Learning in Verification

Jan Křetínský

Technical University of Munich, Germany

2nd FoPSS Summer School @ FLoC:
Logic and Learning

Oxford
July 4, 2018

Controller synthesis and verification

Controller synthesis and verification

Table of Contents

1. Introduction
2. Strategy computation using reinforcement learning
3. Strategy representation using decision-tree learning
4. Further examples

Formal methods

+ precise
– scalability issues

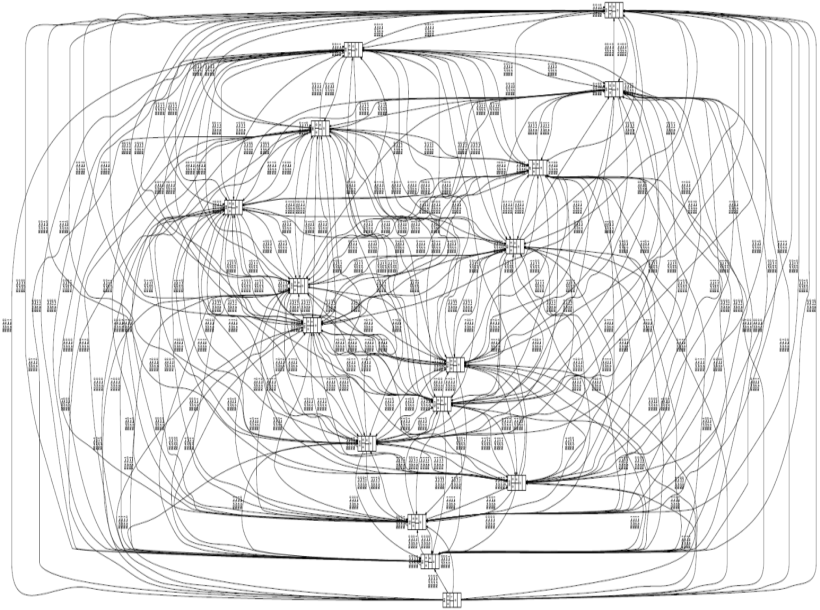
Formal methods

+ precise

– scalability issues

MEM-OUT

Formal methods and machine learning



Formal methods and machine learning

Formal methods

- \+ precise
- – scalability issues
- – can be hard to use

Learning

- – weaker guarantees
- \+ scalable
- \+ simpler solutions

different objectives

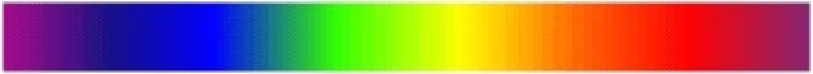
Formal methods and machine learning

Formal methods

- \+ precise
- – scalability issues
- – can be hard to use

Learning

- – weaker guarantees
- \+ scalable
- \+ simpler solutions



Formal methods and machine learning

Formal methods

+ precise
– scalability issues
– can be hard to use

Learning

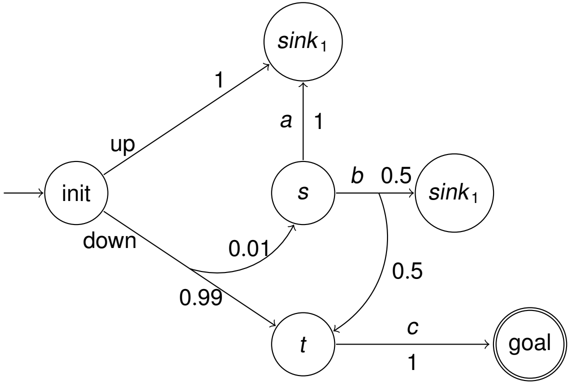
– weaker guarantees
+ scalable
+ simpler solutions

precise computation

focus on important stuff

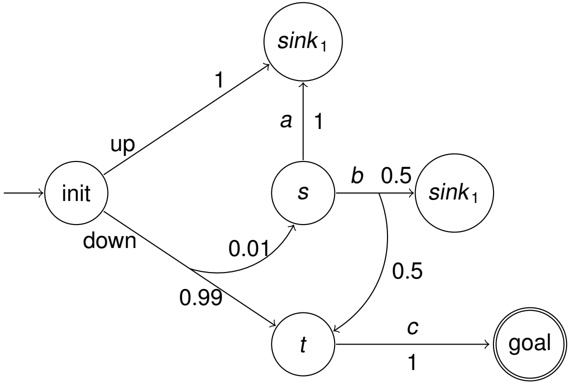
Example: Reachability in Markov decision processes

$$(S,\ s_{\text{init}} \in S,\ A,\ \Delta : S \times A \to \mathcal{D}(S) \cup \{\perp\})$$



Example: Reachability in Markov decision processes

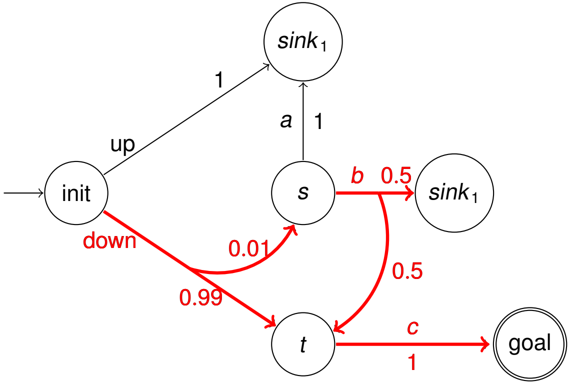
$$(S,\ s_{\text{init}} \in S,\ A,\ \Delta : S \times A \to \mathcal{D}(S) \cup \{\perp\})$$



