Exact Quantitative Probabilistic Model Checking Through Rational Search

Matthew S. Bauer[1] Umang Mathur[1] Rohit Chadha[2]
A. Prasad Sistla[3] Mahesh Viswanathan[1]

[1] University of Illinois, Urbana Champaign

[2] University of Missouri

[3] University of Illinois, Chicago

Probabilistic Systems Everywhere

Systems exhibiting stochastic behavior :

- Modeling unreliable/unpredictable behavior - processor failure, message loss, etc.,
- Model-based performance evaluation - analyze average waiting time, delay queue length, etc.,
- Cryptographic protocols and encryption schemes privacy and security
- Distributed and network protocols - break symmetry

Modeling Probabilistic Systems

State-transition systems with probabilistic transitions, remain the popular choice for modeling probabilistic systems.

Modeling Probabilistic Systems

State-transition systems with probabilistic transitions, remain the popular choice for modeling probabilistic systems.

Discrete Time Markov Chains

A DTMC is a tuple $\mathcal{M} = (Z, \Delta, L)$, where

- Z is a finite set of states
- $\Delta : Z \to \text{Dist}(Z)$ is the probabilistic transition function of $\mathcal{M}$
- $L : Z \to 2^{\text{AP}}$ is a labeling function

Modeling Probabilistic Systems

State-transition systems with probabilistic transitions, remain the popular choice for modeling probabilistic systems.

Discrete Time Markov Chains

A DTMC is a tuple $\mathcal{M} = (Z, \Delta, L)$, where

- Z is a finite set of states
- $\Delta : Z \to \text{Dist}(Z)$ is the probabilistic transition function of $\mathcal{M}$
- $L : Z \to 2^{\text{AP}}$ is a labeling function

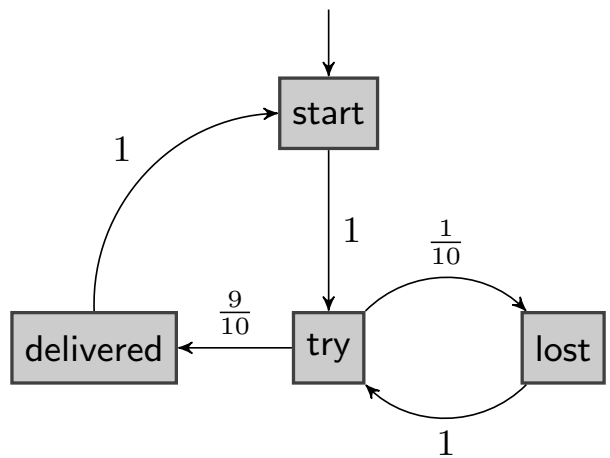


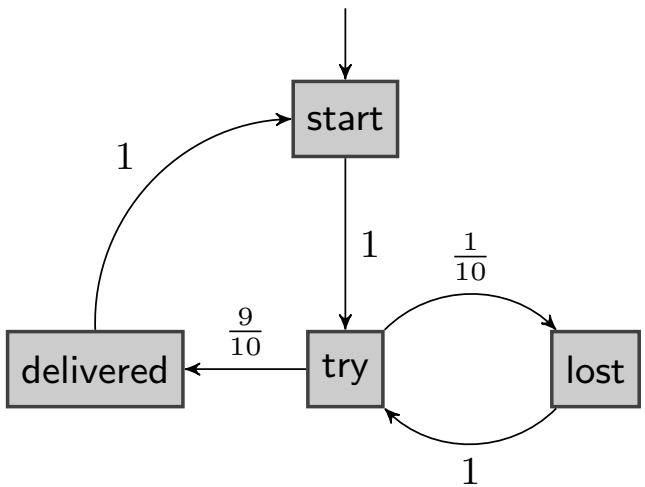
Figure: DTMC modeling a simple communication protocol

Computing Reachability Probabilities

Verification of probabilistic systems often involves checking the probability of reaching some set of states.

Computing Reachability Probabilities

Verification of probabilistic systems often involves checking the probability of reaching some set of states.

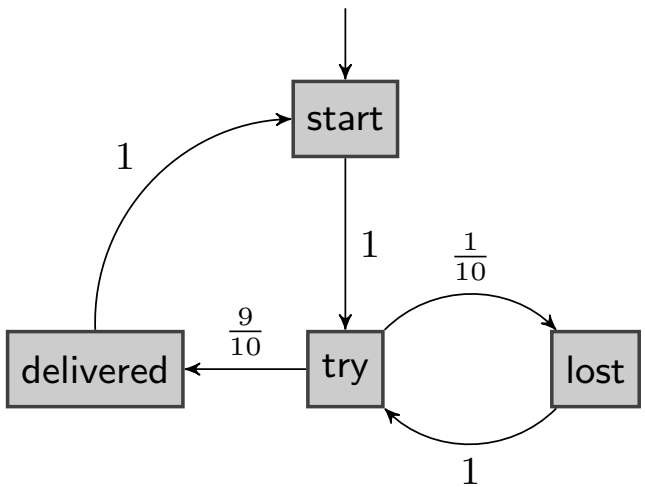


Probability of reaching the state 'delivered' starting from each state ?

Computing Reachability Probabilities

Verification of probabilistic systems often involves checking the probability of reaching some set of states.

x_z : probability of reaching 'delivered' starting from z

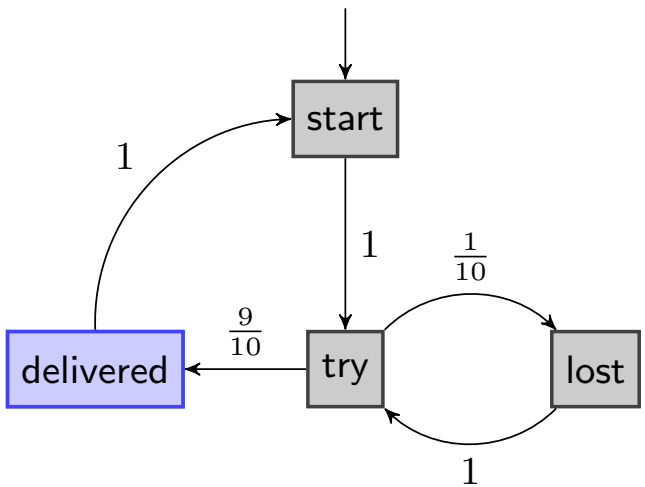


Probability of reaching the state 'delivered' starting from each state ?

Computing Reachability Probabilities

Verification of probabilistic systems often involves checking the probability of reaching some set of states.

x_z : probability of reaching 'delivered' starting from z

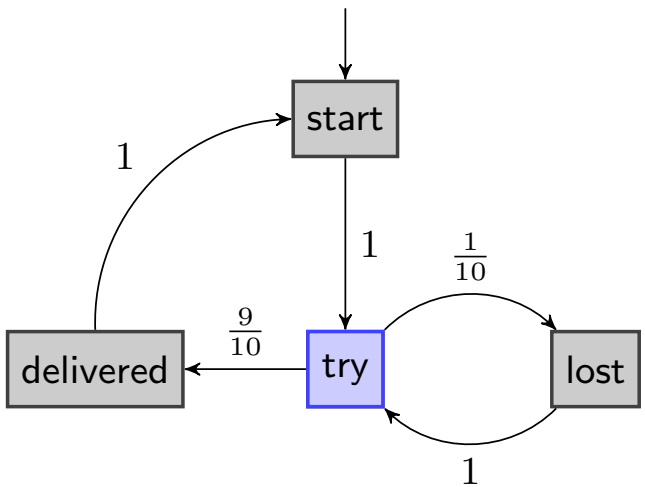


$x_{\text{delivered}} = 1.0$

Probability of reaching the state 'delivered' starting from each state ?

Computing Reachability Probabilities

Verification of probabilistic systems often involves checking the probability of reaching some set of states.



x_z : probability of reaching 'delivered' starting from z

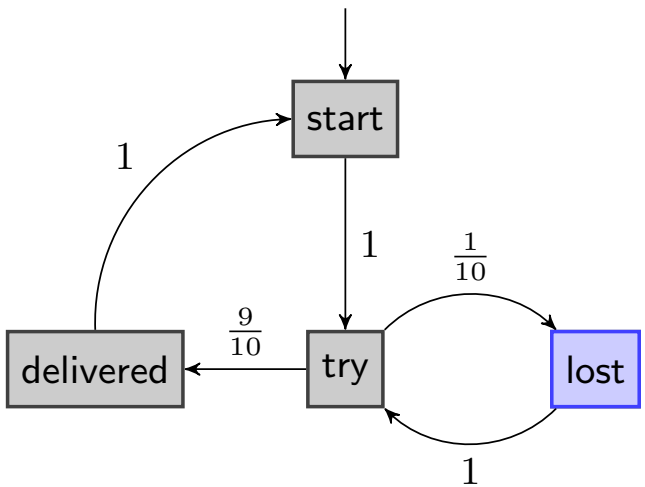
$$x_{\text{try}} = \frac{9}{10} * x_{\text{delivered}} + \frac{1}{10} * x_{\text{lost}}$$

$$x_{\text{delivered}} = 1.0$$

Probability of reaching the state 'delivered' starting from each state ?

Computing Reachability Probabilities

Verification of probabilistic systems often involves checking the probability of reaching some set of states.



x_z : probability of reaching 'delivered' starting from z

$$x_{\text{lost}} = 1.0 * x_{\text{try}}$$

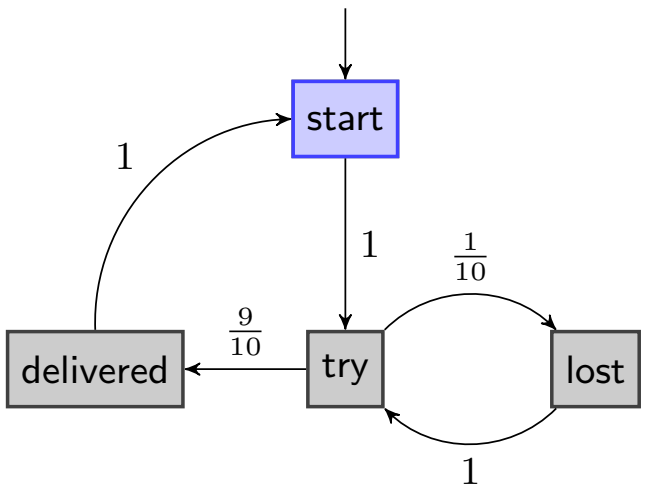
$$x_{\text{try}} = \frac{9}{10} * x_{\text{delivered}} + \frac{1}{10} * x_{\text{lost}}$$

$$x_{\text{delivered}} = 1.0$$

Probability of reaching the state 'delivered' starting from each state ?

Computing Reachability Probabilities

Verification of probabilistic systems often involves checking the probability of reaching some set of states.



x_z : probability of reaching 'delivered' starting from z

$$x_{\text{start}} = 1.0 * x_{\text{try}}$$
$$x_{\text{lost}} = 1.0 * x_{\text{try}}$$
$$x_{\text{try}} = \frac{9}{10} * x_{\text{delivered}} + \frac{1}{10} * x_{\text{lost}}$$
$$x_{\text{delivered}} = 1.0$$

Probability of reaching the state 'delivered' starting from each state ?

Computing Reachability Probabilities : Linear Programming

Computing Reachability Probabilities : Linear Programming

– Computing reachability probabilities in DTMC = linear programming

$$\overline{x} = A\overline{x} + \overline{b}$$

Computing Reachability Probabilities : Linear Programming

– Computing reachability probabilities in DTMC = linear programming

$$\overline{x} = A\overline{x} + \overline{b}$$

– Linear programming

Computing Reachability Probabilities : Linear Programming

- Computing reachability probabilities in DTMC = linear programming

$$\overline{x} = A\overline{x} + \overline{b}$$

- Linear programming
  - doesn't scale well to large models (despite low asymptotic complexity)

Computing Reachability Probabilities : Linear Programming

- Computing reachability probabilities in DTMC = linear programming

$$\overline{x} = A\overline{x} + \overline{b}$$

- Linear programming
  - doesn't scale well to large models (despite low asymptotic complexity)
  - rarely used in practice for quantitative model checking

Computing Reachability Probabilities : Linear Programming

- Computing reachability probabilities in DTMC = linear programming

$$\overline{x} = A\overline{x} + \overline{b}$$

- Linear programming
  - doesn't scale well to large models (despite low asymptotic complexity)
  - rarely used in practice for quantitative model checking
- State-of-the-art model checkers, such as PRISM, resort to techniques that compute approximations

Approximate Model Checking : Value Iteration

– $\overline{x} = F(\overline{x})$,

Approximate Model Checking : Value Iteration

– $\overline{x} = F(\overline{x})$, where $F(\overline{x}) = A\overline{x} + \overline{b}$

Approximate Model Checking : Value Iteration

- $\overline{x} = F(\overline{x})$, where $F(\overline{x}) = A\overline{x} + \overline{b}$
  - Fixpoint equation
  - Unique solution
  - F is monotone

Approximate Model Checking : Value Iteration

- $\overline{x} = F(\overline{x})$, where $F(\overline{x}) = A\overline{x} + \overline{b}$
  - Fixpoint equation
  - Unique solution
  - F is monotone
- The following iterative sequence converges to the unique fixpoint :

$$\overline{x}^{(i+1)} = A\overline{x}^{(i)} + \overline{b}$$

starting from $\overline{x}^{(0)} = (0, 0, \ldots, 0)^T$

Approximate Model Checking : Value Iteration

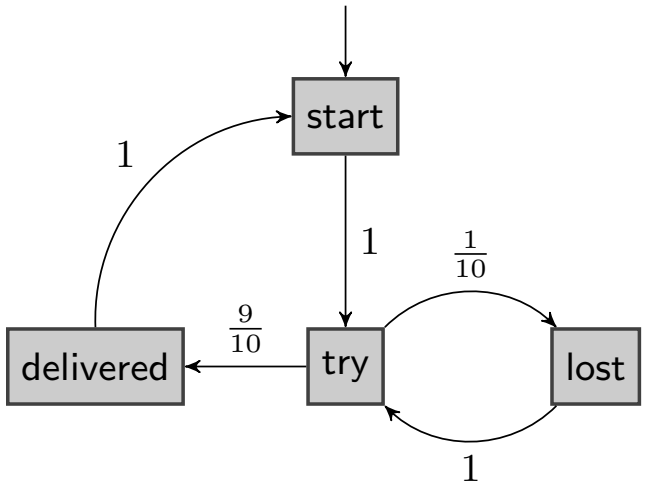
- $\overline{x} = F(\overline{x})$, where $F(\overline{x}) = A\overline{x} + \overline{b}$
  - Fixpoint equation
  - Unique solution
  - F is monotone
- The following iterative sequence converges to the unique fixpoint :

$$\overline{x}^{(i+1)} = A\overline{x}^{(i)} + \overline{b}$$

  starting from $\overline{x}^{(0)} = (0, 0, \ldots, 0)^T$

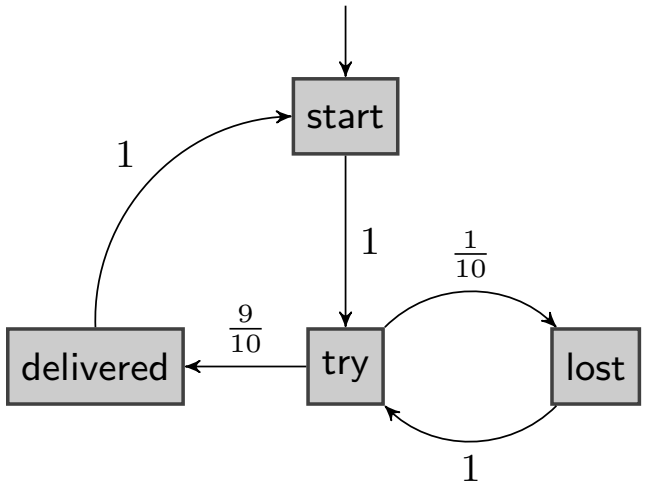
- This technique is called value iteration

Approximate Model Checking : Value Iteration



Probability of reaching 'delivered' from each state?

