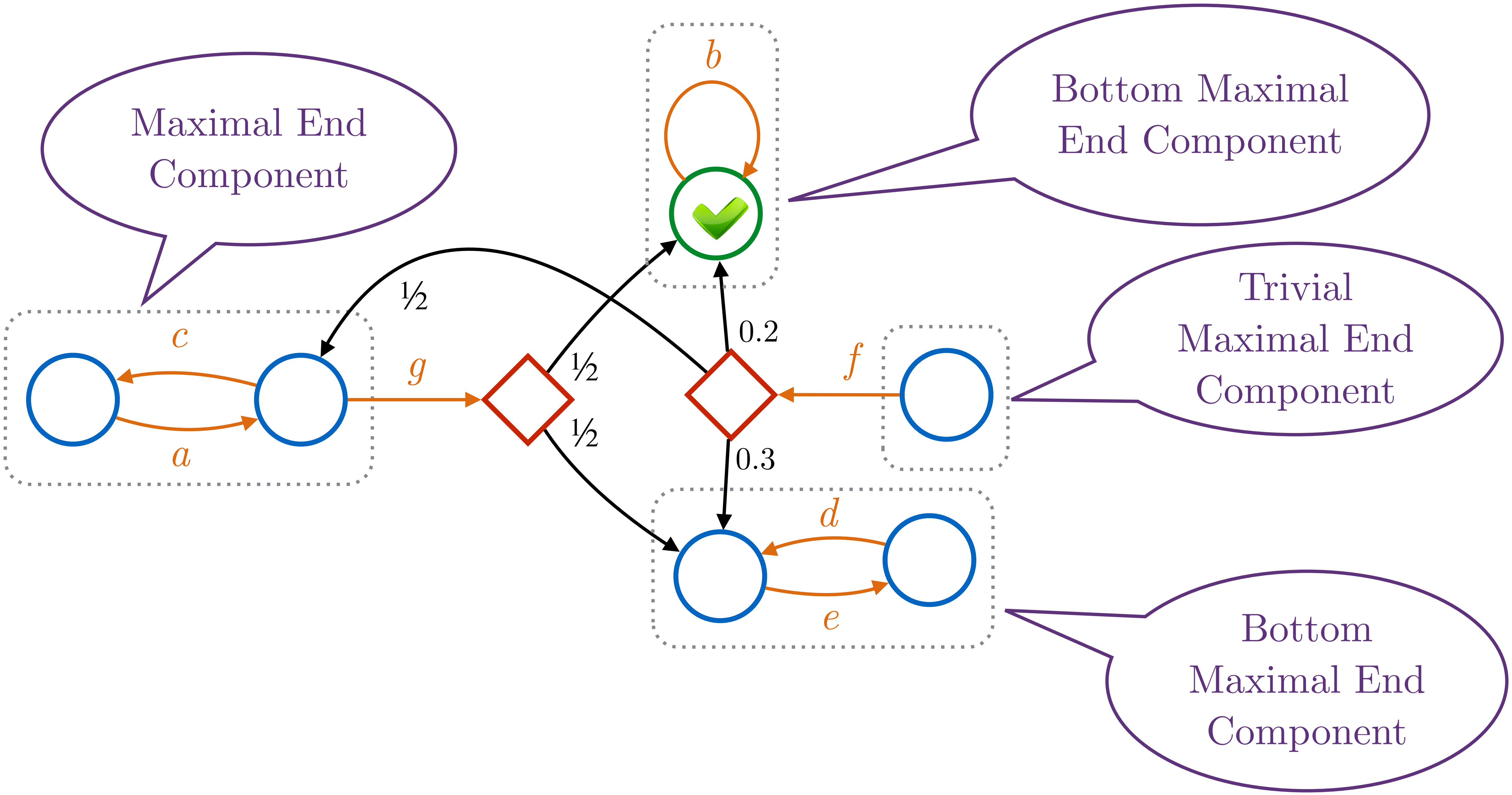
NEW! Use Maximal End Components… (computable in polynomial time)
and trivialize them! Now, unicity of the fixed point

[de Alfaro, 1997]

An even smaller MDP for minimal probabilities

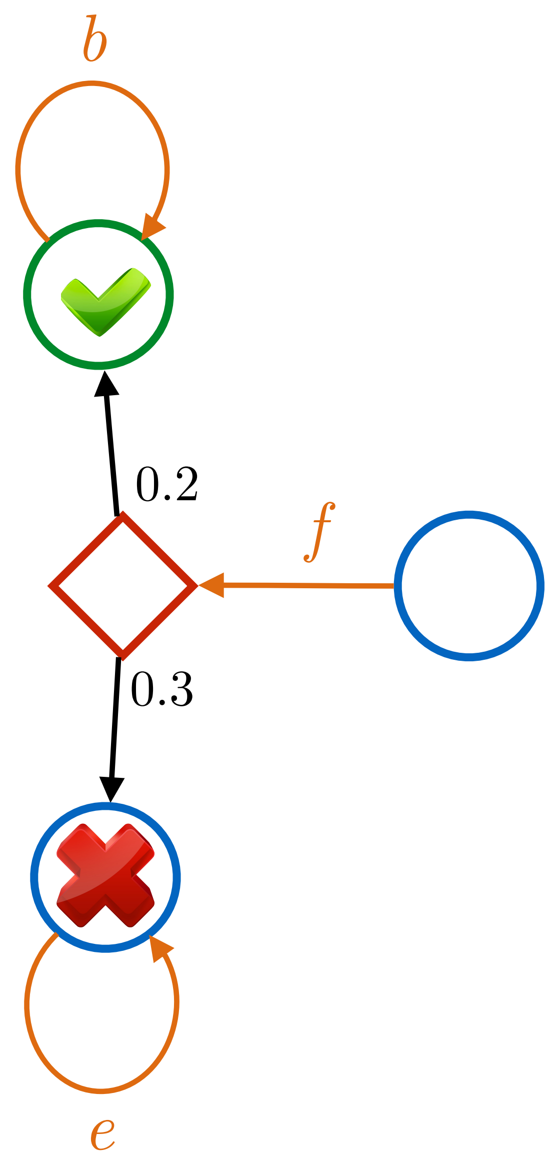


An even smaller MDP for minimal probabilities



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 $Pr^{\min}_{\mathcal{M}}(\mathsf{F}\checkmark)$

$x_{\checkmark} := 1$; $x_{\times} := 0$; $y_{\checkmark} := 1$; $y_{\times} := 0$

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 $(x_s)_{s \in S}, (y_s)_{s \in S}$