$$\max_{\text{strategy } \sigma} \mathbb{P}^{\sigma}[\Diamond \text{ goal}]$$

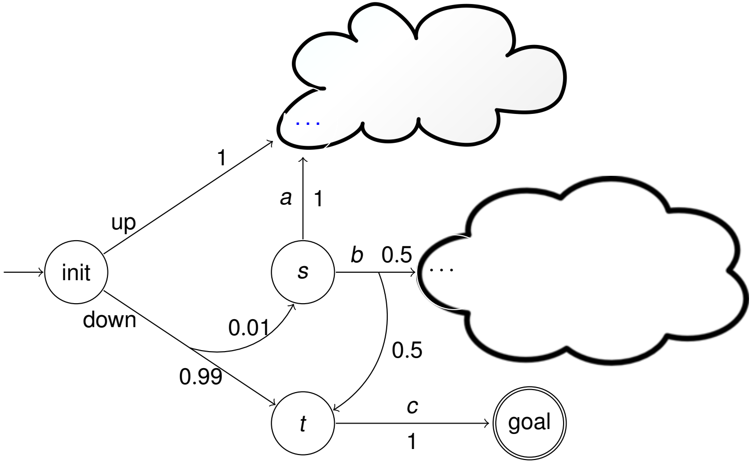
Example: Reachability in Markov decision processes

$$(S,\ s_{\text{init}} \in S,\ A,\ \Delta : S \times A \to \mathcal{D}(S) \cup \{\perp\})$$



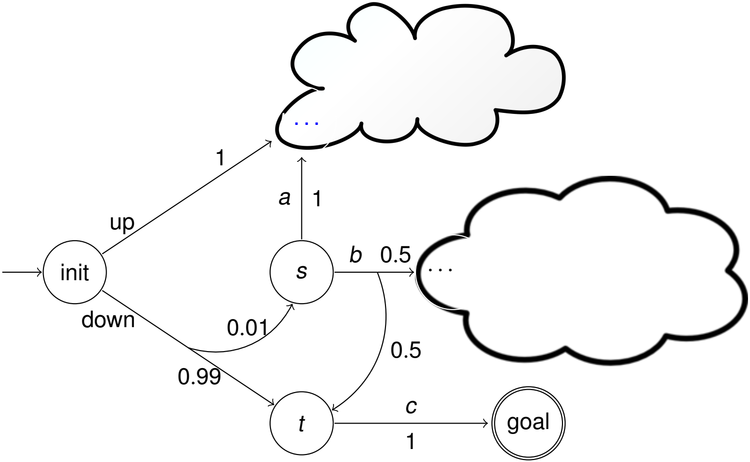
$$\max_{\text{strategy } \sigma} \mathbb{P}^{\sigma}[\Diamond \text{ goal}]$$

Example: Reachability in Markov decision processes



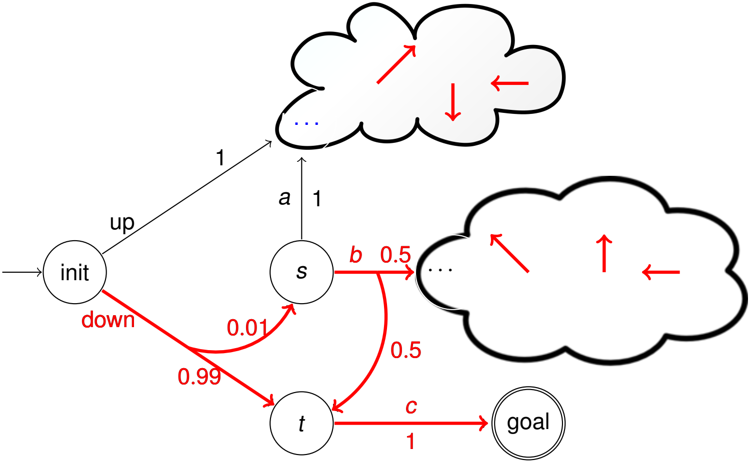
$$\max_{\text{strategy } \sigma} \mathbb{P}^{\sigma}[\Diamond \text{ goal}]$$

Example: Reachability in Markov decision processes



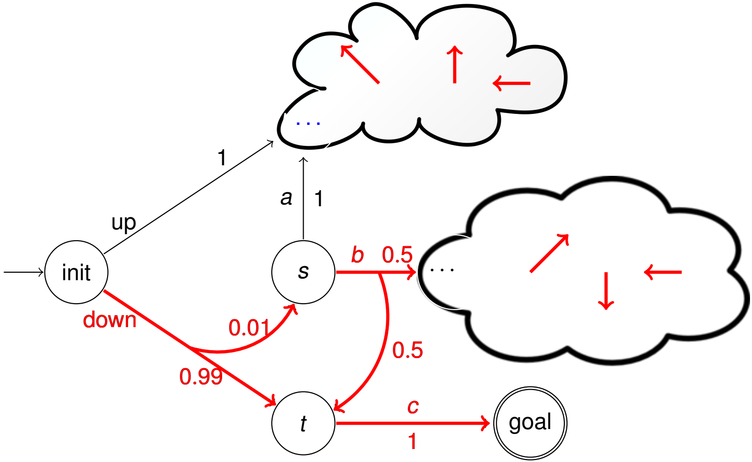
$$\max_{\text{strategy } \sigma} \mathbb{P}^{\sigma}[\Diamond \text{ goal}]$$