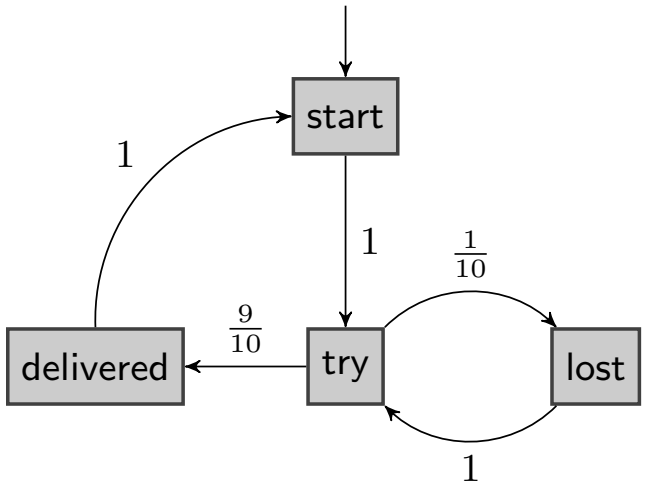
Approximate Model Checking : Value Iteration



$$x_{\text{start}}^{(i+1)} = 1.0 * x_{\text{try}}^{(i)}$$
$$x_{\text{try}}^{(i+1)} = \frac{9}{10} * x_{\text{delivered}}^{(i)} + \frac{1}{10} * x_{\text{lost}}^{(i)}$$
$$x_{\text{lost}}^{(i+1)} = 1.0 * x_{\text{try}}^{(i)}$$
$$x_{\text{delivered}}^{(i+1)} = 1.0$$

Probability of reaching 'delivered' from each state?

Approximate Model Checking : Value Iteration

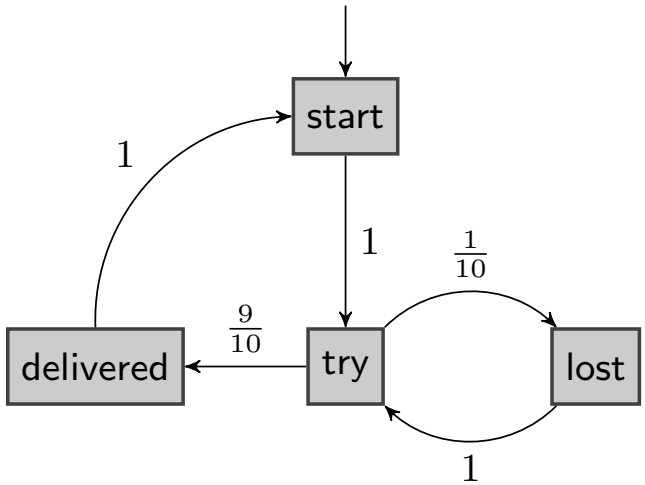


$$x_{\text{start}}^{(i+1)} = 1.0 * x_{\text{try}}^{(i)}$$
$$x_{\text{try}}^{(i+1)} = \frac{9}{10} * x_{\text{delivered}}^{(i)} + \frac{1}{10} * x_{\text{lost}}^{(i)}$$
$$x_{\text{lost}}^{(i+1)} = 1.0 * x_{\text{try}}^{(i)}$$
$$x_{\text{delivered}}^{(i+1)} = 1.0$$

Probability of reaching 'delivered' from each state?

Value Iteration :

Approximate Model Checking : Value Iteration



$$x_{\text{start}}^{(i+1)} = 1.0 * x_{\text{try}}^{(i)}$$
$$x_{\text{try}}^{(i+1)} = \frac{9}{10} * x_{\text{delivered}}^{(i)} + \frac{1}{10} * x_{\text{lost}}^{(i)}$$
$$x_{\text{lost}}^{(i+1)} = 1.0 * x_{\text{try}}^{(i)}$$
$$x_{\text{delivered}}^{(i+1)} = 1.0$$

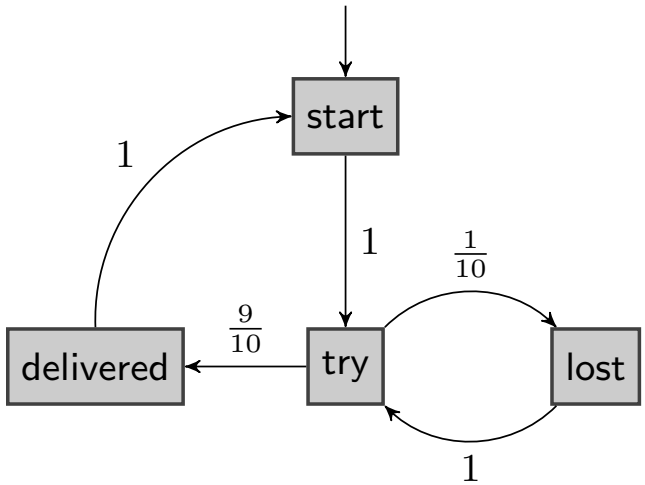
Probability of reaching 'delivered' from each state?

Value Iteration :

x_{start}	0
x_{try}	0
x_{lost}	0
$x_{\text{delivered}}$	0

Init

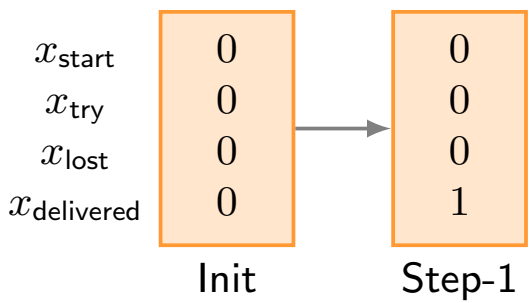
Approximate Model Checking : Value Iteration



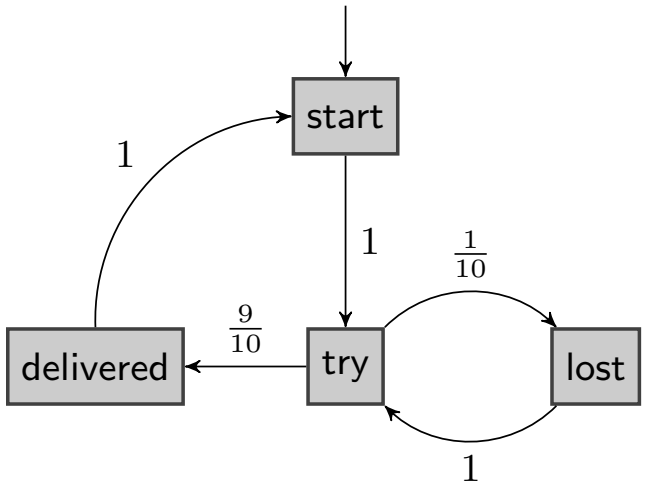
$$x_{\text{start}}^{(i+1)} = 1.0 * x_{\text{try}}^{(i)}$$
$$x_{\text{try}}^{(i+1)} = \frac{9}{10} * x_{\text{delivered}}^{(i)} + \frac{1}{10} * x_{\text{lost}}^{(i)}$$
$$x_{\text{lost}}^{(i+1)} = 1.0 * x_{\text{try}}^{(i)}$$
$$x_{\text{delivered}}^{(i+1)} = 1.0$$

Probability of reaching 'delivered' from each state?

Value Iteration :



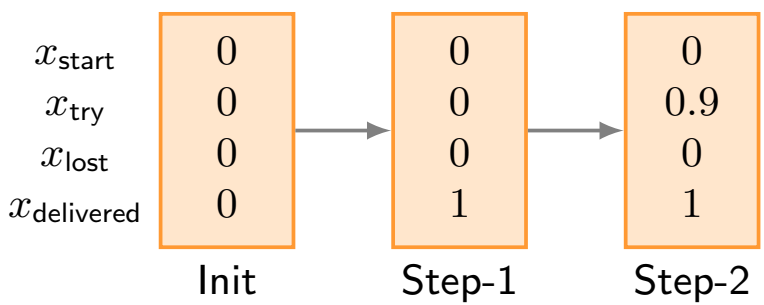
Approximate Model Checking : Value Iteration



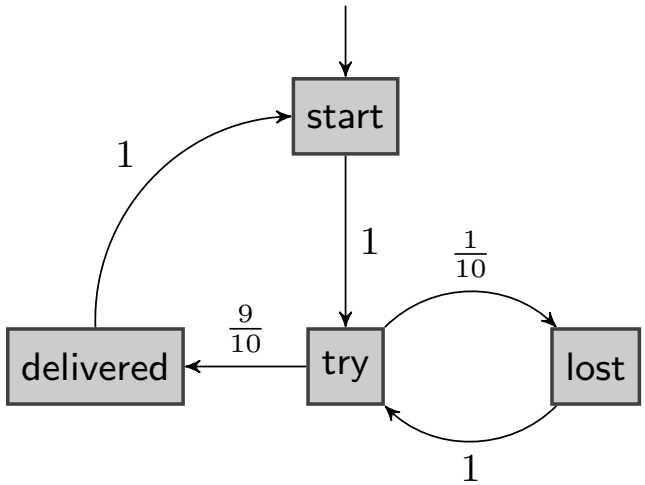
$$x_{\text{start}}^{(i+1)} = 1.0 * x_{\text{try}}^{(i)}$$
$$x_{\text{try}}^{(i+1)} = \frac{9}{10} * x_{\text{delivered}}^{(i)} + \frac{1}{10} * x_{\text{lost}}^{(i)}$$
$$x_{\text{lost}}^{(i+1)} = 1.0 * x_{\text{try}}^{(i)}$$
$$x_{\text{delivered}}^{(i+1)} = 1.0$$

Probability of reaching 'delivered' from each state?

Value Iteration :



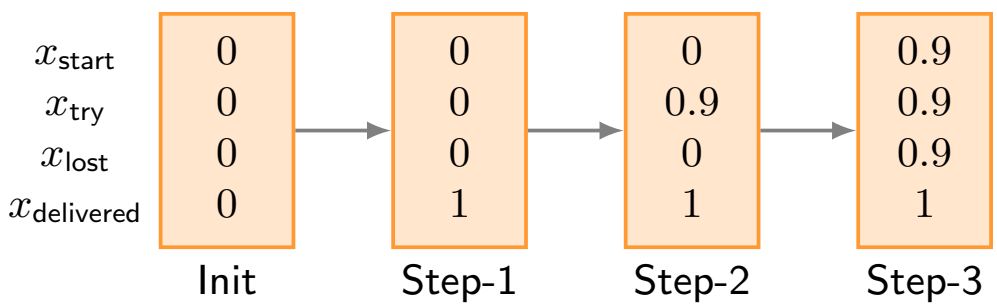
Approximate Model Checking : Value Iteration



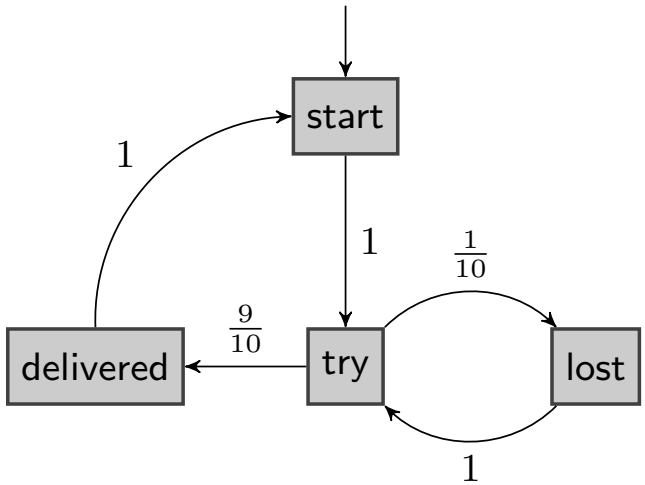
$$x_{\text{start}}^{(i+1)} = 1.0 * x_{\text{try}}^{(i)}$$
$$x_{\text{try}}^{(i+1)} = \frac{9}{10} * x_{\text{delivered}}^{(i)} + \frac{1}{10} * x_{\text{lost}}^{(i)}$$
$$x_{\text{lost}}^{(i+1)} = 1.0 * x_{\text{try}}^{(i)}$$
$$x_{\text{delivered}}^{(i+1)} = 1.0$$

Probability of reaching 'delivered' from each state?

Value Iteration :



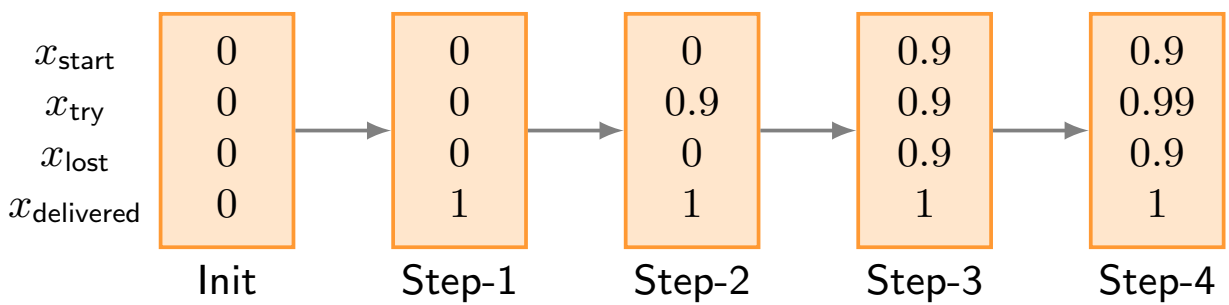
Approximate Model Checking : Value Iteration



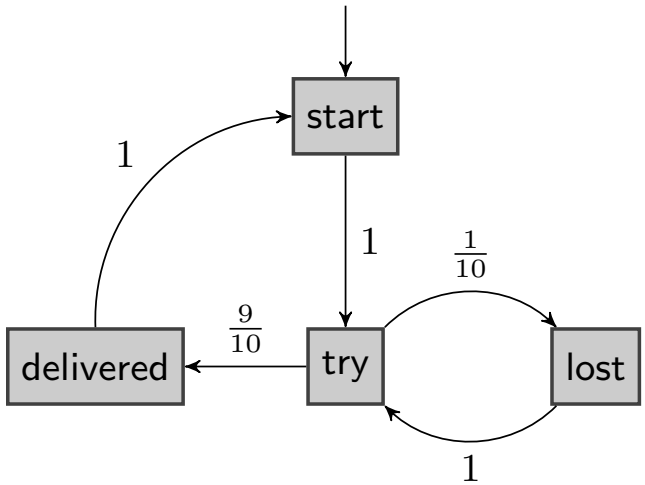
$$x_{\text{start}}^{(i+1)} = 1.0 * x_{\text{try}}^{(i)}$$
$$x_{\text{try}}^{(i+1)} = \frac{9}{10} * x_{\text{delivered}}^{(i)} + \frac{1}{10} * x_{\text{lost}}^{(i)}$$
$$x_{\text{lost}}^{(i+1)} = 1.0 * x_{\text{try}}^{(i)}$$
$$x_{\text{delivered}}^{(i+1)} = 1.0$$

Probability of reaching 'delivered' from each state?

Value Iteration :



Approximate Model Checking : Value Iteration



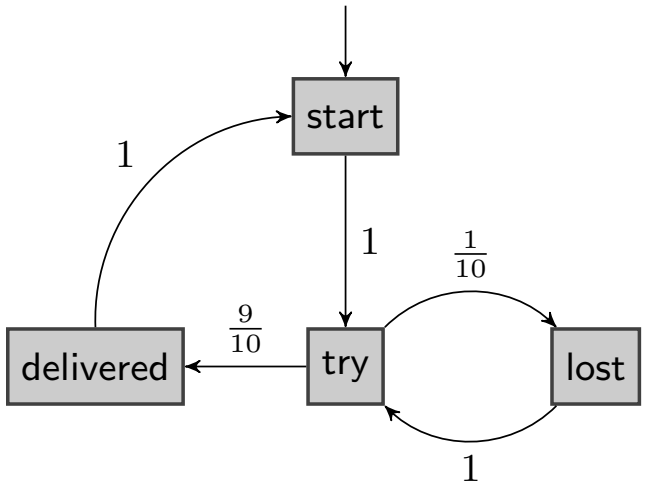
$$x_{\text{start}}^{(i+1)} = 1.0 * x_{\text{try}}^{(i)}$$
$$x_{\text{try}}^{(i+1)} = \frac{9}{10} * x_{\text{delivered}}^{(i)} + \frac{1}{10} * x_{\text{lost}}^{(i)}$$
$$x_{\text{lost}}^{(i+1)} = 1.0 * x_{\text{try}}^{(i)}$$
$$x_{\text{delivered}}^{(i+1)} = 1.0$$

Probability of reaching 'delivered' from each state?