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

Interval iteration algorithm in reduced MDPs

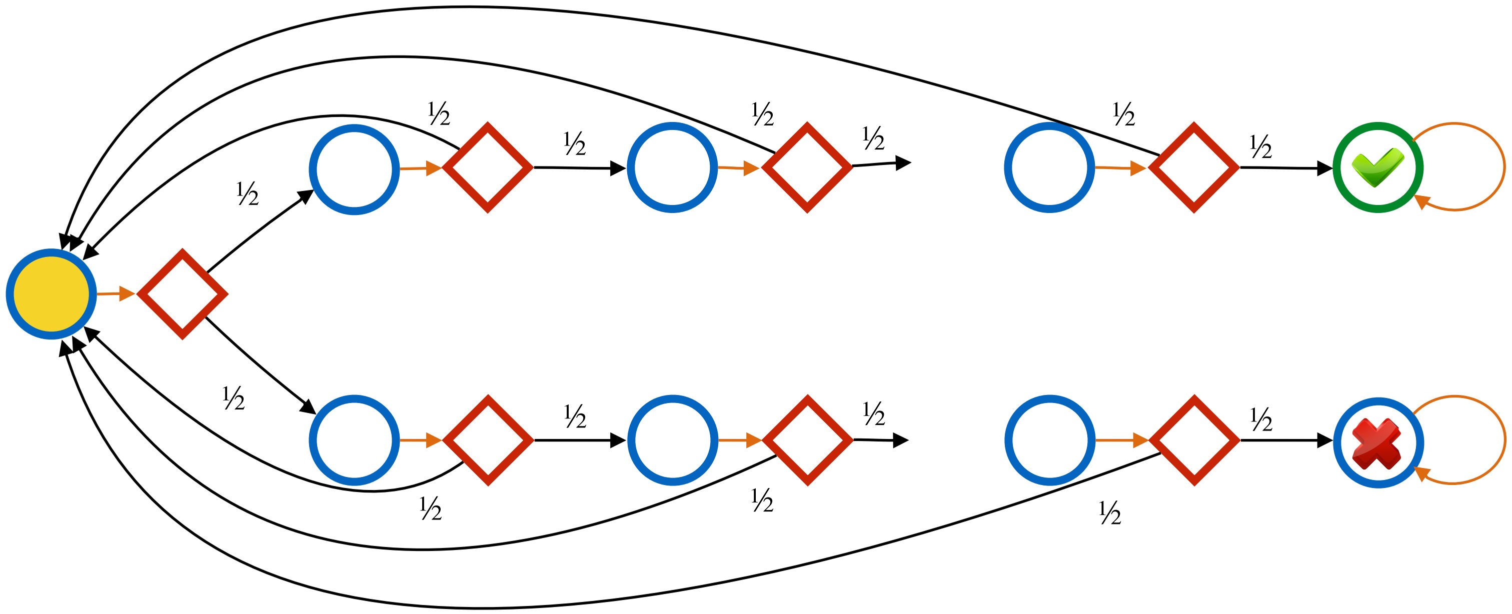
Input: Min-reduced MDP $\mathcal{M} = (S, \alpha_{\mathcal{M}}, \delta_{\mathcal{M}})$, convergence threshold ε
Output: Under- and over-approximation of $Pr^{\min}_{\mathcal{M}}(\mathsf{F}\,✓)$

```
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 $\mathrm{Pr}_s^{\max}(\mathsf{F}\,✓)$.

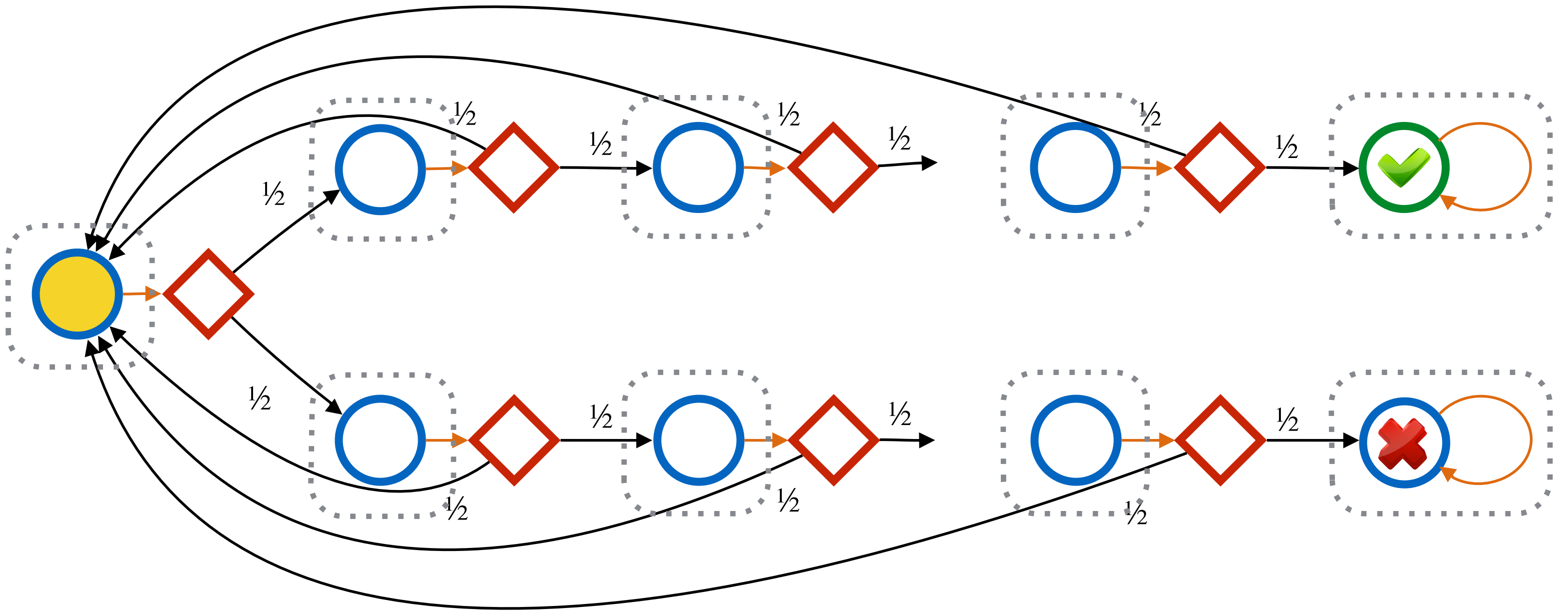
Possible speed-up: only check size of interval for a given state...