Example: Reachability in Markov decision processes



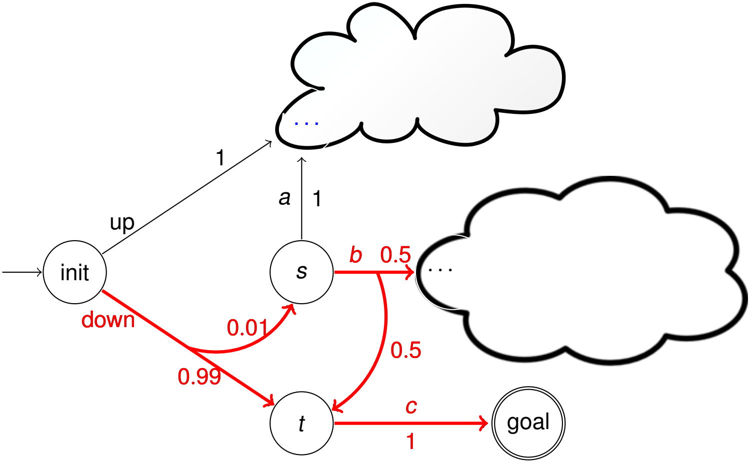
$$\max_{\text{strategy } \sigma} \mathbb{P}^{\sigma}[\Diamond \text{ goal}]$$

Example: Reachability in Markov decision processes



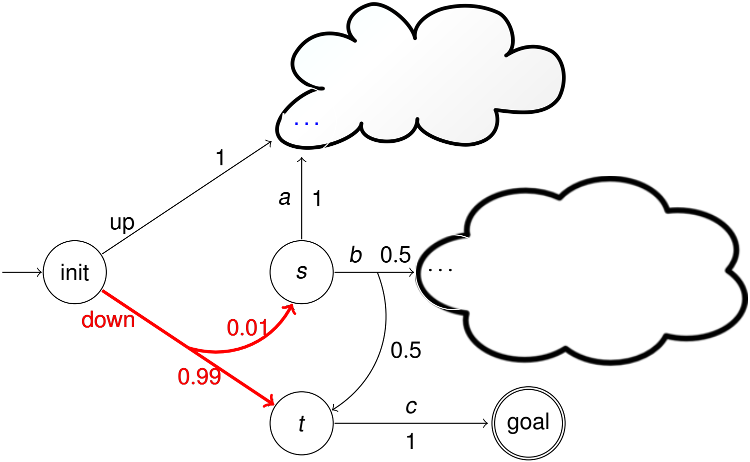
$$\max_{\text{strategy } \sigma} \mathbb{P}^{\sigma}[\Diamond \text{ goal}]$$

Example: Reachability in Markov decision processes



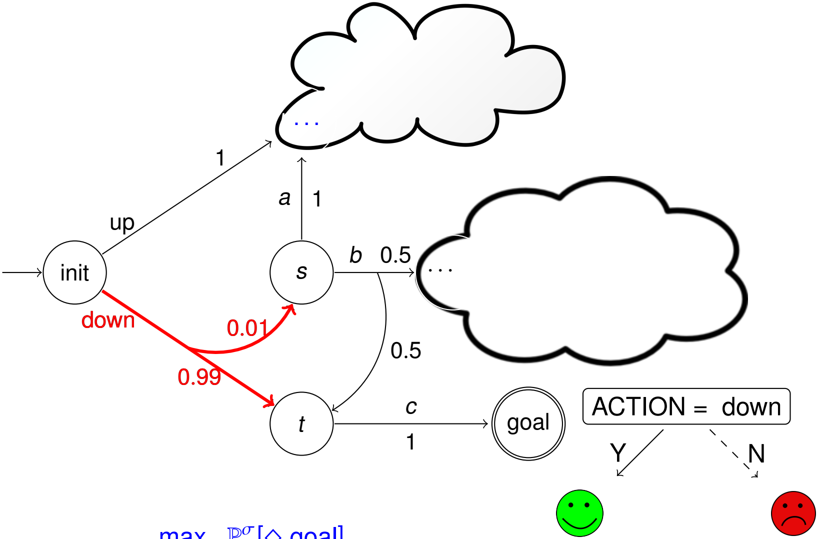
$$\max_{\text{strategy } \sigma} \mathbb{P}^{\sigma}[\Diamond \text{ goal}]$$

Example: Reachability in Markov decision processes



$$\max_{\text{strategy } \sigma} \mathbb{P}^{\sigma}[\Diamond \text{ goal}]$$

Example: Reachability in Markov decision processes



$$\max_{\text{strategy } \sigma} \mathbb{P}^{\sigma}[\Diamond \text{ goal}]$$

Table of Contents

1. Introduction
2. Strategy computation using reinforcement learning
3. Strategy representation using decision-tree learning
4. Further examples

Strategy computation for reachability in MDP

Traditional solution techniques

- Linear programming (LP)
  - precise
  - polynomial time, but practically slow
- Strategy iteration (SI)
  - precise
  - monotonically improving
  - exponential time and **costly evaluation**, but quite ok
- Value iteration (VI)
  - convergent
  - monotonically improving
  - until recently **no general stopping criterion / current error bound**
  - exponential, but fast

Strategy computation for reachability in MDP

Traditional solution techniques

- Linear programming (LP)
  - precise
  - polynomial time, but practically slow
- Strategy iteration (SI)
  - precise
  - monotonically improving
  - exponential time and **costly evaluation**, but quite ok
- Value iteration (VI)
  - convergent
  - monotonically improving
  - until recently **no general stopping criterion** / **current error bound**
  - exponential, but fast

Probably approximately correct techniques

- for (some) black-box systems with (some) objectives
- Statistical model checking (SMC)
- Reinforcement learning (RL)

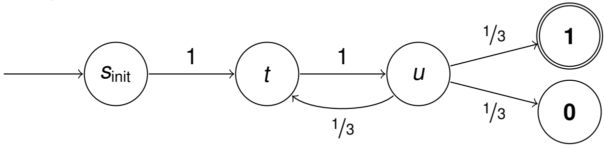
Value iteration for Markov chains

Markov chain

- $|A| \leq 1$, sink **0** and goal **1**

Compute $\mathbb{P}_s[\Diamond \mathbf{1}]$ for each $s \in S$

Example:



Value iteration for Markov chains

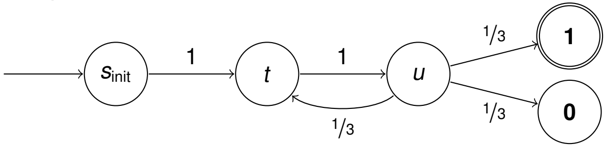
Markov chain

- $|A| \leq 1$, sink **0** and goal **1**

Compute $\mathbb{P}_s[\Diamond \mathbf{1}]$ for each $s \in S$

- iteratively approximate (from below)
- $L : S \to [0, 1]$
- $L(s) := \sum_{s' \in S} \Delta(s, s') \cdot L(s')$

Example:



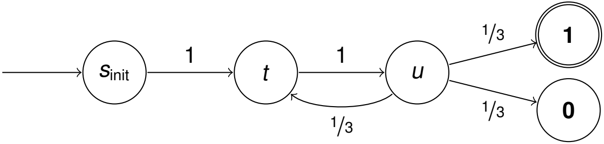
Markov chain

- $|A| \leq 1$, sink **0** and goal **1**

Compute $\mathbb{P}_s[\Diamond \mathbf{1}]$ for each $s \in S$

- iteratively approximate (from below)
- $L : S \to [0, 1]$
- $L(s) := \sum_{s' \in S} \Delta(s, s') \cdot L(s')$

Example:



iteration	s	t	u

Markov chains: VI vs. SMC vs. BRTDP


```
L(·) ← 0
L(1) ← 1
repeat
    for all        transitions s ⟶ do
        UPDATE(s ⟶)
until ?
```

```
procedure UPDATE(s ⟶)
    L(s) := Σ_{s'∈S} Δ(s, s') · L(s')
```

$L(\cdot) \leftarrow 0$

$L(\mathbf{1}) \leftarrow 1$

repeat

for all transitions $s \longrightarrow$ **do**

$\textsc{Update}(s \longrightarrow)$

until ?

procedure $\textsc{Update}(s \longrightarrow)$

$L(s) := \sum_{s' \in S} \Delta(s, s') \cdot L(s')$

Markov chains: VI vs. SMC vs. BRTDP

repeat

    sample a path from s_{init} to $\{\mathbf{1}, \mathbf{0}\}$

until confidence on the statistics is high enough

Markov chains: VI vs. SMC vs. BRTDP

More frequently update what is **visited** more frequently

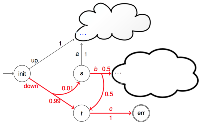
$L(\cdot) \leftarrow 0$
$L(\mathbf{1}) \leftarrow 1$

repeat
    sample a path from s_{init} to $\{\mathbf{1}, \mathbf{0}\}$
    for all visited transitions $s \longrightarrow$ **do**
        $\textsc{Update}(s \longrightarrow)$
until ?

---

procedure $\textsc{Update}(s \longrightarrow)$

    $L(s) := \sum_{s' \in S} \Delta(s, s') \cdot L(s')$



Markov chains: VI vs. SMC vs. BRTDP

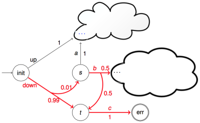
More frequently update what is **visited** more frequently

$L(\cdot) \leftarrow 0$, $U(\cdot) \leftarrow 1$
$L(\mathbf{1}) \leftarrow 1$, $U(\mathbf{0}) \leftarrow 0$

repeat
    sample a path from s_{init} to $\{\mathbf{1}, \mathbf{0}\}$
    for all visited transitions $s \longrightarrow$ **do**
        $\textsc{Update}(s \longrightarrow)$
until $U(s_{\text{init}}) - L(s_{\text{init}}) < \epsilon$

---

procedure $\textsc{Update}(s \longrightarrow)$
    $U(s) := \sum_{s' \in S} \Delta(s, s') \cdot U(s')$
    $L(s) := \sum_{s' \in S} \Delta(s, s') \cdot L(s')$



BRTDP on MDP

$L(\cdot,\cdot) \leftarrow 0, U(\cdot,\cdot) \leftarrow 1$
$U(\mathbf{0},\cdot) \leftarrow 0, L(\mathbf{1},\cdot) \leftarrow 1$

repeat

    for all transitions $s \xrightarrow{a}$ **do**
        $\textsc{Update}(s \xrightarrow{a})$

until $U(s_{\text{init}}) - L(s_{\text{init}}) < \epsilon$

---

procedure $\textsc{Update}(s \xrightarrow{a})$
    $U(s,a) := \sum_{s' \in S} \Delta(s,a,s') \cdot U(s')$
    $L(s,a) := \sum_{s' \in S} \Delta(s,a,s') \cdot L(s')$
    $U(s) := \max_{a \in A} U(s,a)$
    $L(s) := \max_{a \in A} L(s,a)$

BRTDP on MDP

More frequently update what is **visited** more frequently

$L(\cdot,\cdot) \leftarrow 0, U(\cdot,\cdot) \leftarrow 1$
$U(\mathbf{0},\cdot) \leftarrow 0, L(\mathbf{1},\cdot) \leftarrow 1$

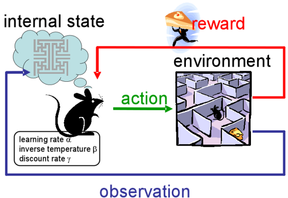
repeat
    sample a path from s_{init} to $\{\mathbf{1},\mathbf{0}\}$
    for all visited transitions $s \xrightarrow{a}$ **do**
        $\textsc{Update}(s \xrightarrow{a})$
until $U(s_{\text{init}}) - L(s_{\text{init}}) < \epsilon$