Value Iteration :

	Init	Step-1	Step-2	Step-3	Step-4	Step-5
x_{start}	0	0	0	0.9	0.9	0.99
x_{try}	0	0	0.9	0.9	0.99	0.99
x_{lost}	0	0	0	0.9	0.9	0.99
$x_{\text{delivered}}$	0	1	1	1	1	1

Approximate Model Checking : Value Iteration



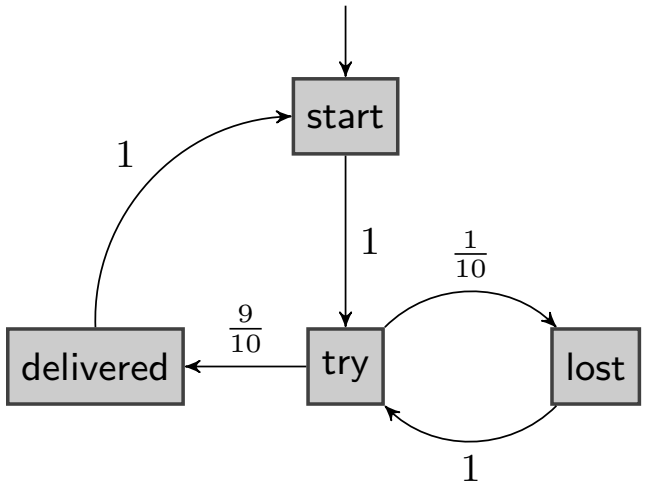
$$x_{\text{start}}^{(i+1)} = 1.0 * x_{\text{try}}^{(i)}$$
$$x_{\text{try}}^{(i+1)} = \frac{9}{10} * x_{\text{delivered}}^{(i)} + \frac{1}{10} * x_{\text{lost}}^{(i)}$$
$$x_{\text{lost}}^{(i+1)} = 1.0 * x_{\text{try}}^{(i)}$$
$$x_{\text{delivered}}^{(i+1)} = 1.0$$

Probability of reaching 'delivered' from each state?

Value Iteration :

	Init	Step-1	Step-2	Step-3	Step-4	Step-5	Step-6
x_{start}	0	0	0	0.9	0.9	0.99	0.99
x_{try}	0	0	0.9	0.9	0.99	0.99	0.999
x_{lost}	0	0	0	0.9	0.9	0.99	0.99
$x_{\text{delivered}}$	0	1	1	1	1	1	1

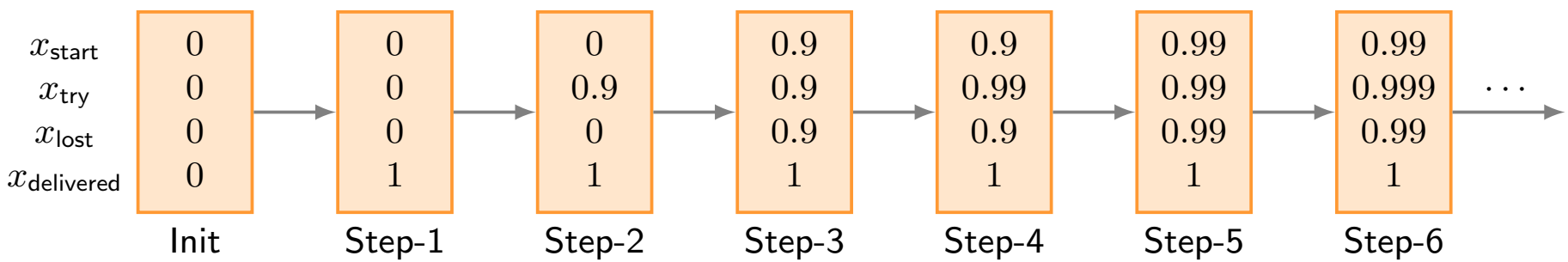
Approximate Model Checking : Value Iteration



$$x_{\text{start}}^{(i+1)} = 1.0 * x_{\text{try}}^{(i)}$$
$$x_{\text{try}}^{(i+1)} = \frac{9}{10} * x_{\text{delivered}}^{(i)} + \frac{1}{10} * x_{\text{lost}}^{(i)}$$
$$x_{\text{lost}}^{(i+1)} = 1.0 * x_{\text{try}}^{(i)}$$
$$x_{\text{delivered}}^{(i+1)} = 1.0$$

Probability of reaching 'delivered' from each state?

Value Iteration :



Value Iteration

More general setting :

Value Iteration

More general setting :

- Probabilistic Computation Tree Logic (PCTL) :
  - Probabilistic analogue of CTL
  - Probabilistic quantifier : $\mathbb{P}_{\bowtie p}(\cdot)$
  - Modal operators : $\mathcal{X}$ (next), $\mathcal{U}$ (until).

- Is the probability of 'delivered' without being 'lost' ≥ 0.575 ?

$$\mathbb{P}_{\geq 0.575}[\neg \text{lost}\, \mathcal{U}\, \text{delivered}]$$

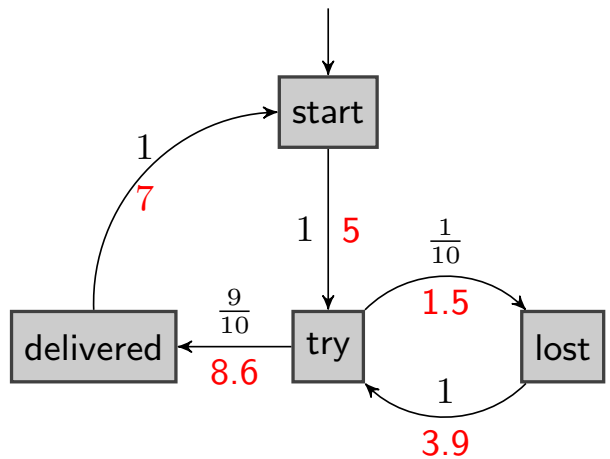
- Is every message almost surely delivered?

$$\mathbb{P}_{=1}[\text{true}\, \mathcal{U}\, \text{delivered}]$$

Value Iteration

More general setting :

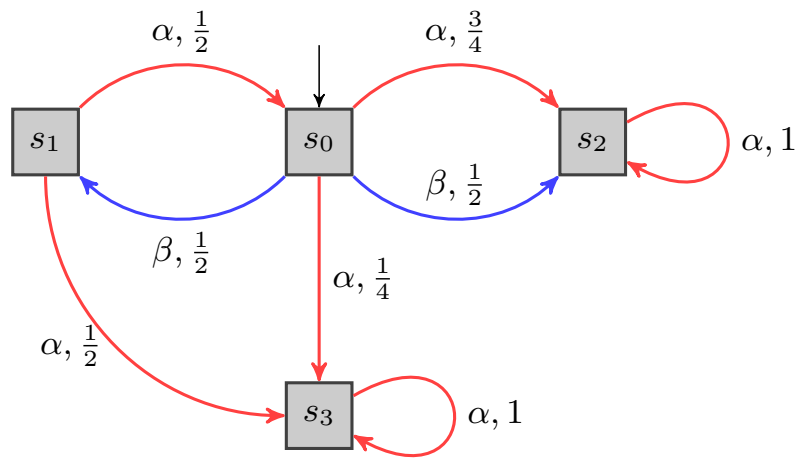
- Probabilistic Computation Tree Logic (PCTL) :
  - Probabilistic analogue of CTL
  - Probabilistic quantifier : $\mathbb{P}_{\bowtie p}(\cdot)$
  - Modal operators : $\mathcal{X}$ (next), $\mathcal{U}$ (until).
- Reward/cost structure
  - Costs associated with transitions
  - Expected cost to reach a state?



Value Iteration

More general setting :

- Probabilistic Computation Tree Logic (PCTL) :
  - Probabilistic analogue of CTL
  - Probabilistic quantifier : $\mathbb{P}_{\bowtie p}(\cdot)$
  - Modal operators : $\mathcal{X}$ (next), $\mathcal{U}$ (until).
- Reward/cost structure
  - Costs associated with transitions
  - Expected cost to reach a state?
- Markov Decision Processes (MDPs)
  - Non-deterministic choice (actions)
  - Probability distribution for every action from a state



Value Iteration : Convergence

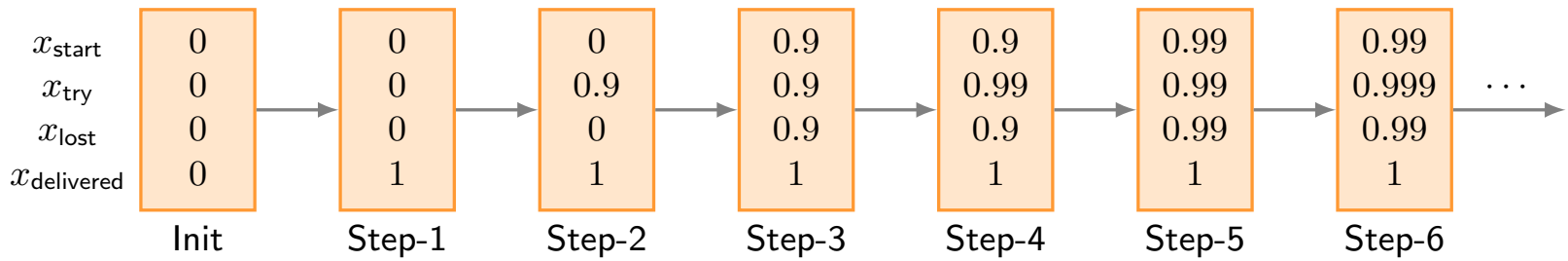
– Value iteration converges to the correct answer, in the limit

Value Iteration : Convergence

- Value iteration converges to the correct answer, in the limit
- The limit may not be reached in any finite number of steps

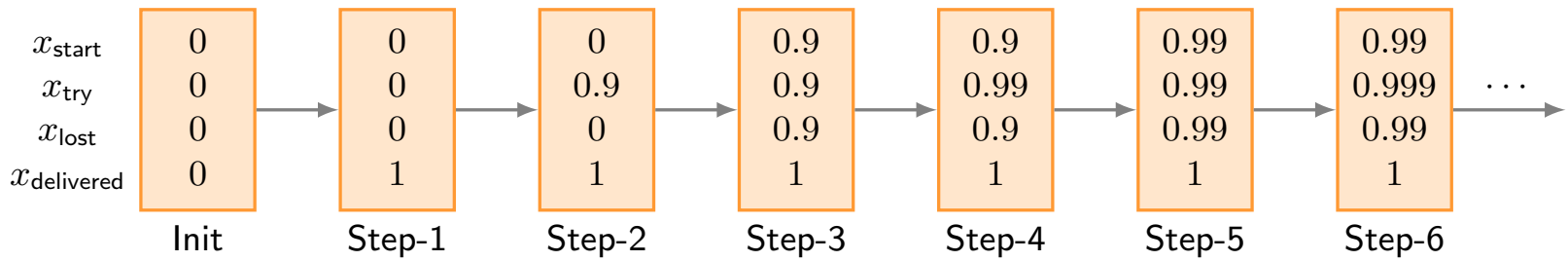
Value Iteration : Convergence

- Value iteration converges to the correct answer, in the limit
- The limit may not be reached in any finite number of steps



Value Iteration : Convergence

- Value iteration converges to the correct answer, in the limit
- The limit may not be reached in any finite number of steps

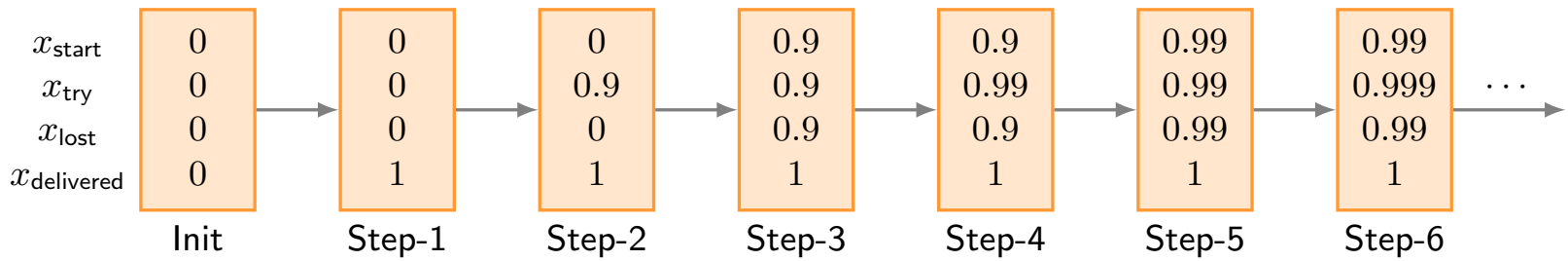


- At Step-$(2i+1)$,

$$\begin{pmatrix} x_{\text{start}} \\ x_{\text{try}} \\ x_{\text{lost}} \\ x_{\text{delivered}} \end{pmatrix} = \begin{pmatrix} 1-10^{-i} \\ 1-10^{-i} \\ 1-10^{-i} \\ 1 \end{pmatrix}$$

Value Iteration : Convergence

- Value iteration converges to the correct answer, in the limit
- The limit may not be reached in any finite number of steps



- At Step-$(2i+1)$,

$$\begin{pmatrix} x_{\text{start}} \\ x_{\text{try}} \\ x_{\text{lost}} \\ x_{\text{delivered}} \end{pmatrix} = \begin{pmatrix} 1-10^{-i} \\ 1-10^{-i} \\ 1-10^{-i} \\ 1 \end{pmatrix}$$

- In the limit,

$$\begin{pmatrix} x_{\text{start}} \\ x_{\text{try}} \\ x_{\text{lost}} \\ x_{\text{delivered}} \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

Value Iteration : Convergence

– Value iteration may not converge in any finite number of steps

Value Iteration : Convergence

- Value iteration may not converge in any finite number of steps
- Model checkers need to stop the iterations in a finite number of steps

Value Iteration : Convergence

- Value iteration may not converge in any finite number of steps
- Model checkers need to stop the iterations in a finite number of steps
- Common criteria : difference between successive vectors becomes small
  - Absolute criterion : $||V^{(i+1)} - V^{(i)}|| \leq \epsilon$
  - Relative criterion : $||\frac{V^{(i+1)} - V^{(i)}}{V^{(i)}}|| \leq \epsilon$

Value Iteration : Convergence

- Value iteration may not converge in any finite number of steps
- Model checkers need to stop the iterations in a finite number of steps
- Common criteria : difference between successive vectors becomes small
  - Absolute criterion : $||V^{(i+1)} - V^{(i)}|| \leq \epsilon$
  - Relative criterion : $||\frac{V^{(i+1)} - V^{(i)}}{V^{(i)}}|| \leq \epsilon$
- Problem : high magnitude changes are preceded by periods of stability

Value Iteration : Convergence

- Value iteration may not converge in any finite number of steps
- Model checkers need to stop the iterations in a finite number of steps
- Common criteria : difference between successive vectors becomes small
  - Absolute criterion : $||V^{(i+1)} - V^{(i)}|| \leq \epsilon$
  - Relative criterion : $||\frac{V^{(i+1)} - V^{(i)}}{V^{(i)}}|| \leq \epsilon$
- Problem : high magnitude changes are preceded by periods of stability
- Unknown quality of the resulting approximation

Exact Quantitative Model Checking

- Approximate solution techniques can lead to unreliable results
- Incorrect analysis of systems
- Verification tools must strive to get the exact answers
- Existing techniques for exact model checking:

1. Linear programming
2. Parametric model checking
3. State Elimination

} Implemented in state-of-the-art quantitative model checkers - PRISM, STORM, etc.,

Rational Search : Key Ideas

Rational Search Insight :

Rational Search : Key Ideas

Rational Search Insight :

1. When transition probabilities are rational, the exact solution vector also has rational entries

Rational Search : Key Ideas

Rational Search Insight :

1. When transition probabilities are rational, the exact solution vector also has rational entries

2. Approximate answers resulting from value iteration can be used to find the exact rational solution

Rational Search : Key Ideas

Rational Search Insight :

1. When transition probabilities are rational, the exact solution vector also has rational entries
2. Approximate answers resulting from value iteration can be used to find the exact rational solution
3. Checking if a rational vector is the correct solution is easy