Rate of convergence



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

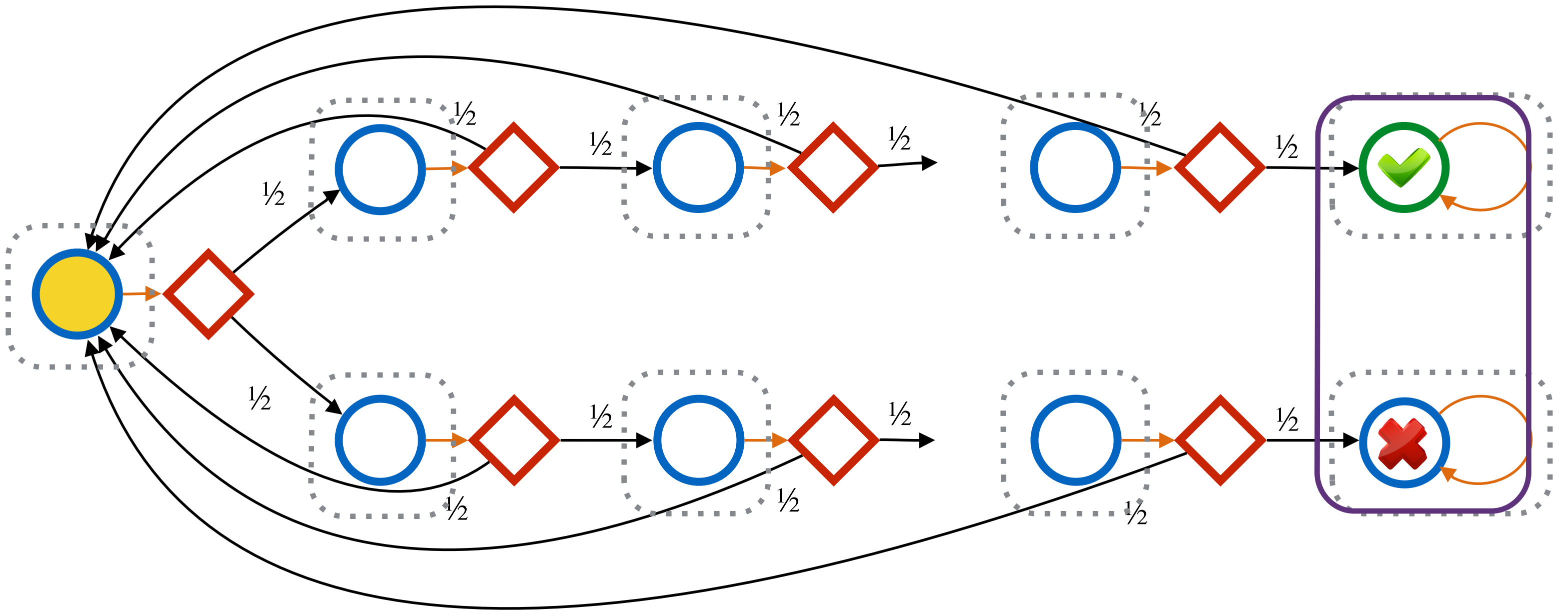
Rate of convergence



2 BMECs and only trivial MECs

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

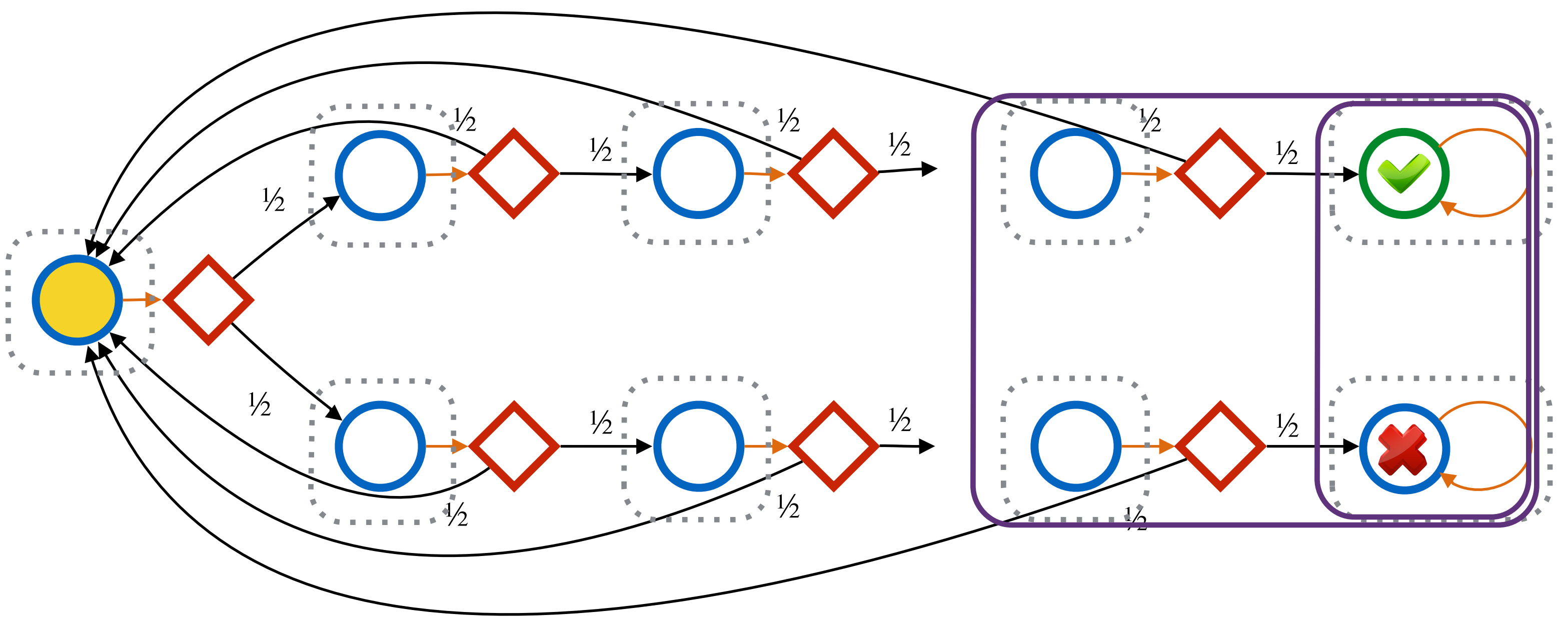
Rate of convergence



2 BMECs and only trivial MECs
attractor decomposition: length I

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

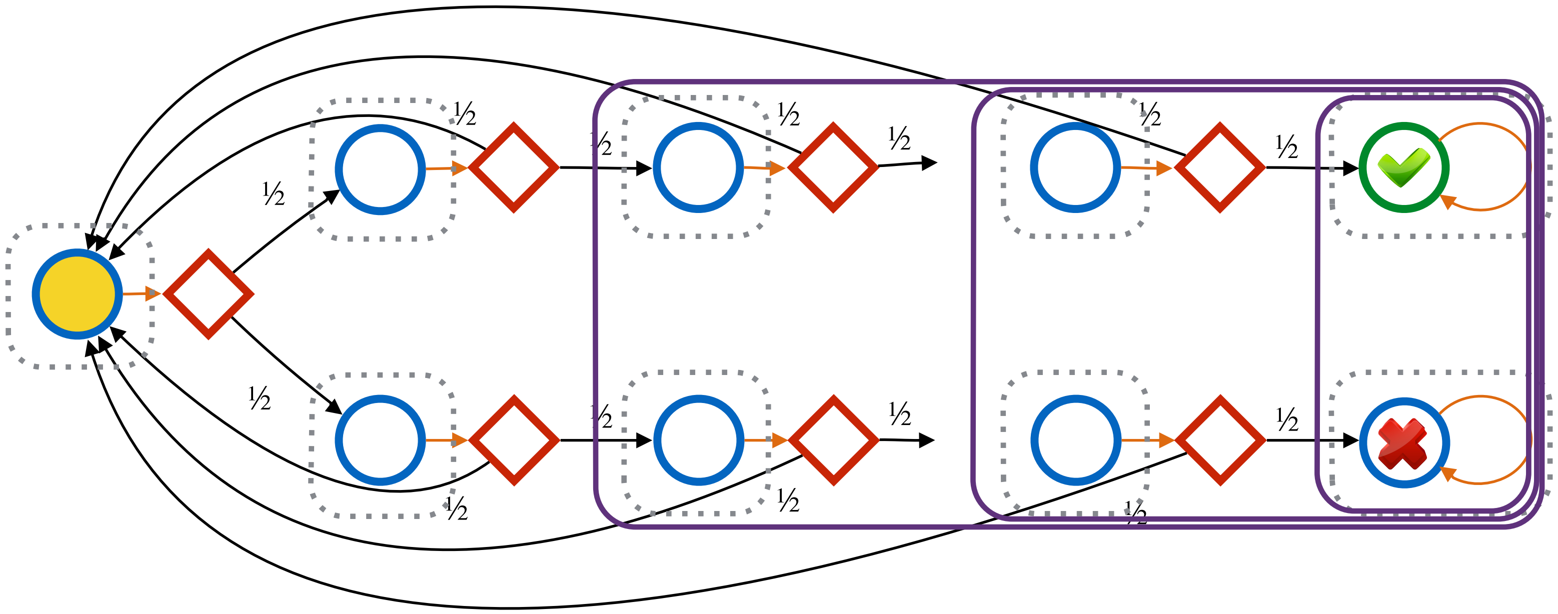
Rate of convergence



2 BMECs and only trivial MECs
attractor decomposition: length I

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

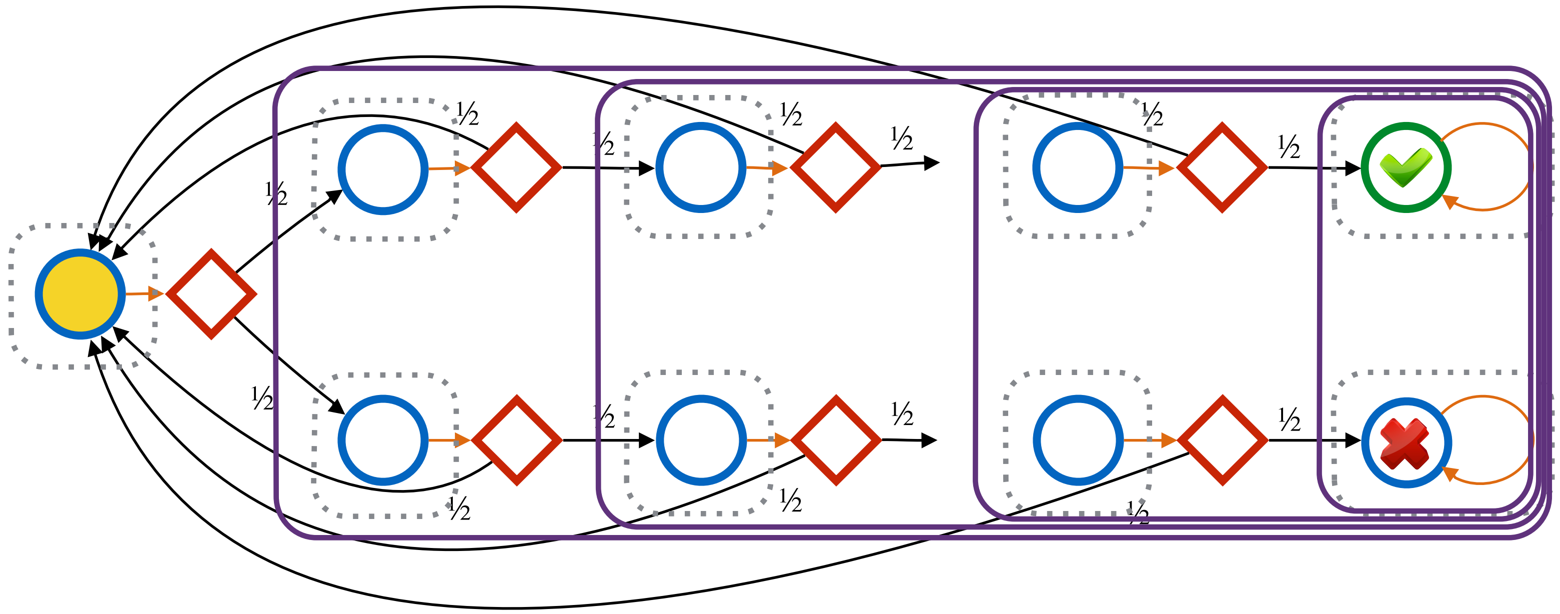