BRTDP on MDP

More frequently update what is **visited** more frequently
by **reasonably good** strategies

$L(\cdot,\cdot) \leftarrow 0$, $U(\cdot,\cdot) \leftarrow 1$
$U(\mathbf{0},\cdot) \leftarrow 0$, $L(\mathbf{1},\cdot) \leftarrow 1$

repeat
    sample a path from s_{init} to $\{\mathbf{1},\mathbf{0}\}$
    for all visited transitions $s \xrightarrow{a}$ **do**
        $\textsc{Update}(s \xrightarrow{a})$
until $U(s_{\mathrm{init}}) - L(s_{\mathrm{init}}) < \epsilon$



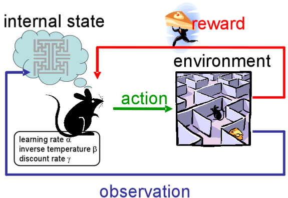
BRTDP on MDP

More frequently update what is **visited** more frequently
by **reasonably good** strategies

$L(\cdot,\cdot) \leftarrow 0, U(\cdot,\cdot) \leftarrow 1$
$U(\mathbf{0},\cdot) \leftarrow 0, L(\mathbf{1},\cdot) \leftarrow 1$

repeat
  sample a path from s_{init} to $\{\mathbf{1},\mathbf{0}\}$ ▷ pick action $\arg\max_a U(s \xrightarrow{a})$
  for all visited transitions $s \xrightarrow{a}$ **do**
    $\textsc{Update}(s \xrightarrow{a})$

until $U(s_{\text{init}}) - L(s_{\text{init}}) < \epsilon$



BRTDP on MDP

More frequently update what is **visited** more frequently
by **reasonably good** strategies

$L(\cdot,\cdot) \leftarrow 0,\ U(\cdot,\cdot) \leftarrow 1$

$U(\mathbf{0},\cdot) \leftarrow 0,\ L(\mathbf{1},\cdot) \leftarrow 1$

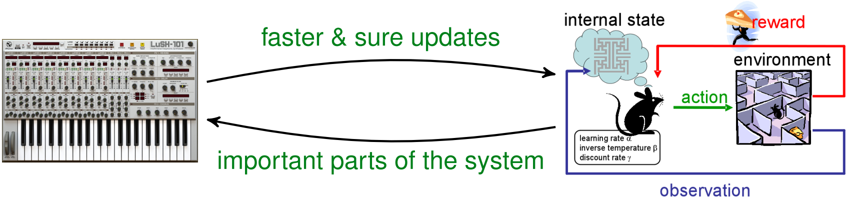
repeat

  sample a path from s_{init} to $\{\mathbf{1},\mathbf{0}\}$ ▹ pick action $\arg\max_a U(s \xrightarrow{a})$

  for all visited transitions $s \xrightarrow{a}$ **do**

    $\textsc{Update}(s \xrightarrow{a})$

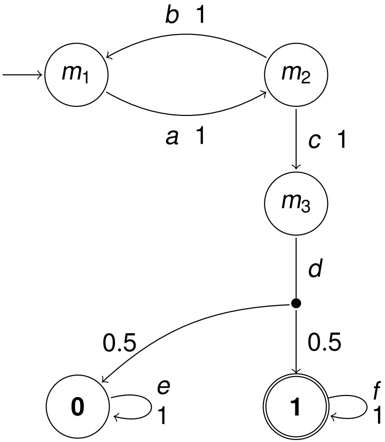
until $U(s_{\mathrm{init}}) - L(s_{\mathrm{init}}) < \epsilon$



Experimental results

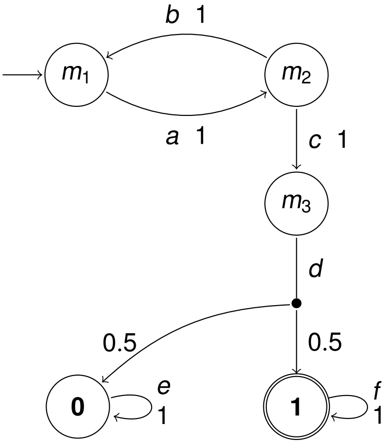
Example	Visited states	
	PRISM	with RL
zeroconf	4,427,159	977
wlan	5,007,548	1,995
firewire	19,213,802	32,214
mer	26,583,064	1,950

Verification: General case with **end components**



Upper bounds:

- identify ECs from (long enough) *simulations*
- contract them *on the fly*



Model not known

- can observe states, not transition probabilities

SMC / PAC RL for MDP

Model not known

- can observe states, not transition probabilities
- cannot use

1: **procedure** UPDATE($s \xrightarrow{a}$)
2: $\quad U(s,a) := \sum_{s' \in S} \Delta(s,a,s') \cdot U(s')$
3: $\quad L(s,a) := \sum_{s' \in S} \Delta(s,a,s') \cdot L(s')$
4: $\quad U(s) := \max_{a \in A} U(s,a)$
5: $\quad L(s) := \max_{a \in A} L(s,a)$

SMC / PAC RL for MDP

Model not known

- can observe states, not transition probabilities
- cannot use

  procedure UPDATE($s \xrightarrow{a}$)
  $\quad U(s,a) := \sum_{s' \in S} \Delta(s,a,s') \cdot U(s')$
  $\quad L(s,a) := \sum_{s' \in S} \Delta(s,a,s') \cdot L(s')$
  $\quad U(s) := \max_{a \in A} U(s,a)$
  $\quad L(s) := \max_{a \in A} L(s,a)$

- instead use experimental average
- $\implies$ probably approximately correct (PAC) RL
- a.k.a. statistical model checking (SMC)