Rational Search : Key Ideas

Rational Search Insight :

1. When transition probabilities are rational, the exact solution vector also has rational entries

2. Approximate answers resulting from value iteration can be used to find the exact rational solution

3. Checking if a rational vector is the correct solution is easy
   - Fixpoint check : $\overline{x} = A\overline{x} + \overline{b}$
   - Unique solution : Only the correct answer passes the check

Rational Search : Overview

Rational Search Overview :

Rational Search : Overview

Rational Search Overview :

1. Perform value iteration : approximate solution vector

Rational Search : Overview

Rational Search Overview :

1. Perform value iteration : approximate solution vector
2. 'Sharpen' approximation :

Rational Search : Overview

Rational Search Overview :

1. Perform value iteration : approximate solution vector
2. 'Sharpen' approximation :
   - find a rational vector
   - close to the approximate solution
   - representable using few bits
   - guaranteed to be correct for good quality approximations

Rational Search : Overview

Rational Search Overview :

1. Perform value iteration : approximate solution vector
2. 'Sharpen' approximation :
   - find a rational vector
   - close to the approximate solution
   - representable using few bits
   - guaranteed to be correct for good quality approximations
3. Confirm using fix-point check : $\overline{x} = A\overline{x} + \overline{b}$

Rational Search : Overview

Rational Search Overview :

1. Perform value iteration : approximate solution vector
2. 'Sharpen' approximation :
   - find a rational vector
   - close to the approximate solution
   - representable using few bits
   - guaranteed to be correct for good quality approximations
3. Confirm using fix-point check : $\overline{x} = A\overline{x} + \overline{b}$
4. Refine approximation if not a fixpoint