Rate of convergence



2 BMECs and only trivial MECs
attractor decomposition: length I

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

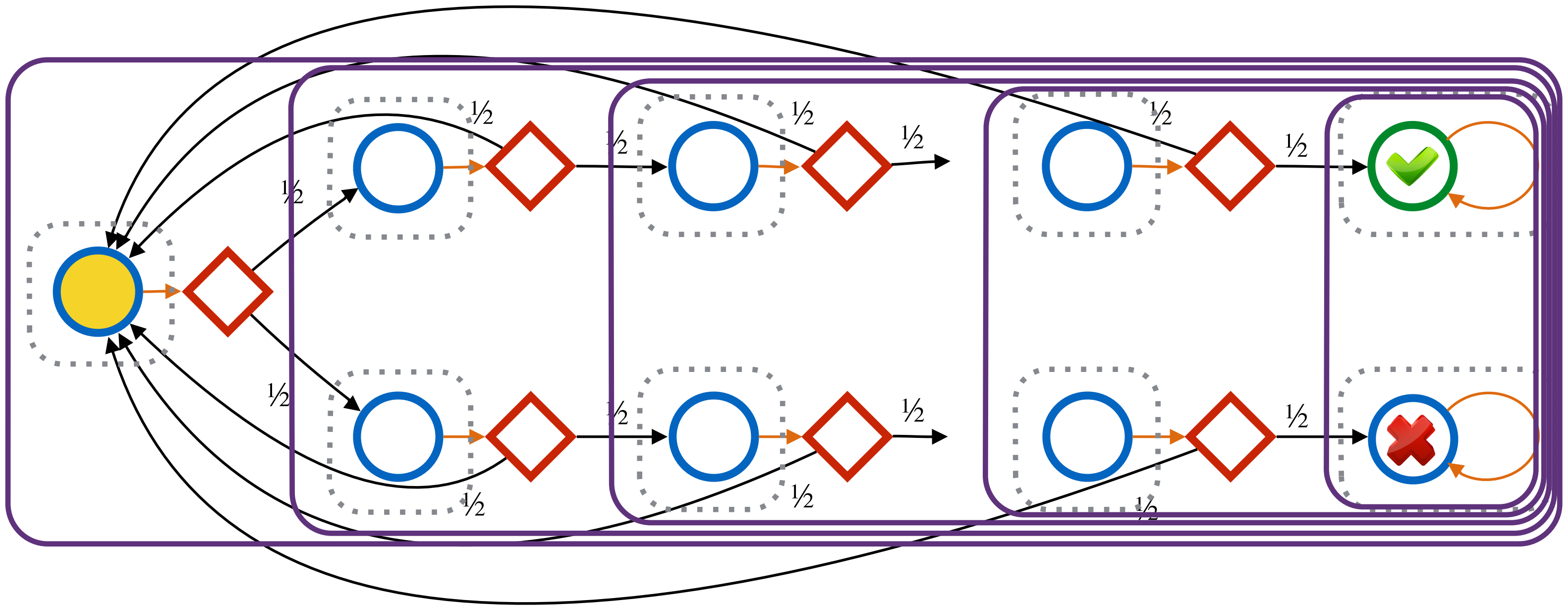
Rate of convergence



2 BMECs and only trivial MECs
attractor decomposition: length I

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

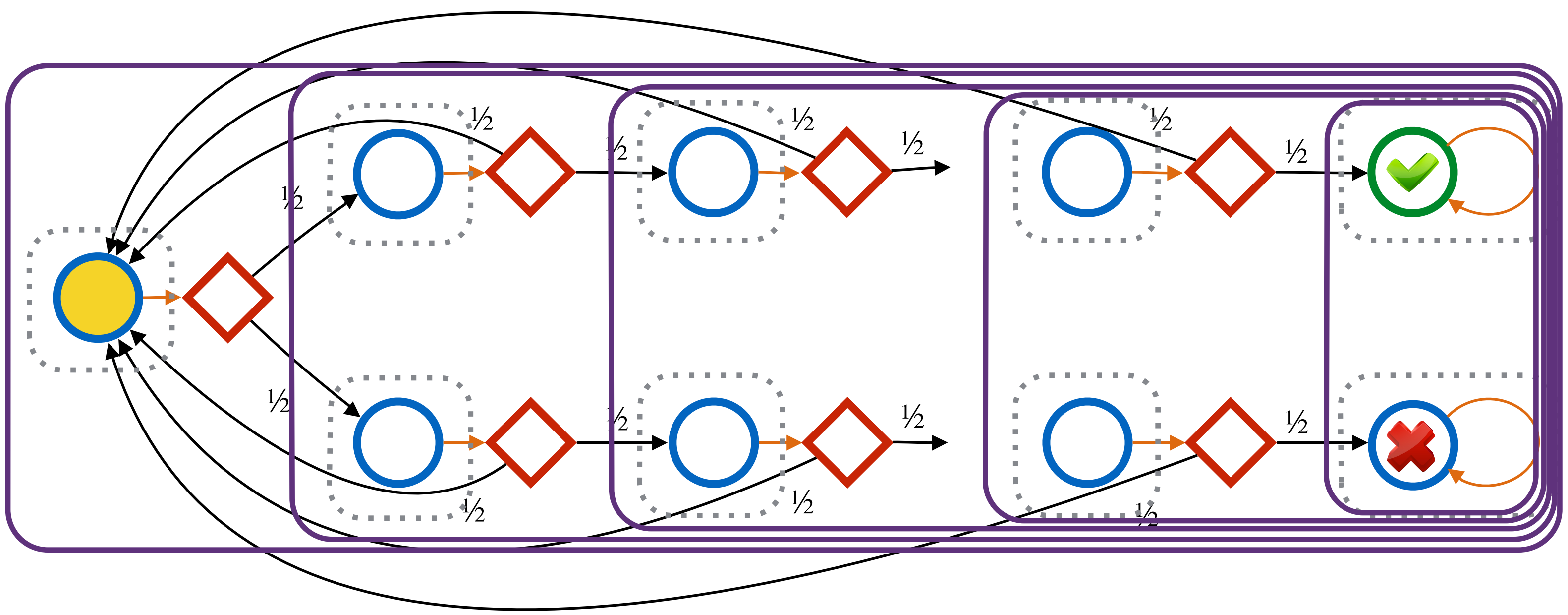
Rate of convergence



2 BMECs and only trivial MECs
attractor decomposition: length I

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

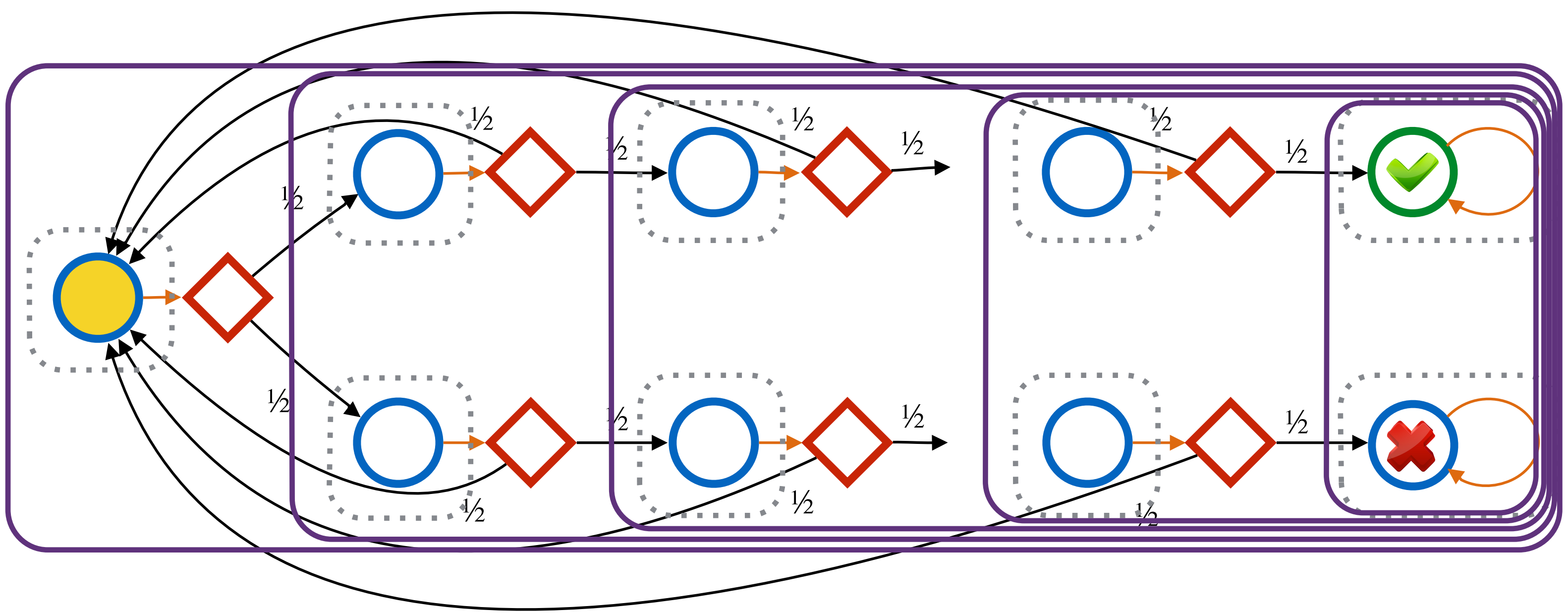
Rate of convergence