SMC / PAC RL for MDP

Model not known

- can observe states, not transition probabilities
- cannot use

1: **procedure** UPDATE($s \xrightarrow{a}$)
2: $U(s,a) := \sum_{s' \in S} \Delta(s,a,s') \cdot U(s')$
3: $L(s,a) := \sum_{s' \in S} \Delta(s,a,s') \cdot L(s')$
4: $U(s) := \max_{a \in A} U(s,a)$
5: $L(s) := \max_{a \in A} L(s,a)$

- instead use experimental average
- $\implies$ probably approximately correct (PAC) RL
- a.k.a. statistical model checking (SMC)
- for discounted reward due to *Strehl, Li, Wiewiora, Langford, Littman: PAC model-free reinforcement learning. ICML 2006*
- for reachability:
  - not polynomial, but exponential
  - need bounds (also L)
  - U requires the "EC trick"

SMC / PAC RL for MDP

Model not known

- try many runs before concluding the value is significantly lower
- overly safe value changes

Model not known

- try many runs before concluding the value is significantly lower
- overly safe value changes

procedure UPDATE$(s \xrightarrow{a})$

  if $counter(s \xrightarrow{a}) = m$ **then**

  else

    $accum^U(s \xrightarrow{a}) \leftarrow accum^U(s \xrightarrow{a}) + U(s')$

    $counter(s \xrightarrow{a}) \leftarrow counter(s \xrightarrow{a}) + 1$

SMC / PAC RL for MDP

Model not known

- try many runs before concluding the value is significantly lower
- overly safe value changes

procedure $\textsc{Update}(s \xrightarrow{a})$
    if $counter(s \xrightarrow{a}) = m$ **then**
        if $\dfrac{accum^U(s \xrightarrow{a})}{m} < U(s \xrightarrow{a})$ **then**
            $U(s \xrightarrow{a}) \leftarrow \dfrac{accum^U(s \xrightarrow{a})}{m}$
            $accum^U(s \xrightarrow{a}) \leftarrow 0$
            $c(s \xrightarrow{a}) = 0$
    else
        $accum^U(s \xrightarrow{a}) \leftarrow accum^U(s \xrightarrow{a}) + U(s')$
        $counter(s \xrightarrow{a}) \leftarrow counter(s \xrightarrow{a}) + 1$

SMC / PAC RL for MDP

Model not known

- try many runs before concluding the value is significantly lower
- overly safe value changes

procedure $\textsc{Update}(s \xrightarrow{a})$

  if $\mathit{counter}(s \xrightarrow{a}) = m$ **then**

    if $\dfrac{\mathit{accum}^U(s \xrightarrow{a})}{m} < U(s \xrightarrow{a}) \boxed{-\ \xi}$ **then**

      $U(s \xrightarrow{a}) \leftarrow \dfrac{\mathit{accum}^U(s \xrightarrow{a})}{m} \boxed{+\xi}$

      $\mathit{accum}^U(s \xrightarrow{a}) \leftarrow 0$

      $c(s \xrightarrow{a}) = 0$

  else

    $\mathit{accum}^U(s \xrightarrow{a}) \leftarrow \mathit{accum}^U(s \xrightarrow{a}) + U(s')$

    $\mathit{counter}(s \xrightarrow{a}) \leftarrow \mathit{counter}(s \xrightarrow{a}) + 1$

SMC / PAC RL for MDP

Model not known

- try many runs before concluding the value is significantly lower
- overly safe value changes

1: **procedure** $\textsc{Update}(s \xrightarrow{a})$

2: **if** $counter(s \xrightarrow{a}) = m$ **and** $\mathrm{LEARN}(s \xrightarrow{a})$ **then**

3: **if** $\dfrac{accum^U(s \xrightarrow{a})}{m} < U(s \xrightarrow{a}) \boxed{-2\xi}$ **then**

4: $U(s \xrightarrow{a}) \leftarrow \dfrac{accum^U(s \xrightarrow{a})}{m} \boxed{+\xi}$

5: $accum^U(s \xrightarrow{a}) \leftarrow 0$

6: $c(s \xrightarrow{a}) = 0$

7: **else**

8: $accum^U(s \xrightarrow{a}) \leftarrow accum^U(s \xrightarrow{a}) + U(s')$

9: $counter(s \xrightarrow{a}) \leftarrow counter(s \xrightarrow{a}) + 1$

Macro $\mathrm{LEARN}(s \xrightarrow{a})$ is true in the kth call of $\textsc{Update}(s \xrightarrow{a})$ if, since the $(k-2m)$th call of $\textsc{Update}(s \xrightarrow{a})$, line 4 was not executed in any call of $\textsc{Update}(\cdot)$.

Summary: Strategy computation for MDP

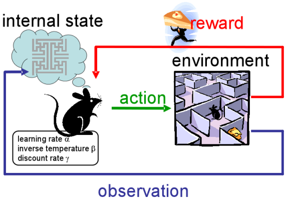
BRTDP (verification) vs. RL (learning)

- reachability vs. (discounted) reward

Approach:

faster & sure updates

important parts of the system

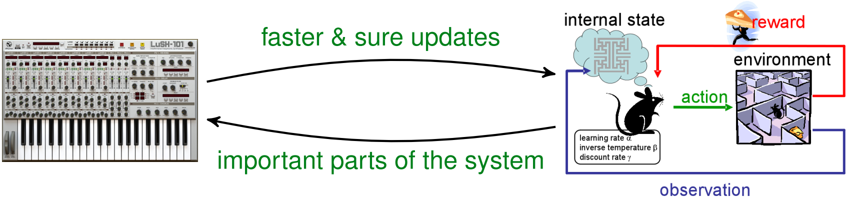


Summary: Strategy computation for MDP

BRTDP (verification) vs. RL (learning)