Rational Search : Overview

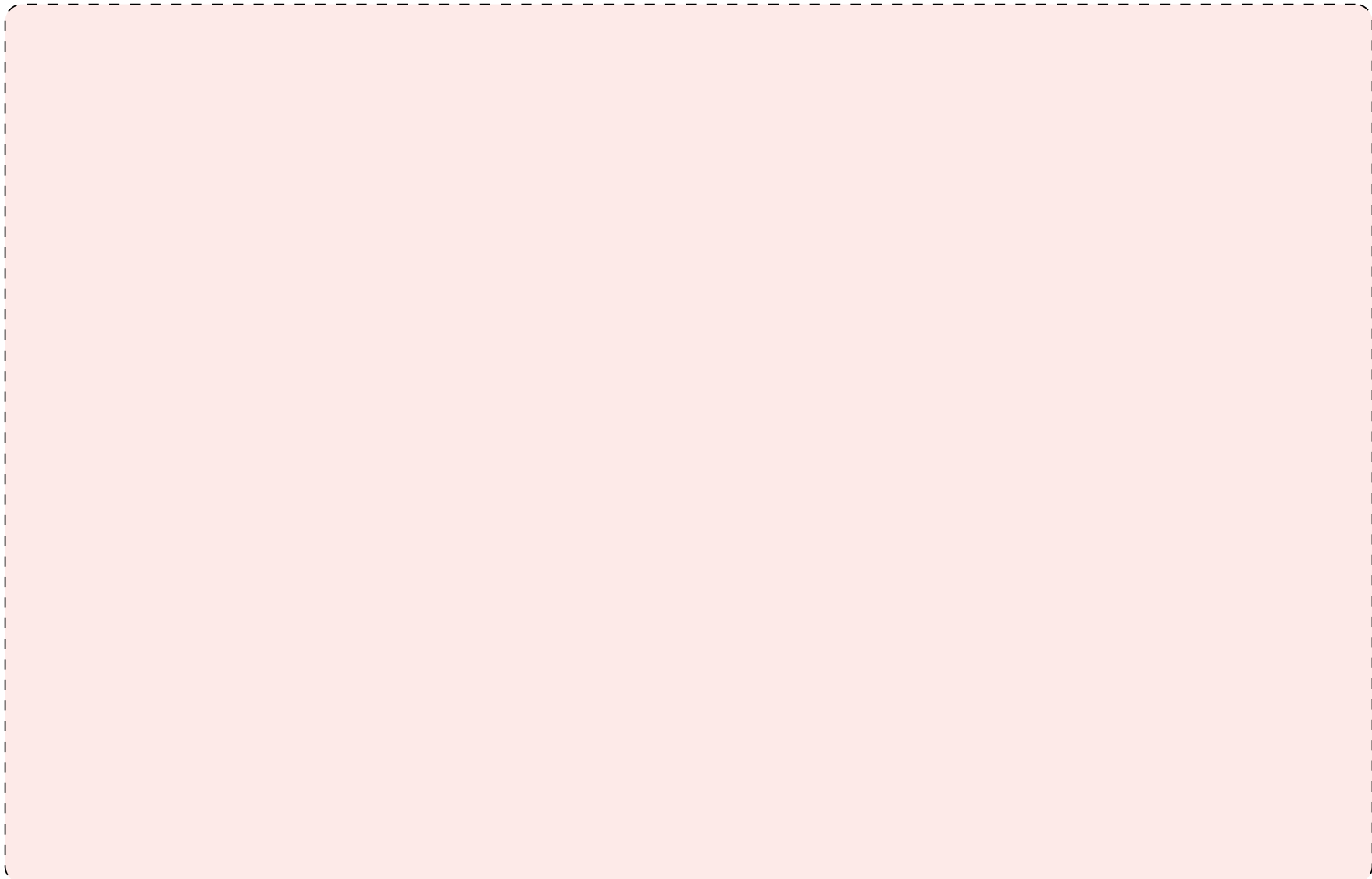


Figure: RationalSearch : Overview

Rational Search : Overview

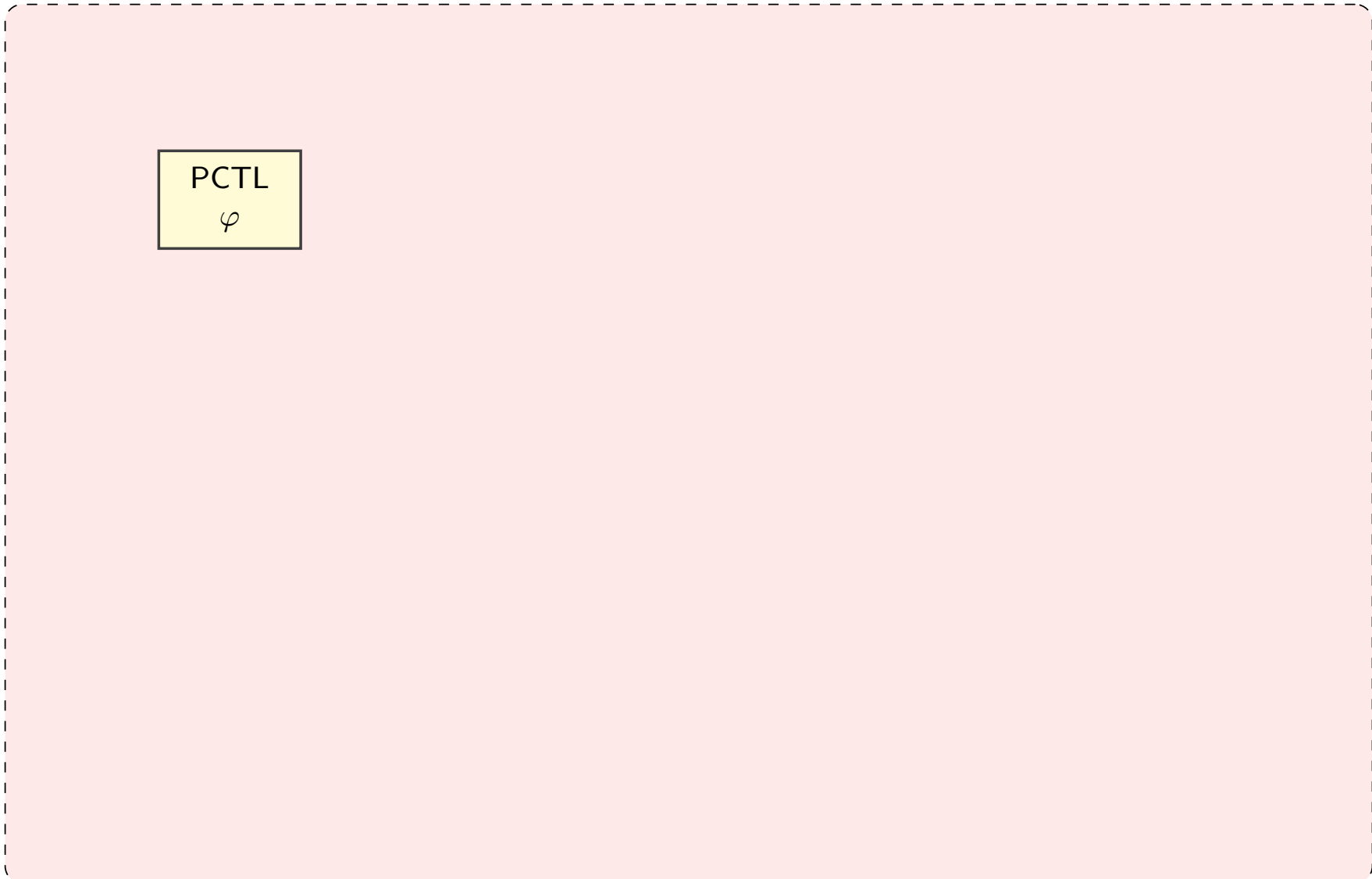


Figure: RationalSearch : Overview

Rational Search : Overview

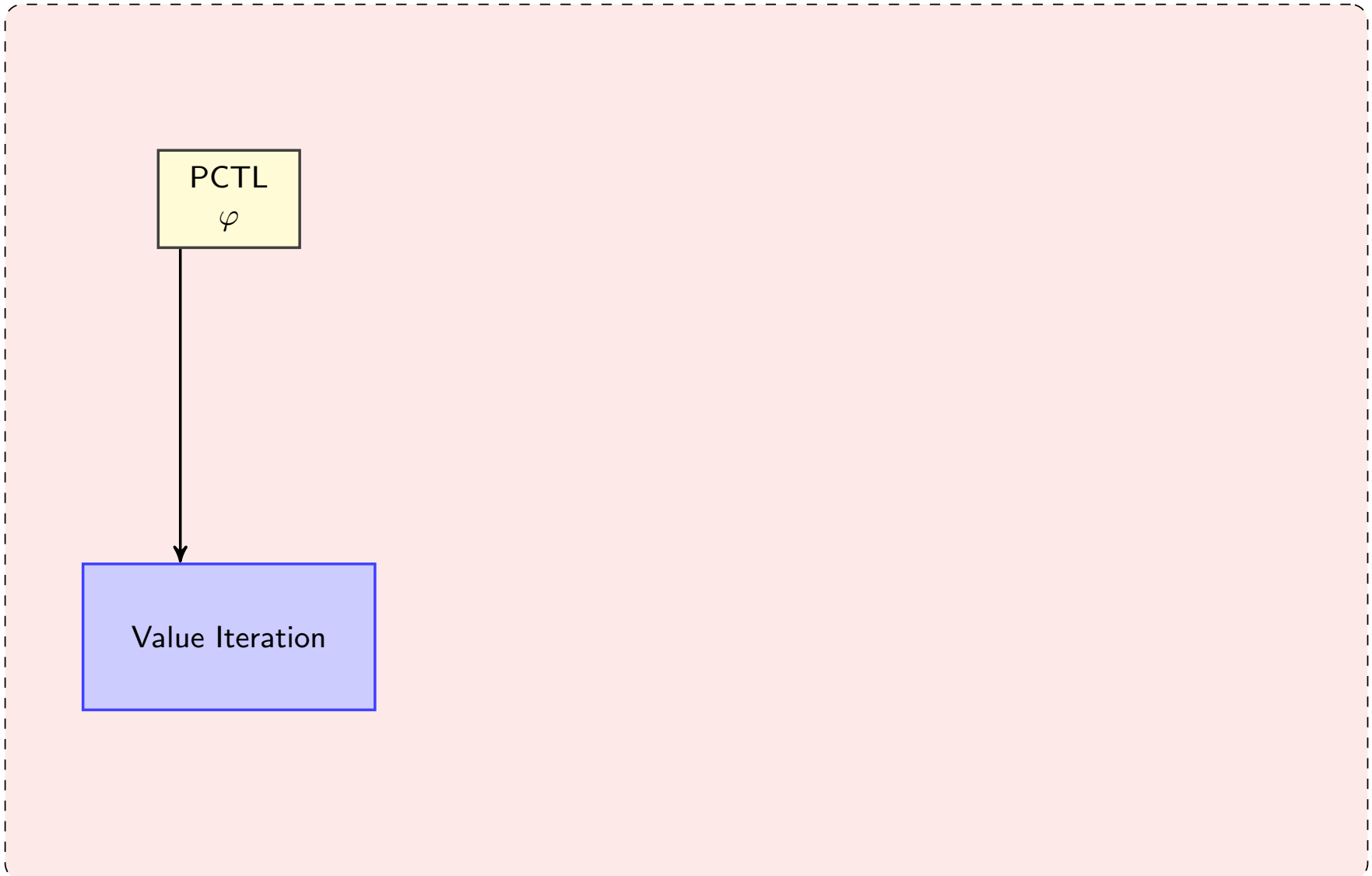


Figure: RationalSearch : Overview

Rational Search : Overview

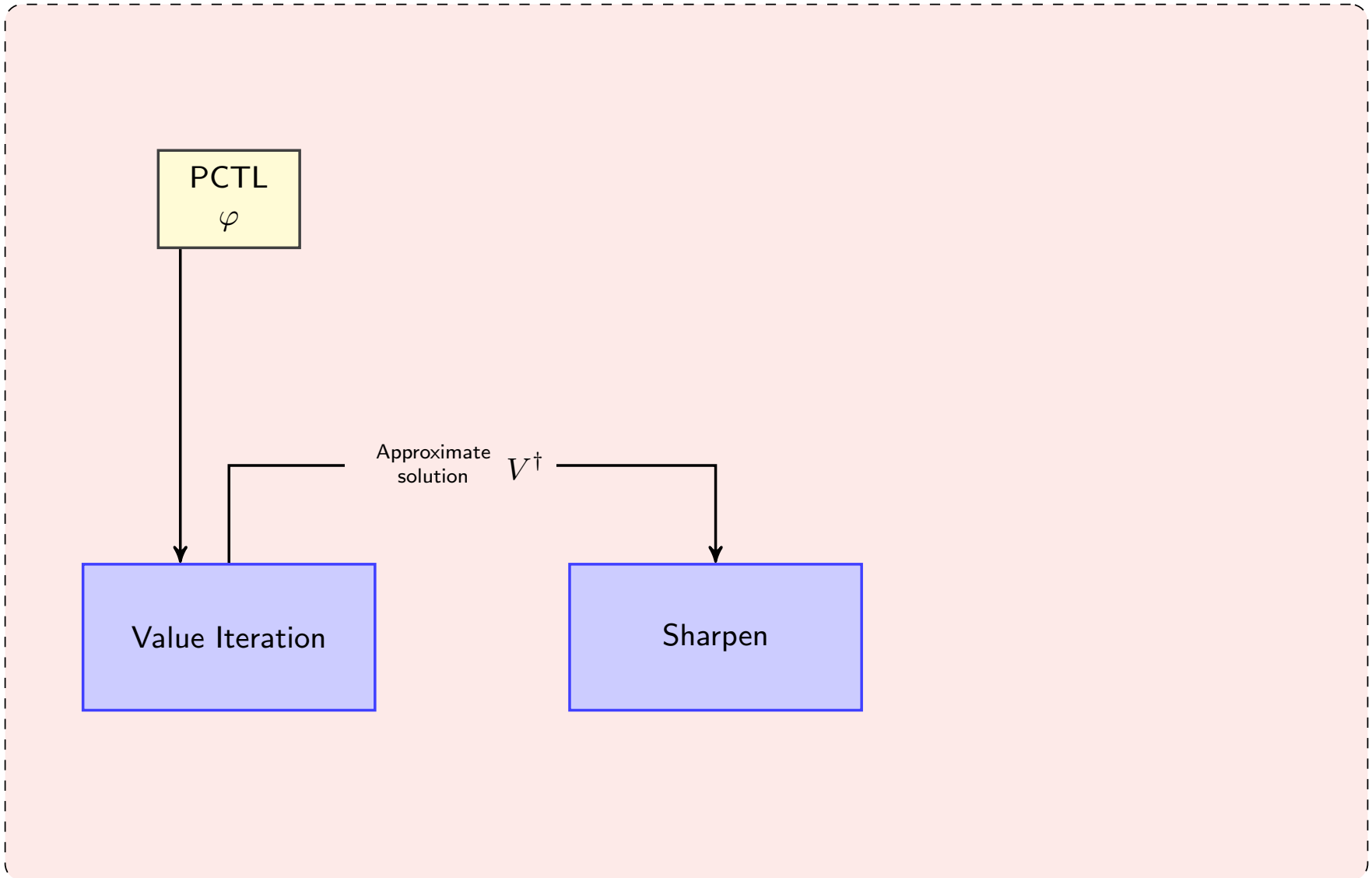


Figure: RationalSearch : Overview

Rational Search : Overview

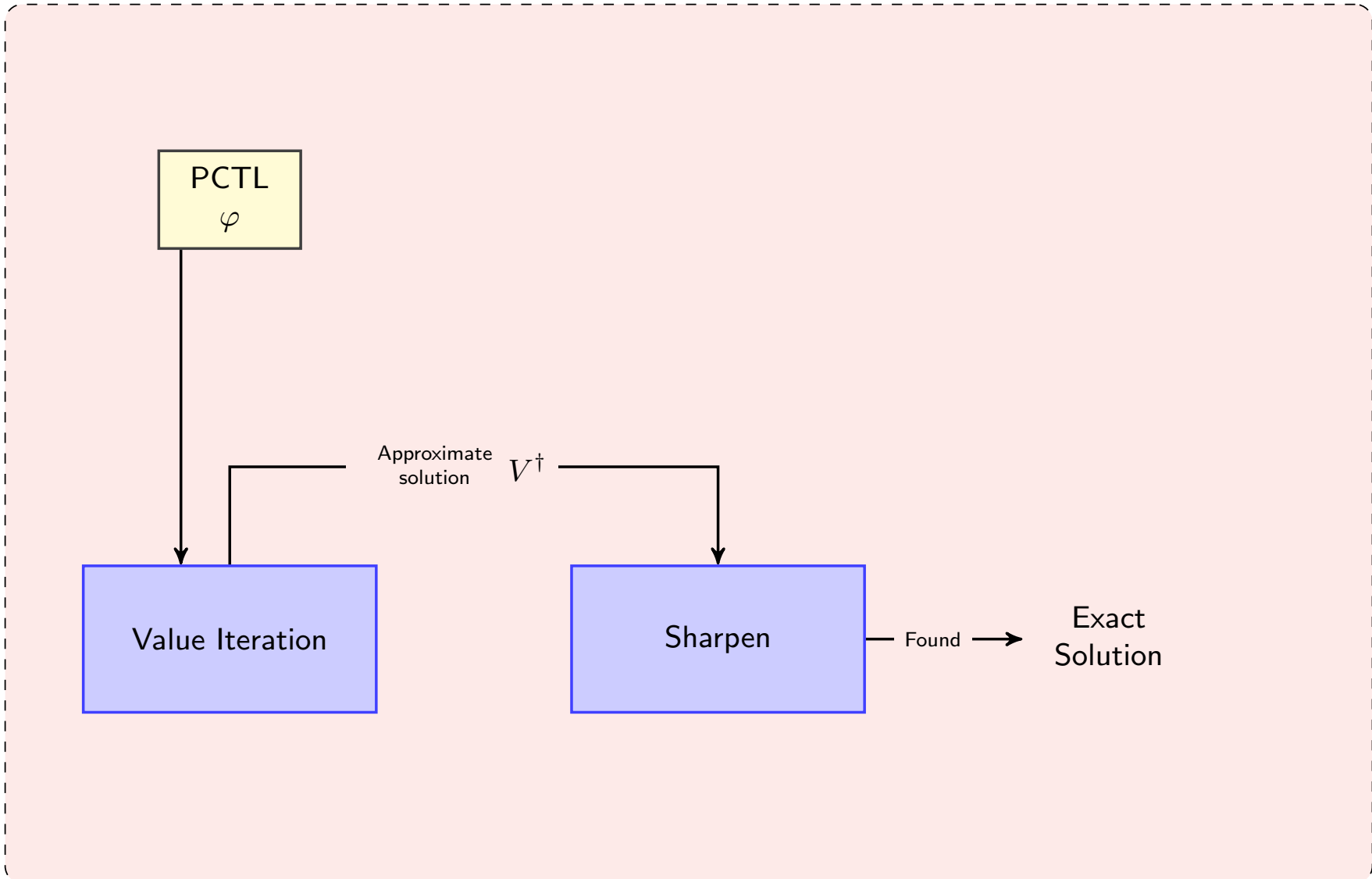


Figure: RationalSearch : Overview

Rational Search : Overview

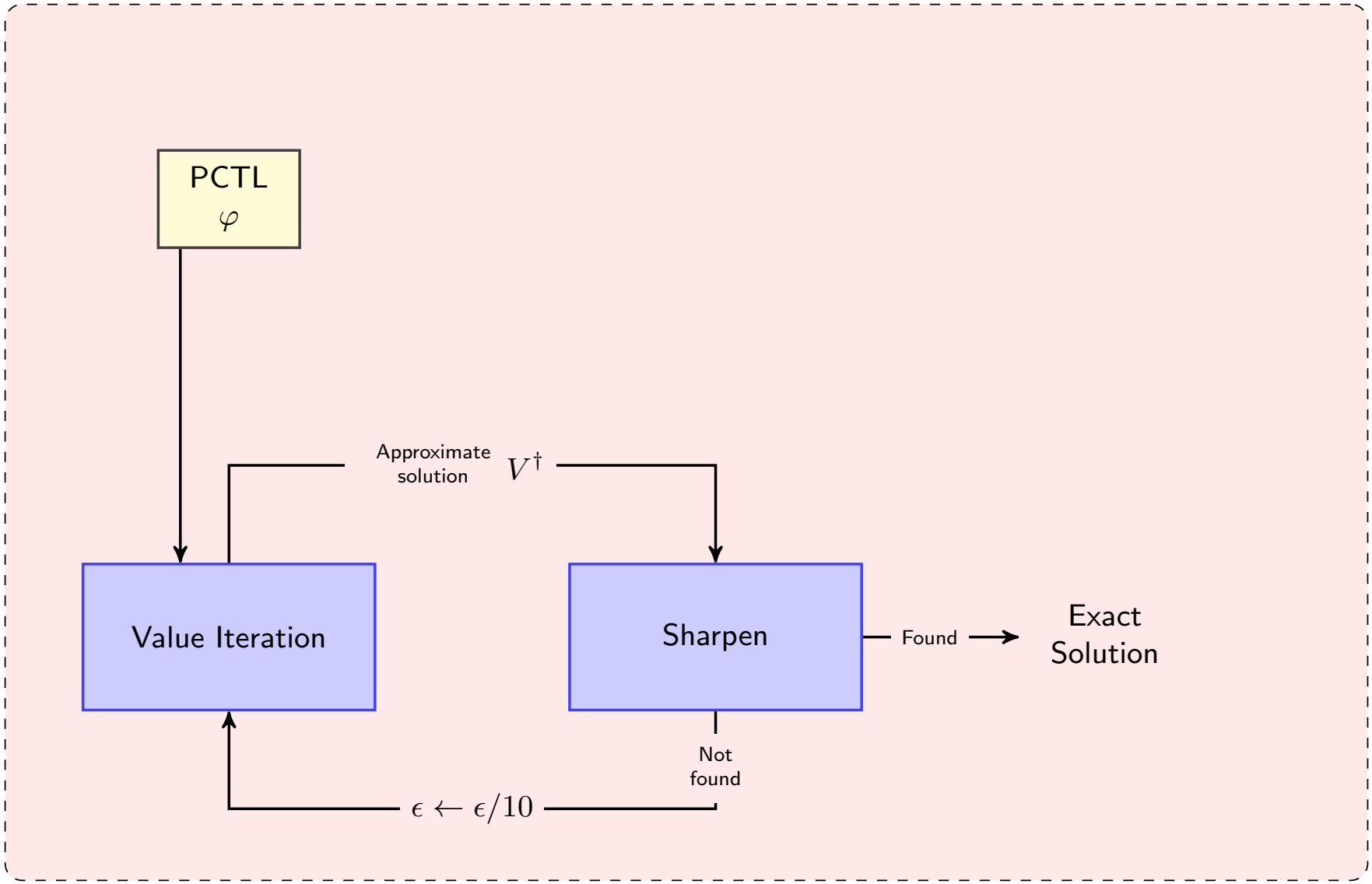


Figure: RationalSearch : Overview

Rational Search : Overview

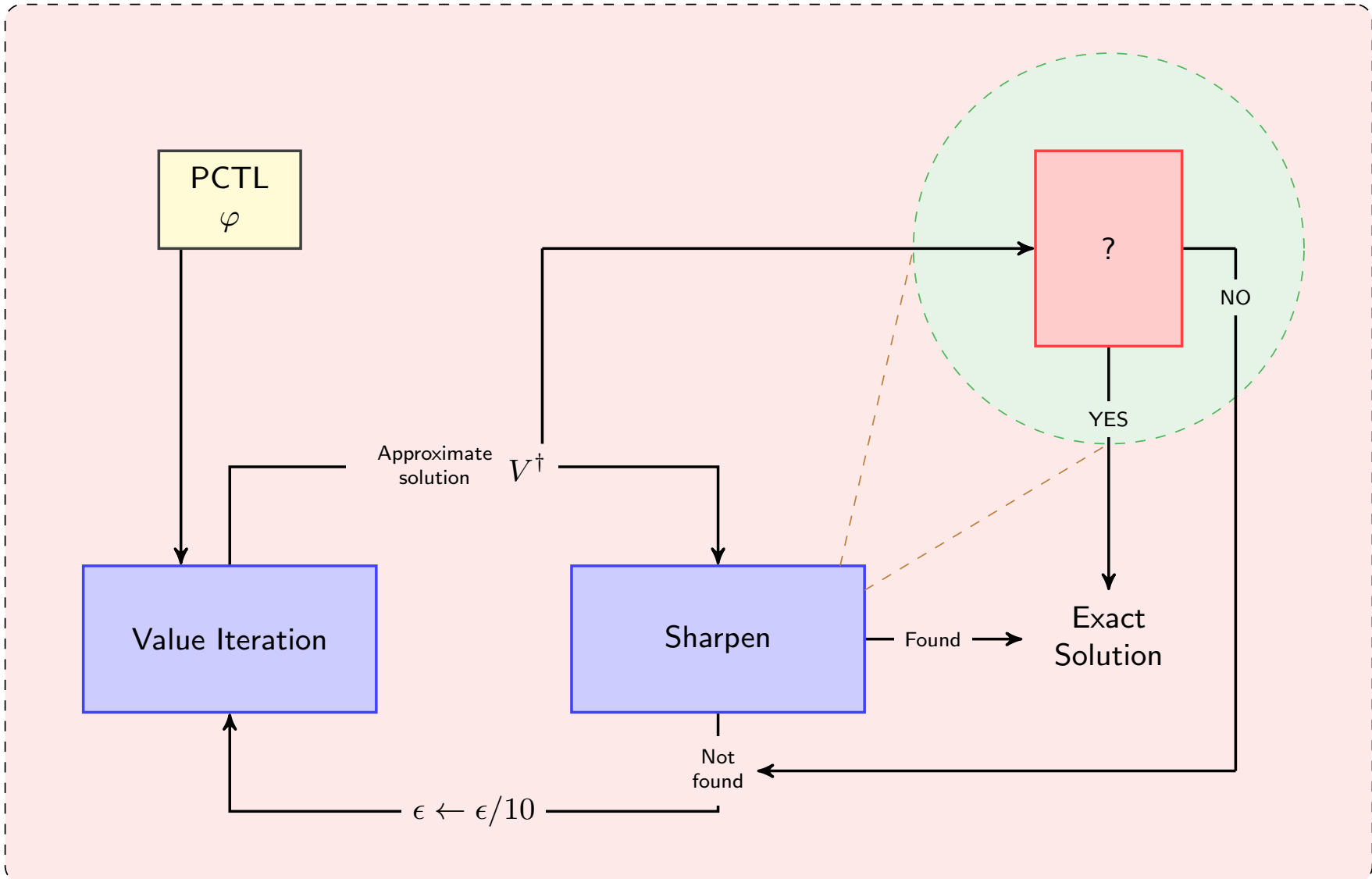


Figure: RationalSearch : Overview

Our Secret Ingredient : Kwek Mehlhorn Algorithm

For any interval $I = [\frac{\alpha}{\beta}, \frac{\gamma}{\delta}]$ with rational endpoints, there is a unique minimal rational $p_{\text{min}}/q_{\text{min}} \in I$ such that

$$\forall p, q \in \mathbb{N}, \frac{p}{q} \in I \implies p_{\text{min}} \leq p \text{ and } q_{\text{min}} \leq q$$

Our Secret Ingredient : Kwek Mehlhorn Algorithm

For any interval $I = [\frac{\alpha}{\beta}, \frac{\gamma}{\delta}]$ with rational endpoints, there is a unique minimal rational $p_{\text{min}}/q_{\text{min}} \in I$ such that

$$\forall p, q \in \mathbb{N}, \frac{p}{q} \in I \implies p_{\text{min}} \leq p \text{ and } q_{\text{min}} \leq q$$

Efficient algorithm to locate $p_{\text{min}}/q_{\text{min}}$ in I, due to Kwek and Mehlhorn.

Our Secret Ingredient : Kwek Mehlhorn Algorithm

For any interval $I = [\frac{\alpha}{\beta}, \frac{\gamma}{\delta}]$ with rational endpoints, there is a unique minimal rational $p_{\text{min}}/q_{\text{min}} \in I$ such that

$$\forall p, q \in \mathbb{N}, \frac{p}{q} \in I \implies p_{\text{min}} \leq p \text{ and } q_{\text{min}} \leq q$$

Efficient algorithm to locate $p_{\text{min}}/q_{\text{min}}$ in I, due to Kwek and Mehlhorn.