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: $x_s = \Pr_s^{\min}(\mathsf{F}^{\leq n} ✔)$ $y_s = \Pr_s^{\min}(\mathsf{G}^{\leq n}(\neg ✖))$

Rate of convergence

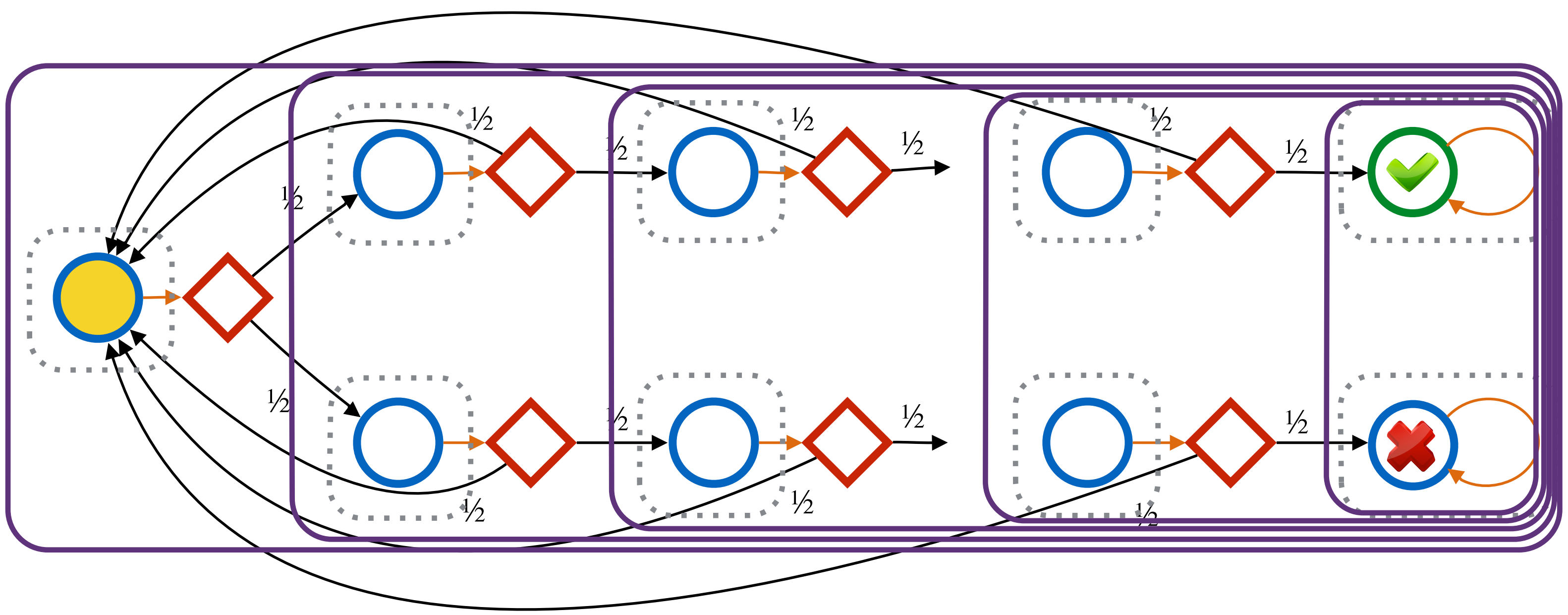


2 BMECs and only trivial MECs
attractor decomposition: length I
smallest positive probability: η

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



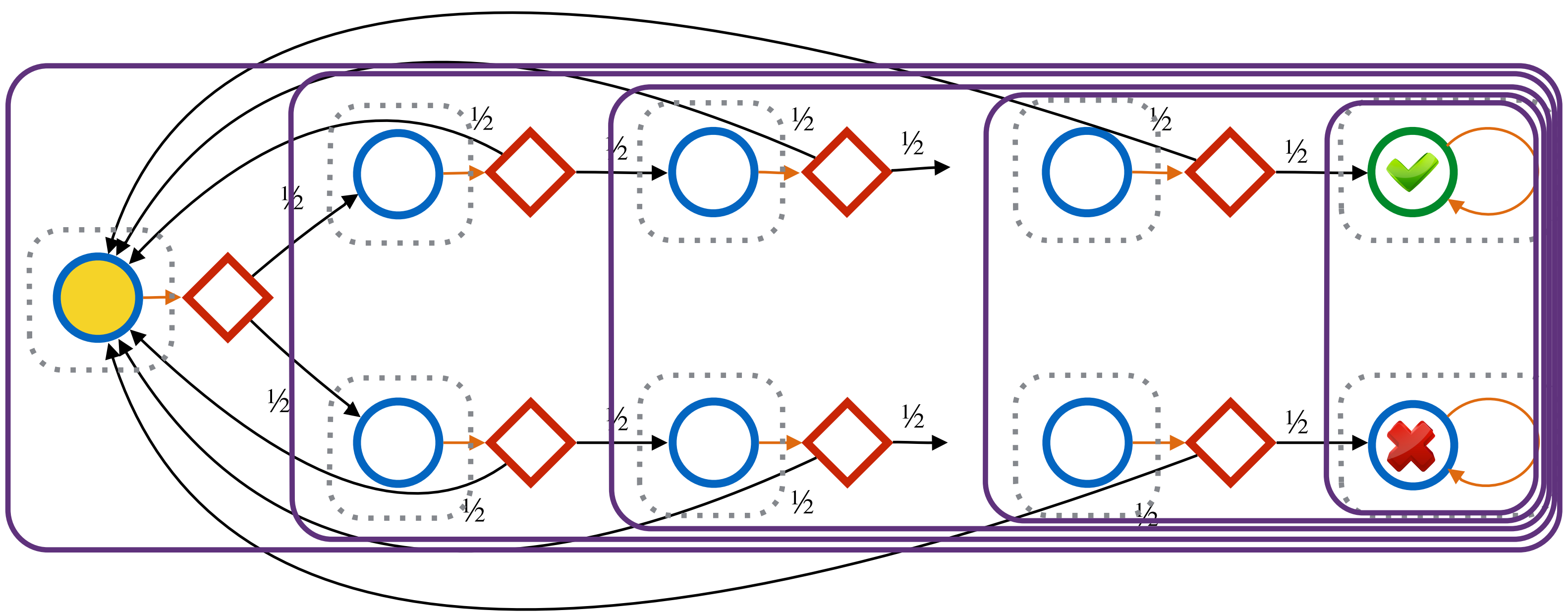
2 BMECs and only trivial MECs
attractor decomposition: length I
smallest positive probability: η

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

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

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