- reachability vs. (discounted) reward
  - slower feedback

Approach:

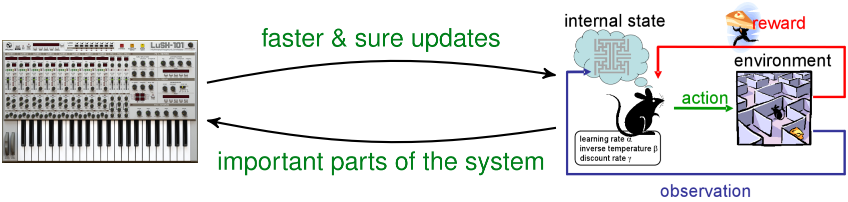


Summary: Strategy computation for MDP

BRTDP (verification) vs. RL (learning)

- reachability vs. (discounted) reward
  - slower feedback
  - incorrect ECs

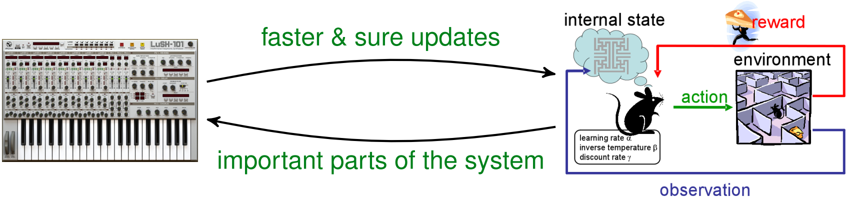
Approach:



BRTDP (verification) vs. RL (learning)

- reachability vs. (discounted) reward
  - slower feedback
  - incorrect ECs
- bounds U, L vs. Q-value

Approach:

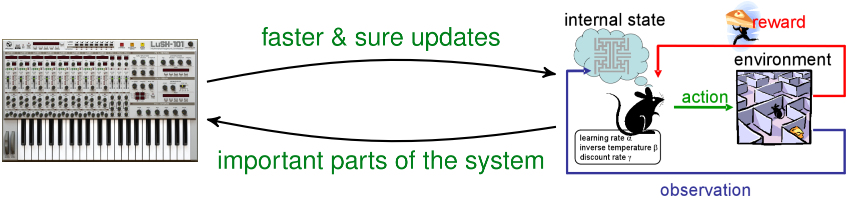


Summary: Strategy computation for MDP

BRTDP (verification) vs. RL (learning)

- reachability vs. (discounted) reward
  - slower feedback
  - incorrect ECs
- bounds U, L vs. Q-value
- exact bounds vs. PAC / no bounds / no convergence

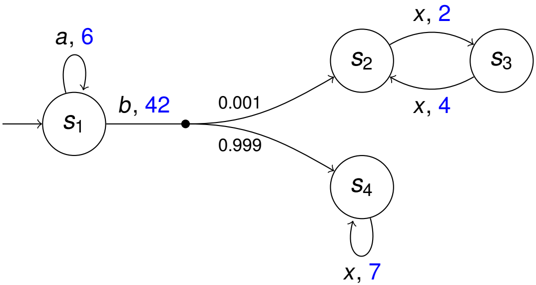
Approach:



Strategy computation for mean payoff in MDP

$MP(\rho_1\ \rho_2\ \rho_3 \cdots) = \liminf_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} \rho_i$

$MP(42\ \ 2\ \ 4\ \ 2\ \ 4\ \ 2\ \ \cdots) = 3$

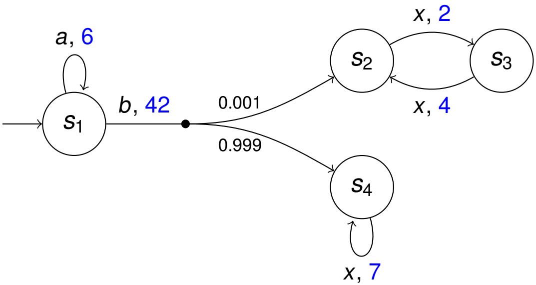


Strategy computation for mean payoff in MDP

$MP(\rho_1\ \rho_2\ \rho_3 \cdots) = \liminf_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} \rho_i$

$MP(42\ \ 2\ \ 4\ \ 2\ \ 4\ \ 2\ \ \cdots) = 3$

$$\max_\sigma \mathbb{E}_\sigma[MP] = \max_\sigma \sum_{MEC\ M} \overbrace{\mathbb{P}_\sigma[\Diamond M]}^{\text{reachability}} \cdot \overbrace{MP(M^\sigma)}^{\text{MP on EC}}$$

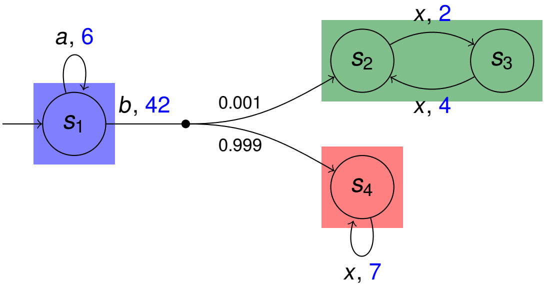


Strategy computation for mean payoff in MDP

$MP(\rho_1\ \rho_2\ \rho_3 \cdots) = \liminf_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} \rho_i$

$MP(42\ \ 2\ \ 4\ \ 2\ \ 4\ \ 2\ \ \cdots) = 3$

$$\max_\sigma \mathbb{E}_\sigma[MP] = \max_\sigma \sum_{MEC\ M} \overbrace{\mathbb{P}_\sigma[\Diamond M]}^{\text{reachability}} \cdot \overbrace{MP(M^\sigma)}^{\text{MP on EC}}$$