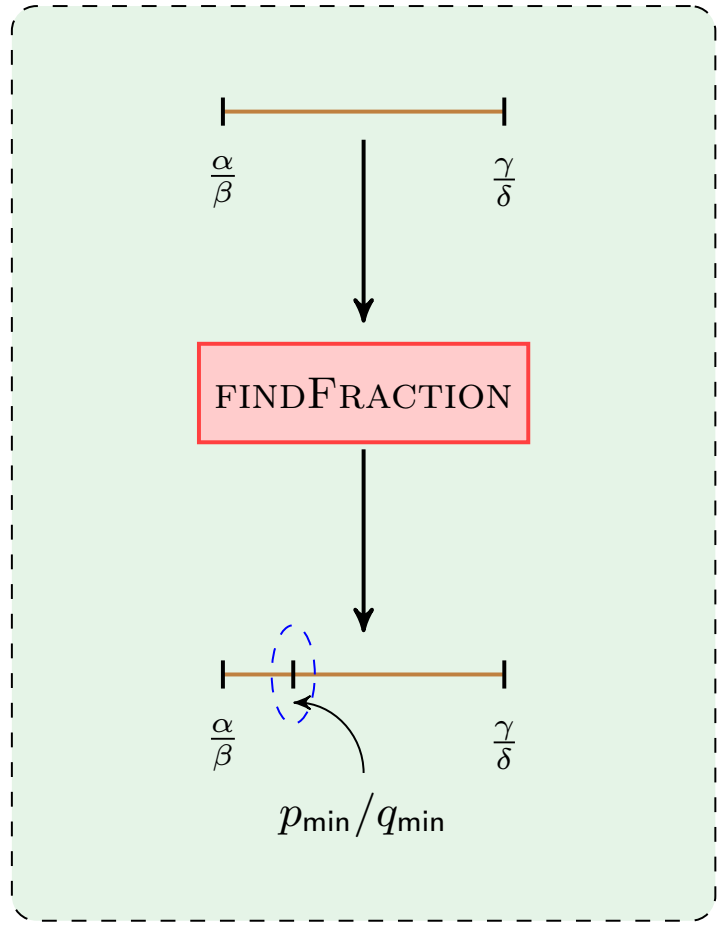


Figure: Kwek Mehlhorn Algorithm

Our Secret Ingredient : Kwek Mehlhorn Algorithm

For a rational interval I,

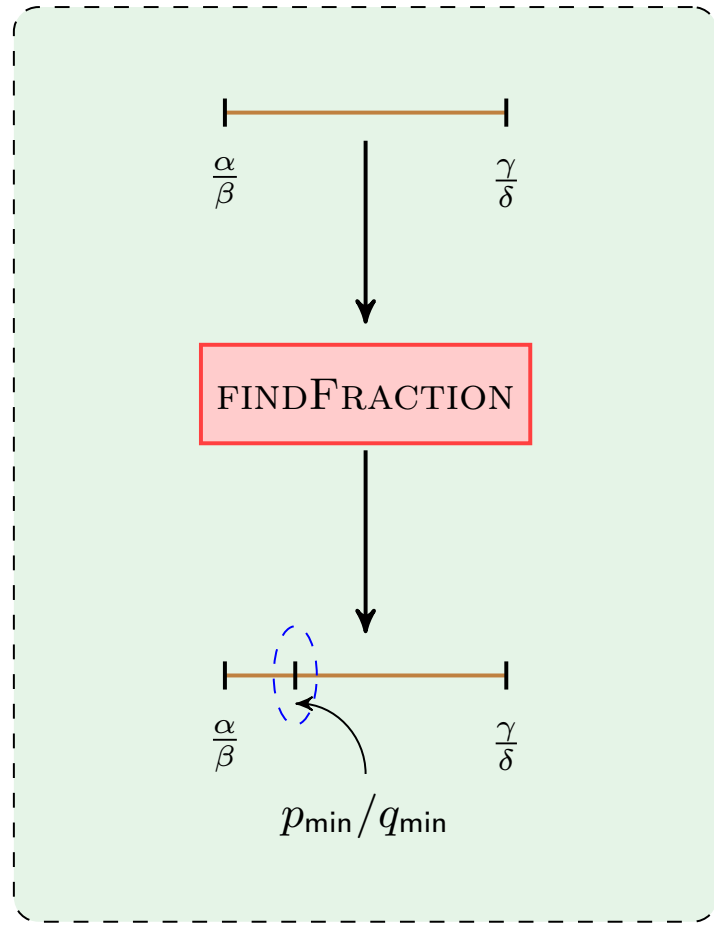


Figure: Kwek Mehlhorn Algorithm

Our Secret Ingredient : Kwek Mehlhorn Algorithm

For a rational interval I,

- If the length of I is small ($I = [\frac{\mu}{2M^2}, \frac{\mu+1}{2M^2}]$), and

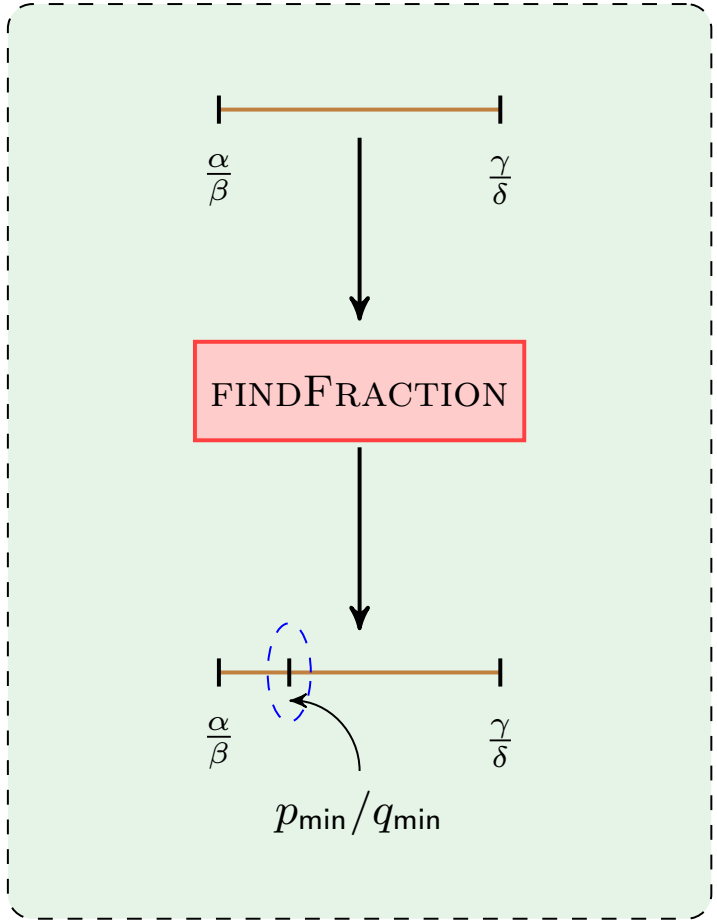


Figure: Kwek Mehlhorn Algorithm

Our Secret Ingredient : Kwek Mehlhorn Algorithm

For a rational interval I,

- If the length of I is small ($I = [\frac{\mu}{2M^2}, \frac{\mu+1}{2M^2}]$), and
- If I contains a rational number p/q of small size ($1 \leq p \leq q \leq M$)

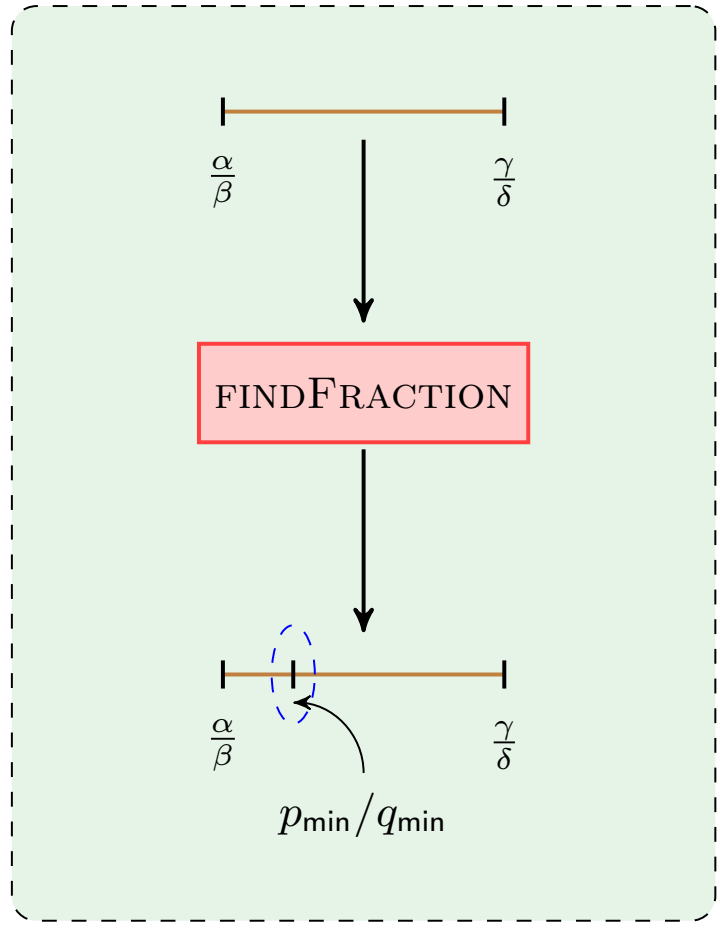


Figure: Kwek Mehlhorn Algorithm

Our Secret Ingredient : Kwek Mehlhorn Algorithm

For a rational interval I,

- If the length of I is small ($I = [\frac{\mu}{2M^2}, \frac{\mu+1}{2M^2}]$), and
- If I contains a rational number p/q of small size ($1 \leq p \leq q \leq M$)
- Then, p/q is the minimal rational in I

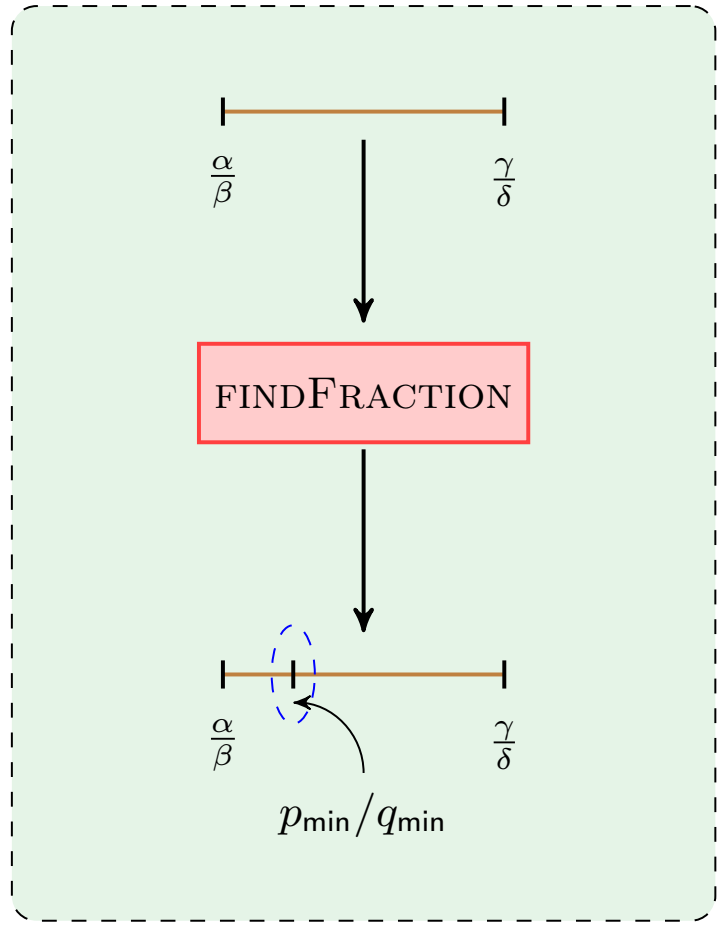


Figure: Kwek Mehlhorn Algorithm

Our Secret Ingredient : Kwek Mehlhorn Algorithm

For a rational interval I,

- If the length of I is small ($I = [\frac{\mu}{2M^2}, \frac{\mu+1}{2M^2}]$), and
- If I contains a rational number p/q of small size ($1 \leq p \leq q \leq M$)
- Then, p/q is the minimal rational in I
- Can be found efficiently (in $O(\log M)$ steps) due to Kwek, Mehlhorn et. al.

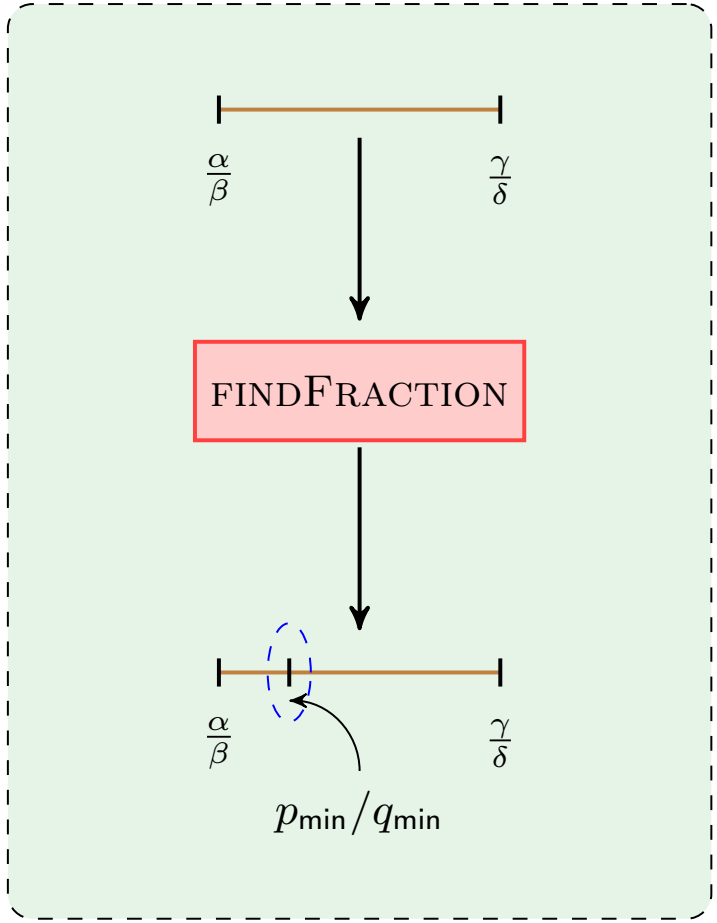


Figure: Kwek Mehlhorn Algorithm

Sharpening An Approximation

Sharpening an approximation :

Figure: Sharpening an approximation

Sharpening An Approximation

$$V^{\dagger}(z) = 0.18 \ldots 33120 \ldots$$

Sharpening an approximation :

1. Value iteration gives approximate vector $V^{\dagger}$

Figure: Sharpening an approximation

Sharpening An Approximation

$$V^{\dagger}(z) = 0.18 \ldots 33120 \ldots$$

Sharpening an approximation :

1. Value iteration gives approximate vector $V^{\dagger}$
   - Supposedly close to the actual solution vector V

Figure: Sharpening an approximation

Sharpening An Approximation

$$V^{\dagger}(z) = 0.18 \ldots 33120 \ldots$$

Sharpening an approximation :

1. Value iteration gives approximate vector $V^{\dagger}$
   - Supposedly close to the actual solution vector V
   - How close?

Figure: Sharpening an approximation

Sharpening An Approximation

$$V^{\dagger}(z) = 0.18\ldots33120\ldots$$

Sharpening an approximation :

1. Value iteration gives approximate vector $V^{\dagger}$
   - Supposedly close to the actual solution vector V
   - How close?
2. Guess a rational vector V^{*} of small size close to $V^{\dagger}$

Figure: Sharpening an approximation

Sharpening An Approximation

Sharpening an approximation :

1. Value iteration gives approximate vector $V^\dagger$
   - Supposedly close to the actual solution vector V
   - How close?
2. Guess a rational vector V^* of small size close to $V^\dagger$
   - For every state z, construct an interval I_z using first d digits of $V^\dagger(z)$

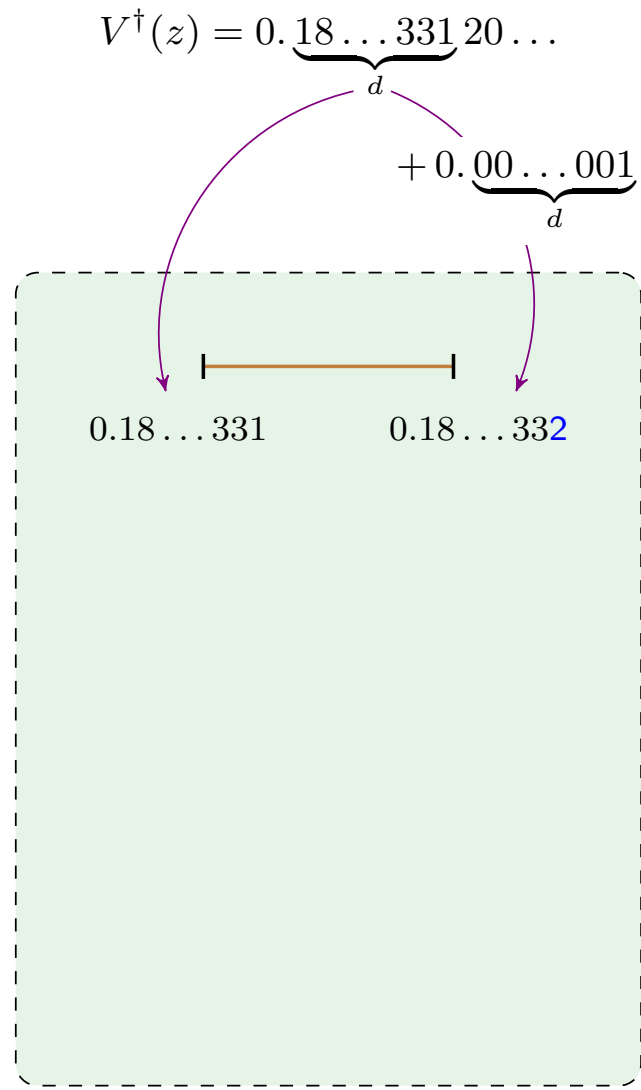


Figure: Sharpening an approximation

Sharpening An Approximation

Sharpening an approximation :