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: $x_s = \mathrm{Pr}_s^{\min}(\mathsf{F}^{\leq n} \checkmark)$ $\quad y_s = \mathrm{Pr}_s^{\min}(\mathsf{G}^{\leq n}(\neg \times))$

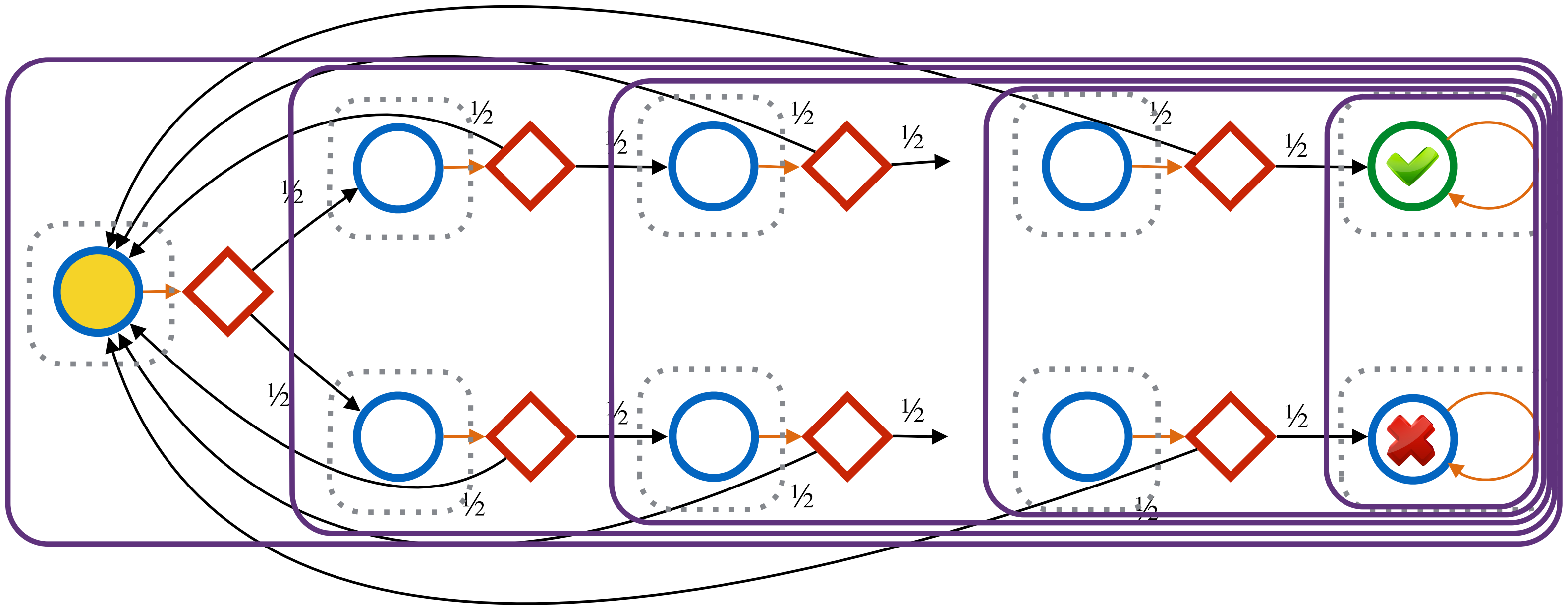
Leaking property: $\forall n \in \mathbb{N} \quad \mathrm{Pr}_s^{\max}(\mathsf{G}^{\leq nI} \neg(\checkmark \vee \times)) \leq (1-\eta^I)^n$

$$y_s^{(nI)} - x_s^{(nI)} = \mathrm{Pr}_s^{\sigma}(\mathsf{G}^{\leq nI}(\neg \times)) - \mathrm{Pr}_s^{\sigma'}(\mathsf{F}^{\leq nI} \checkmark) \leq \mathrm{Pr}_s^{\sigma'}(\mathsf{G}^{\leq nI}(\neg \times)) - \mathrm{Pr}_s^{\sigma'}(\mathsf{F}^{\leq nI} \checkmark)$$

$$= \mathrm{Pr}_s^{\sigma'}(\mathsf{G}^{\leq nI} \neg(\checkmark \vee \times)) \leq (1-\eta^I)^n$$

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: $x_s = \Pr_s^{\min}(\mathsf{F}^{\leq n} ✔)$ $\quad y_s = \Pr_s^{\min}(\mathsf{G}^{\leq n}(\neg ✘))$

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

The interval iteration algorithm converges in at most $I \left\lceil \dfrac{\log \varepsilon}{\log(1-\eta^I)} \right\rceil$ 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 $4N^3 \left\lceil (1/\eta)^N \log_2 d \right\rceil$ 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 $4N^3 \left\lceil (1/\eta)^N \log_2 d \right\rceil$ 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 $4N^3 \left\lceil (1/\eta)^N \log_2 d \right\rceil$ 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: $\alpha = \mathcal{O}(N^N d^{3N^2})$

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

∼

- **[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
- *To be published at ATVA 2014* **[Brázdil, Chatterjee, Chmelík, Forejt, Křetínský, Kwiatkowska, Parker, Ujma, 2014]** same techniques in a machine learning framework with almost sure convergence and computation of non-trivial end components on-the-fly