1. Value iteration gives approximate vector $V^\dagger$
   - Supposedly close to the actual solution vector V
   - How close?
2. Guess a rational vector V^* of small size close to $V^\dagger$
   - For every state z, construct an interval I_z using first d digits of $V^\dagger(z)$
   - Compute minimal fraction in this interval, using Kwek Mehlhorn algorithm

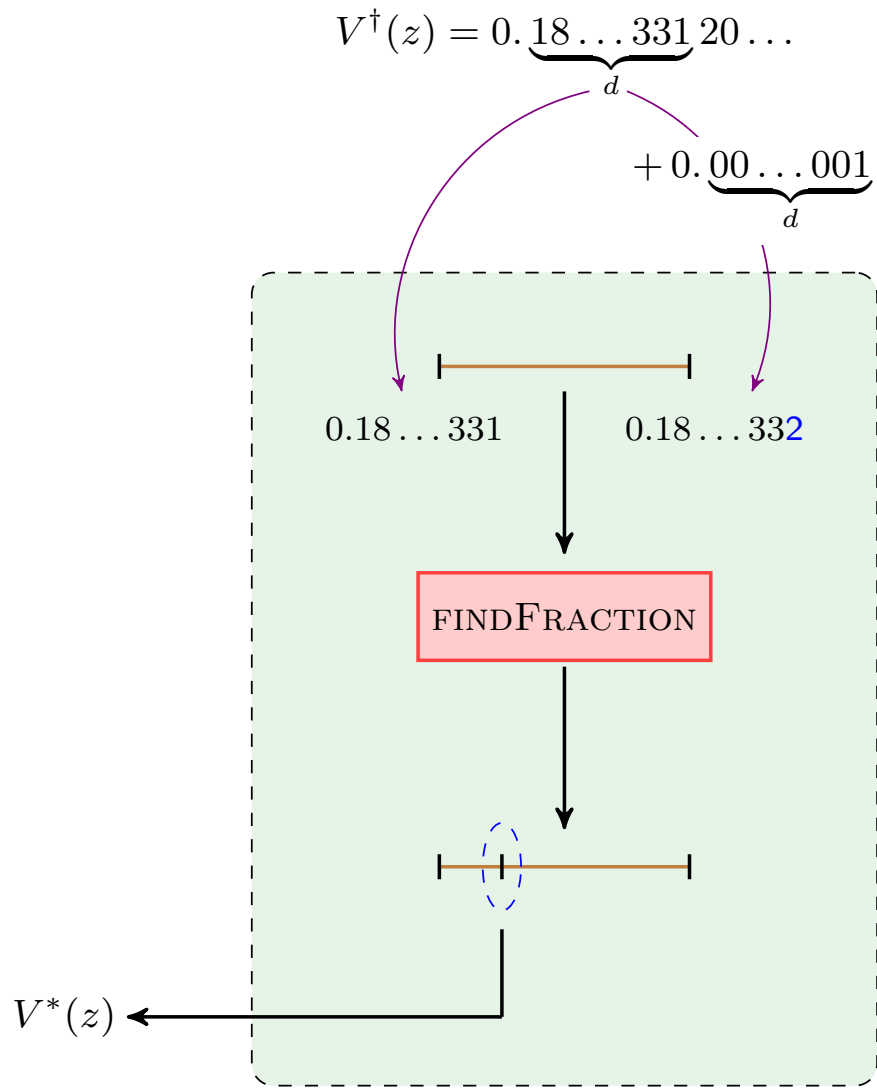


Figure: Sharpening an approximation

Sharpening An Approximation

Is V^* the correct solution ?

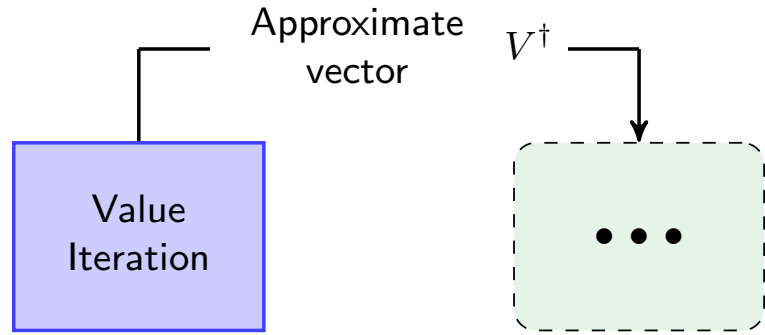


Figure: Sharpening an approximation

Sharpening An Approximation

Is V^* the correct solution ?

- Check if V^* satisfies the fixpoint check

$$V^* = A \cdot V^* + \overline{b}$$

- Only the correct solution passes this check

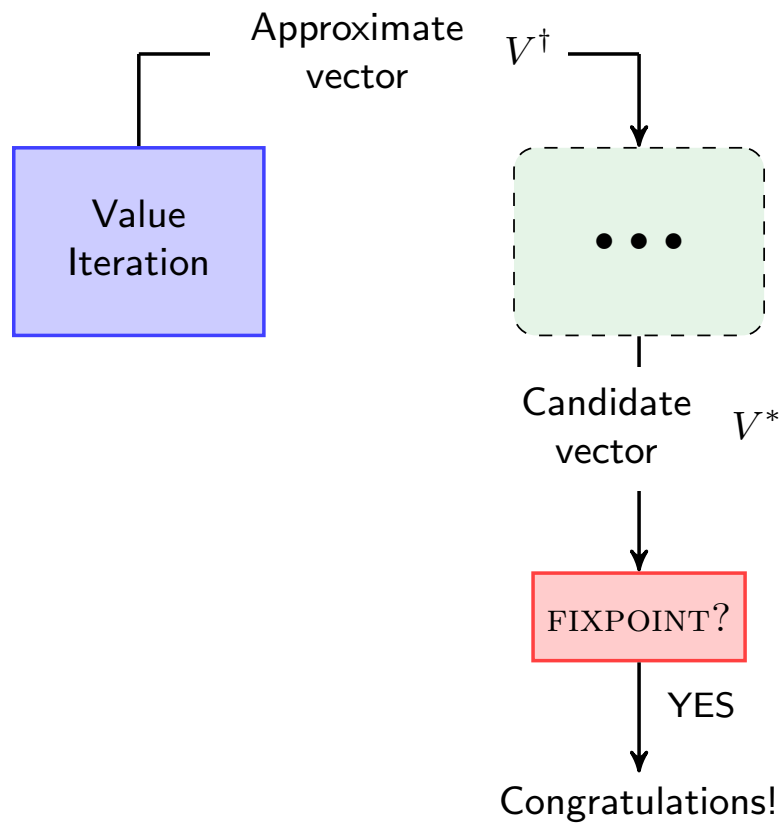


Figure: Sharpening an approximation

Sharpening An Approximation

Is V^* the correct solution ?

- Check if V^* satisfies the fixpoint check

$$V^* = A \cdot V^* + \overline{b}$$

- Only the correct solution passes this check

If V^* does not pass the check :

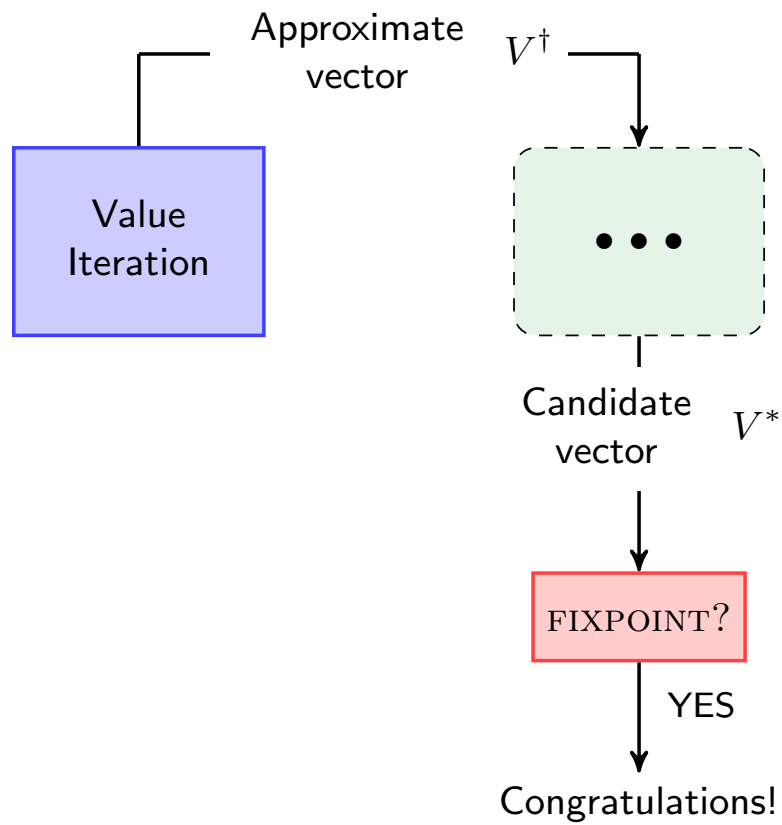


Figure: Sharpening an approximation

Sharpening An Approximation

Is V^* the correct solution ?

- Check if V^* satisfies the fixpoint check

$$V^* = A \cdot V^* + \bar{b}$$

- Only the correct solution passes this check

If V^* does not pass the check :

- 'Bad' initial approximation $V^\dagger$

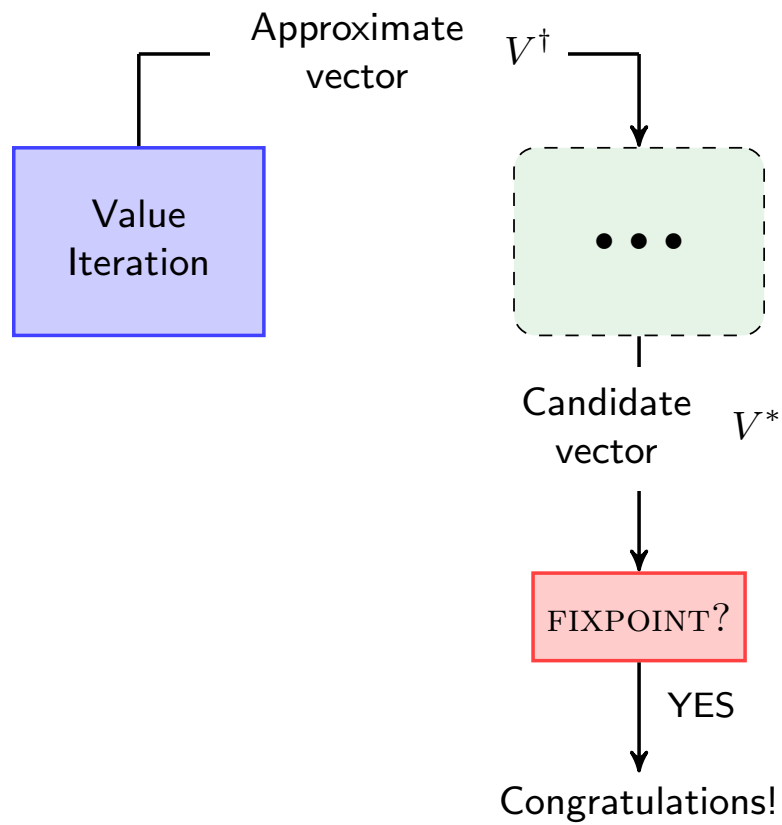


Figure: Sharpening an approximation

Sharpening An Approximation

Is V^* the correct solution ?

- Check if V^* satisfies the fixpoint check

$$V^* = A \cdot V^* + \overline{b}$$

- Only the correct solution passes this check

If V^* does not pass the check :

- 'Bad' initial approximation $V^\dagger$
- Generate a finer approximation by performing more iterations

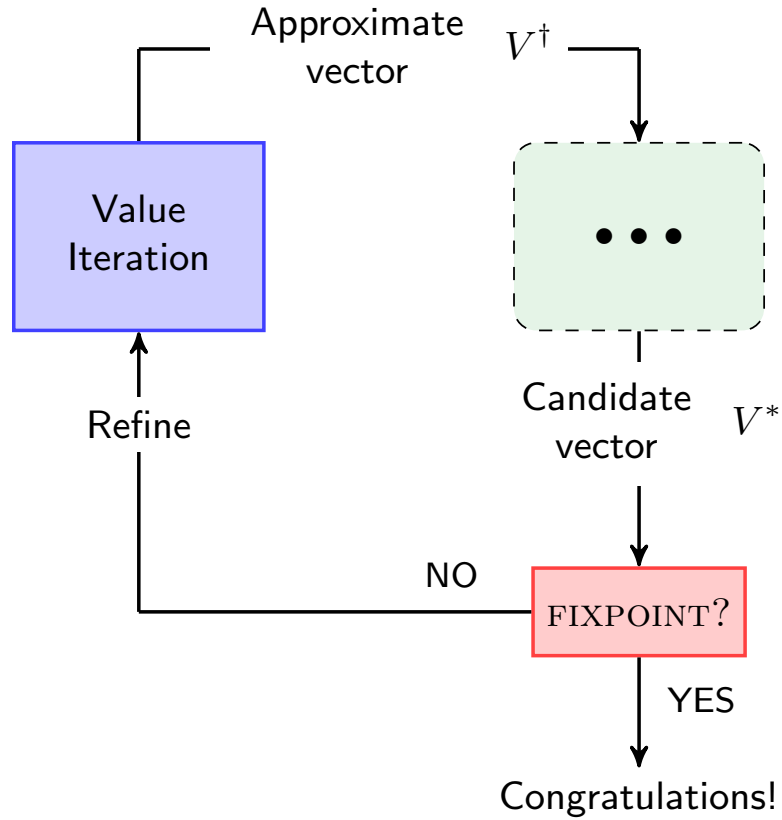


Figure: Sharpening an approximation

Sharpening An Approximation

Is V^* the correct solution ?

- Check if V^* satisfies the fixpoint check

$$V^* = A \cdot V^* + \bar{b}$$

- Only the correct solution passes this check

If V^* does not pass the check :

- 'Bad' initial approximation $V^\dagger$
- Generate a finer approximation by performing more iterations
- Eventually, a 'good' approximation will be generated : Value iteration converges in the limit

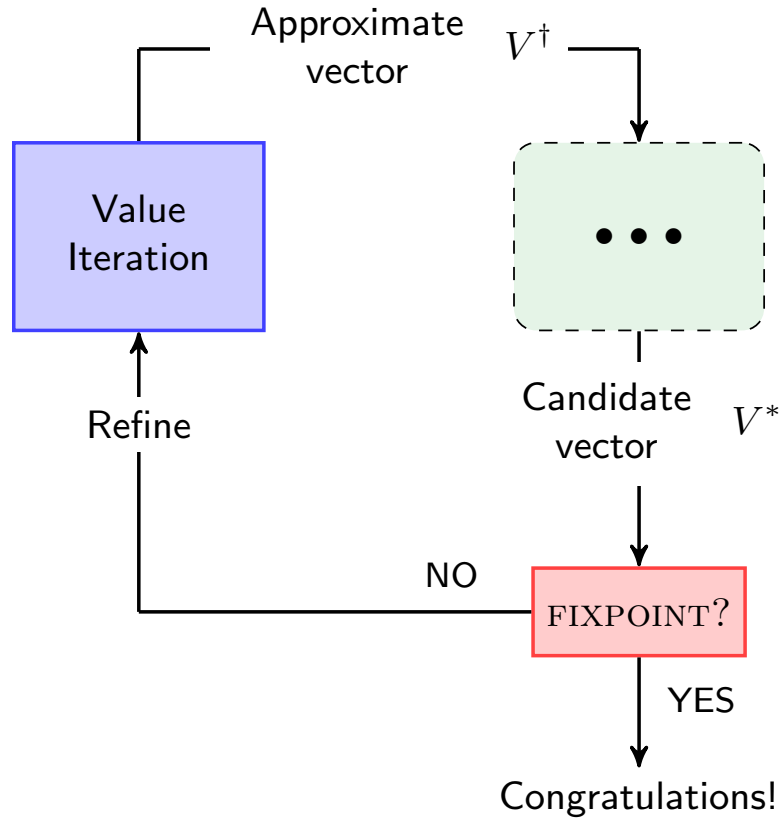


Figure: Sharpening an approximation

Sharpening An Approximation

Is V^* the correct solution ?

- Check if V^* satisfies the fixpoint check

$$V^* = A \cdot V^* + \bar{b}$$

- Only the correct solution passes this check

If V^* does not pass the check :

- 'Bad' initial approximation $V^\dagger$
- Generate a finer approximation by performing more iterations
- Eventually, a 'good' approximation will be generated : Value iteration converges in the limit

Details in the paper

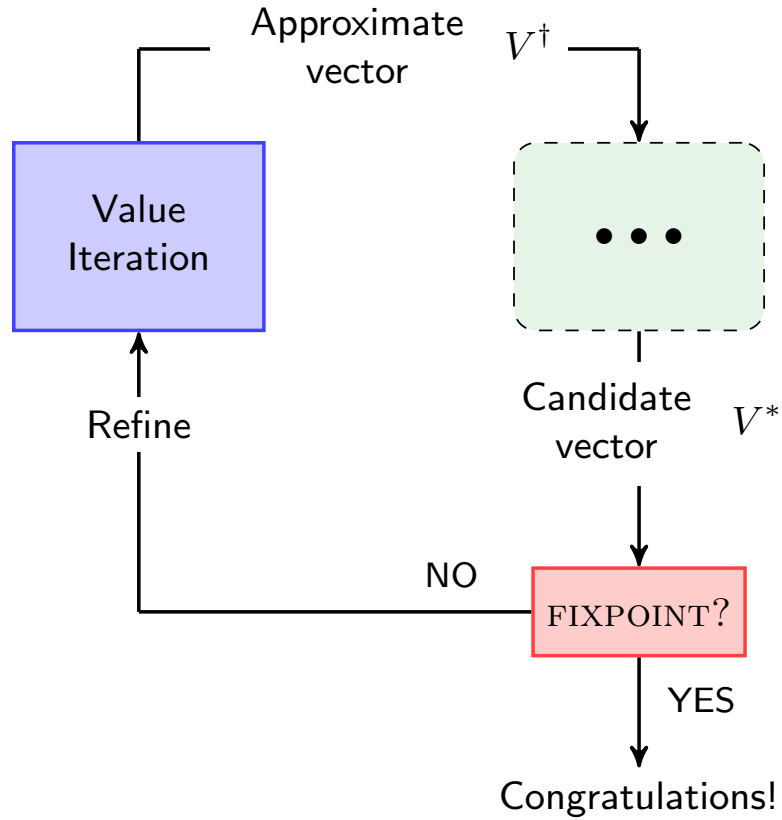


Figure: Sharpening an approximation

Rational Search : Recap

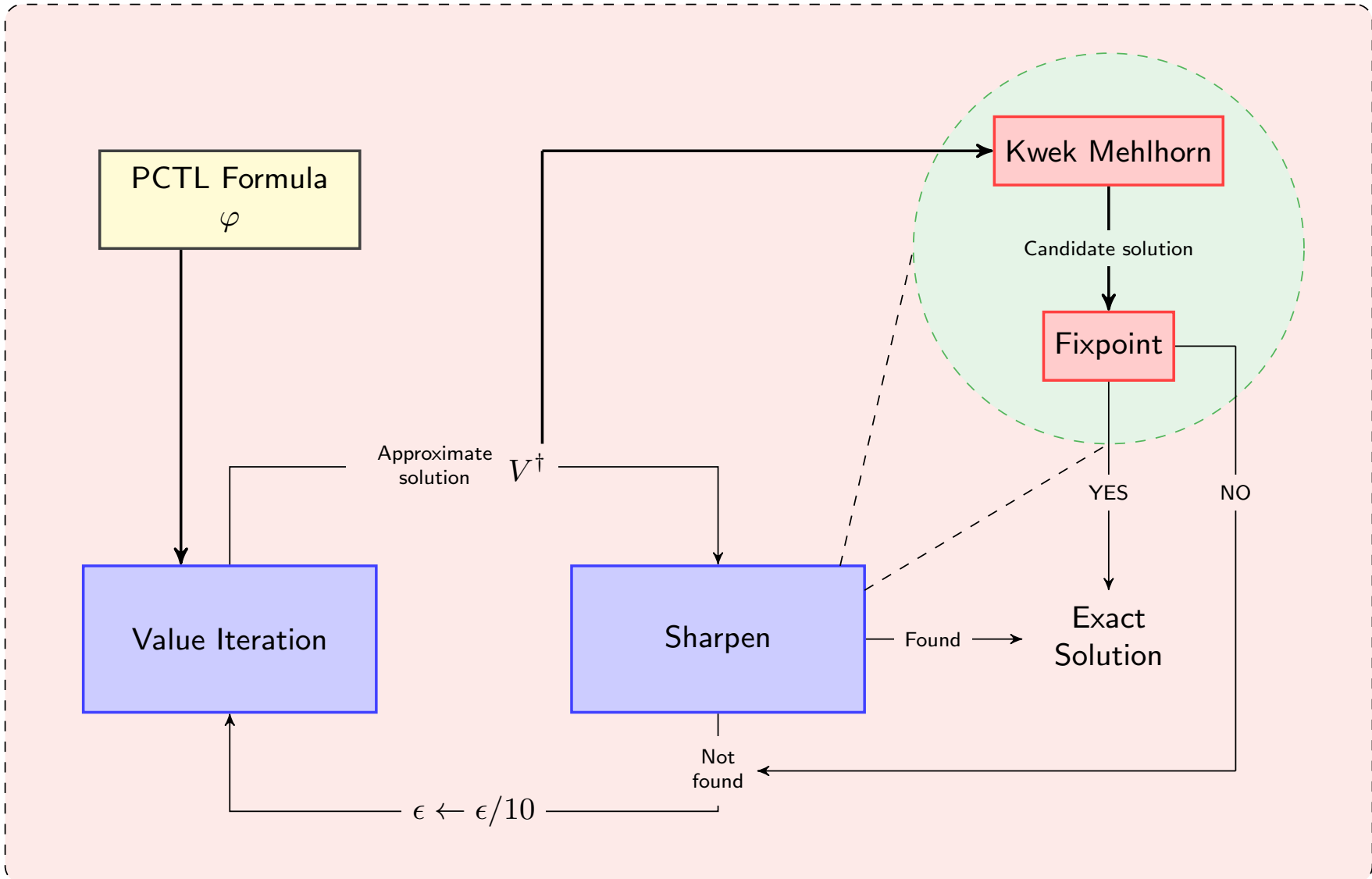


Figure: RationalSearch : Overview

Implementation

– Tool RationalSearch, implemented on top of PRISM 

Implementation

- Tool RationalSearch, implemented on top of PRISM 

- RationalSearch intercepts PRISM's value iteration phase, and rationalizes the values

Implementation

- Tool RationalSearch, implemented on top of PRISM 

- RationalSearch intercepts PRISM's value iteration phase, and rationalizes the values
- Extending PRISM's engines :

Implementation

- Tool RationalSearch, implemented on top of PRISM 

- RationalSearch intercepts PRISM's value iteration phase, and rationalizes the values
- Extending PRISM's engines :
  1. Explicit engine
     - Arbitrary precision libraries for Java : Apfloat, JScience
  2. Symbolic engines : MTBDD, Hybrid and Sparse
     - Value iteration phase uses MTBDDs from CUDD library (written in C)
     - Extended CUDD to handle arbitrary precision rational numbers at leaf nodes
     - GNU MP library in C

Implementation

- Tool RationalSearch, implemented on top of PRISM 

- RationalSearch intercepts PRISM's value iteration phase, and rationalizes the values
- Extending PRISM's engines :
  1. Explicit engine
     - Arbitrary precision libraries for Java : Apfloat, JScience
  2. Symbolic engines : MTBDD, Hybrid and Sparse
     - Value iteration phase uses MTBDDs from CUDD library (written in C)
     - Extended CUDD to handle arbitrary precision rational numbers at leaf nodes
     - GNU MP library in C
- Available for download :
  https://publish.illinois.edu/rationalmodelchecker/

Evaluation

1	2	3	4	5	6	7	8	9	10	11
Model			RationalSearch						PRISM	STORM
Name	Parameter	States	Explicit		MTBDD		Hybrid			
			Time (s)	Overhead (%)	Time (s)	Overhead (%)	Time (s)	Overhead (%)	Time (s)	Time (s)
Biased Coins	11	180K	23.1	336	0.125	179	0.178	225	1449.7	3.2
Dice	6	4.8M	OOM	N/A	1.8	2.1	6.5	12	TO	63
Din. Cryptographers	8	190K	18.9	197	0.278	70	0.364	105	356.2	2.4
Din. Philosophers	3	956	0.41	165	1.9	4.8	0.133	98	3.128	0.65
ECS	14	4.8M	OOM	N/A	2.4	23	11.6	79	TO	TO
Fair Exchange	400	320K	14.6	423	2.0	44	2.2	51	TO	1.1
Firewire	11,000	430K	122.2	225	15.1	0.2	19.5	21	232.3	29.5
Leader Election	4	12K	1.8	226	5.0	30	20.4	25	80	0.042
Virus Infection	3	809	0.5	165	2.8	52	0.17	93	0.98	0.032

Figure: Experimental Evaluation

Evaluation

1	2	3	4	5	6	7	8	9	10	11
Model			RationalSearch						PRISM	STORM
			Explicit		MTBDD		Hybrid			
Name	Parameter	States	Time (s)	Overhead (%)	Time (s)	Overhead (%)	Time (s)	Overhead (%)	Time (s)	Time (s)
Biased Coins	11	180K	23.1	336	0.125	179	0.178	225	1449.7	3.2
Dice	6	4.8M	OOM	N/A	1.8	2.1	6.5	12	TO	63
Din. Cryptographers	8	190K	18.9	197	0.278	70	0.364	105	356.2	2.4
Din. Philosophers	3	956	0.41	165	1.9	4.8	0.133	98	3.128	0.65
ECS	14	4.8M	OOM	N/A	2.4	23	11.6	79	TO	TO
Fair Exchange	400	320K	14.6	423	2.0	44	2.2	51	TO	1.1
Firewire	11,000	430K	122.2	225	15.1	0.2	19.5	21	232.3	29.5
Leader Election	4	12K	1.8	226	5.0	30	20.4	25	80	0.042
Virus Infection	3	809	0.5	165	2.8	52	0.17	93	0.98	0.032

Figure: Experimental Evaluation

Evaluation

1	2	3	4	5	6	7	8	9	10	11
Model			RationalSearch						PRISM	STORM
Name	Parameter	States	Explicit		MTBDD		Hybrid			
			Time (s)	Overhead (%)	Time (s)	Overhead (%)	Time (s)	Overhead (%)	Time (s)	Time (s)
Biased Coins	11	180K	23.1	336	0.125	179	0.178	225	1449.7	3.2
Dice	6	4.8M	OOM	N/A	1.8	2.1	6.5	12	TO	63
Din. Cryptographers	8	190K	18.9	197	0.278	70	0.364	105	356.2	2.4
Din. Philosophers	3	956	0.41	165	1.9	4.8	0.133	98	3.128	0.65
ECS	14	4.8M	OOM	N/A	2.4	23	11.6	79	TO	TO
Fair Exchange	400	320K	14.6	423	2.0	44	2.2	51	TO	1.1
Firewire	11,000	430K	122.2	225	15.1	0.2	19.5	21	232.3	29.5
Leader Election	4	12K	1.8	226	5.0	30	20.4	25	80	0.042
Virus Infection	3	809	0.5	165	2.8	52	0.17	93	0.98	0.032

Figure: Experimental Evaluation

Evaluation

1	2	3	4	5	6	7	8	9	10	11
Model			RationalSearch						PRISM	STORM
Name	Parameter	States	Explicit		MTBDD		Hybrid			
			Time (s)	Overhead (%)	Time (s)	Overhead (%)	Time (s)	Overhead (%)	Time (s)	Time (s)
Biased Coins	11	180K	23.1	336	0.125	179	0.178	225	1449.7	3.2
Dice	6	4.8M	OOM	N/A	1.8	2.1	6.5	12	TO	63
Din. Cryptographers	8	190K	18.9	197	0.278	70	0.364	105	356.2	2.4
Din. Philosophers	3	956	0.41	165	1.9	4.8	0.133	98	3.128	0.65
ECS	14	4.8M	OOM	N/A	2.4	23	11.6	79	TO	TO
Fair Exchange	400	320K	14.6	423	2.0	44	2.2	51	TO	1.1
Firewire	11,000	430K	122.2	225	15.1	0.2	19.5	21	232.3	29.5
Leader Election	4	12K	1.8	226	5.0	30	20.4	25	80	0.042
Virus Infection	3	809	0.5	165	2.8	52	0.17	93	0.98	0.032

Figure: Experimental Evaluation

Conclusions

Conclusions

– Approximate answers from value iteration can result into erroneous analysis

Conclusions

- Approximate answers from value iteration can result into erroneous analysis
- Linear programming based exact quantitative model checking does not scale

Conclusions

- Approximate answers from value iteration can result into erroneous analysis
- Linear programming based exact quantitative model checking does not scale
- Proposed algorithm for exact model checking

Conclusions

- Approximate answers from value iteration can result into erroneous analysis
- Linear programming based exact quantitative model checking does not scale
- Proposed algorithm for exact model checking
  - Uses approximate answers from value iteration

Conclusions

- Approximate answers from value iteration can result into erroneous analysis
- Linear programming based exact quantitative model checking does not scale
- Proposed algorithm for exact model checking
  - Uses approximate answers from value iteration
  - Low overhead : based on fast 'binary-search' like technique for rational numbers

Conclusions

- Approximate answers from value iteration can result into erroneous analysis
- Linear programming based exact quantitative model checking does not scale
- Proposed algorithm for exact model checking
  - Uses approximate answers from value iteration
  - Low overhead : based on fast 'binary-search' like technique for rational numbers
- Implemented algorithm in tool RationalSearch (https://publish.illinois.edu/rationalmodelchecker/)

Conclusions

- Approximate answers from value iteration can result into erroneous analysis
- Linear programming based exact quantitative model checking does not scale
- Proposed algorithm for exact model checking
  - Uses approximate answers from value iteration
  - Low overhead : based on fast 'binary-search' like technique for rational numbers
- Implemented algorithm in tool RationalSearch (https://publish.illinois.edu/rationalmodelchecker/)
- Integration in STORM model checker

Thank You !

Power of RationalSearch

– DTMC modeling 11 biased coins (H : $\frac{1}{3}$, T : $\frac{2}{3}$)

Power of RationalSearch

- DTMC modeling 11 biased coins (H : $\frac{1}{3}$, T : $\frac{2}{3}$)
- Probability of all coins landing heads

Power of RationalSearch

- DTMC modeling 11 biased coins (H : $\frac{1}{3}$, T : $\frac{2}{3}$)
- Probability of all coins landing heads
- Correct answer $= 1/177,147$: period of fraction is $20,000$ digits

Power of RationalSearch

- DTMC modeling 11 biased coins (H : $\frac{1}{3}$, T : $\frac{2}{3}$)
- Probability of all coins landing heads
- Correct answer $= 1/177,147$: period of fraction is $20,000$ digits
- PRISM's answer: '0.000005645029269476758'

Power of RationalSearch

- DTMC modeling 11 biased coins (H : $\frac{1}{3}$, T : $\frac{2}{3}$)
- Probability of all coins landing heads
- Correct answer $= 1/177,147$: period of fraction is $20,000$ digits
- PRISM's answer: '0.000005645029269476758'
- RationalSearch estimates the correct answer

Power of RationalSearch

- DTMC modeling 11 biased coins (H : $\frac{1}{3}$, T : $\frac{2}{3}$)
- Probability of all coins landing heads
- Correct answer $= 1/177,147$: period of fraction is $20,000$ digits
- PRISM's answer: '0.000005645029269476758'
- RationalSearch estimates the correct answer
  - With just first 12 digits '0.000005645029'

Power of RationalSearch

- DTMC modeling 11 biased coins (H : $\frac{1}{3}$, T : $\frac{2}{3}$)
- Probability of all coins landing heads
- Correct answer $= 1/177,147$: period of fraction is $20,000$ digits
- PRISM's answer: '0.000005645029269476758'
- RationalSearch estimates the correct answer
  - With just first 12 digits '0.000005645029'