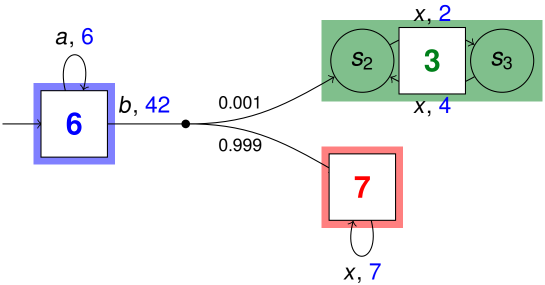


Strategy computation for mean payoff in MDP

$MP(\rho_1\ \rho_2\ \rho_3 \cdots) = \liminf_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} \rho_i$

$MP(42\ \ 2\ \ 4\ \ 2\ \ 4\ \ 2\ \ \cdots) = 3$

$$\max_\sigma \mathbb{E}_\sigma[MP] = \max_\sigma \sum_{MEC\ M} \overbrace{\mathbb{P}_\sigma[\Diamond M]}^{\text{reachability}} \cdot \overbrace{MP(M^\sigma)}^{\text{MP on EC}}$$

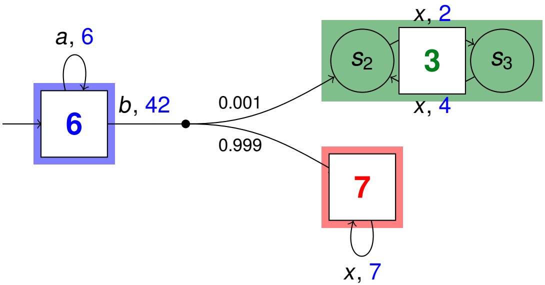


Strategy computation for mean payoff in MDP

$MP(\rho_1\ \rho_2\ \rho_3 \cdots) = \liminf_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} \rho_i$

$MP(42\ \ 2\ \ 4\ \ 2\ \ 4\ \ 2\ \ \cdots) = 3$

$$\max_\sigma \mathbb{E}_\sigma[MP] = \max_\sigma \sum_{MEC\ M} \overbrace{\mathbb{P}_\sigma[\Diamond M]}^{\text{reachability}} \cdot \overbrace{MP(M^\sigma)}^{\text{MP on EC}}$$



Desiderata:

- ignore states with low reachability probability/approx. error/profit
- focus on **highly reachable, uncertain and profitable** states

Strategy computation for mean payoff in MDP

Solution ideas:

1. keep both lower and upper bounds
    - collapse end components (graph trasnformation, on the fly)
    - $\Longrightarrow$ error bound, imprecision
    - $\Longrightarrow$ treat only highly imprecise states

Strategy computation for mean payoff in MDP

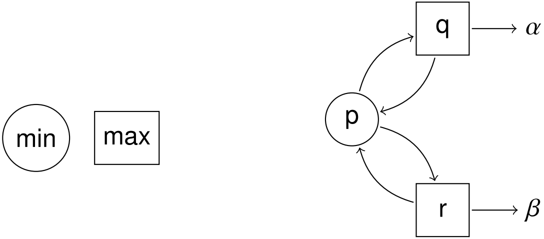
Solution ideas:

1. keep both lower and upper bounds
   - collapse end components (graph trasnformation, on the fly)
   - $\implies$ error bound, imprecision
   - $\implies$ treat only highly imprecise states
2. simulation guided (reinforcement learning)
   - transition probabilities $\implies$ treat only highly reachable states
   - pick currently best actions $\implies$ treat only highly profitable states

Experimental results (MDP with mean payoff)

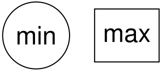
Model (#states, #MECs)	LP	SI	VI	SI*	VI*
cs_nfail3 (184, 38)	2	17	–	4	4
cs_nfail4 (960, 176)	5	1129	–	5	5
sensors1 (462, 132)	3	–	–	4	5
sensors2 (7860, 4001)	101	–	–	13	15
mer3 (15622, 9451)	–	–	–	16	15
mer4 (119305, 71952)	–	–	–	42	64
zeroconf(4730203, ?)	–	–	–	–	10

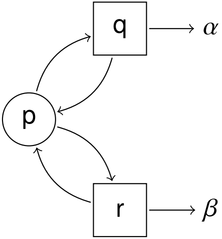
BRTDP needs upper bounds



Strategy computation for simple stochastic games

BRTDP needs upper bounds and intermediate results





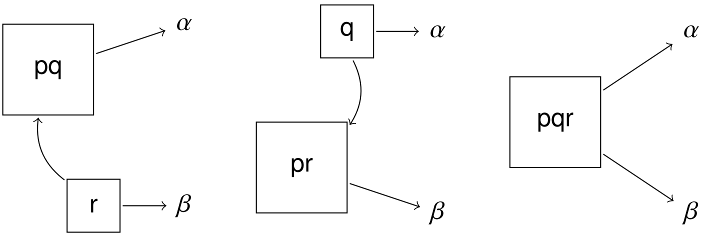


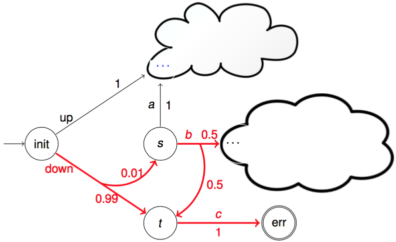
Table of Contents

1. Introduction
2. Strategy computation using reinforcement learning
3. Strategy representation using decision-tree learning
4. Further examples

Small representation of strategies

$\sigma : S \to A$
$\sigma = \{(s, \sigma(s)) \mid s \in S\}$

How to make it more readable?

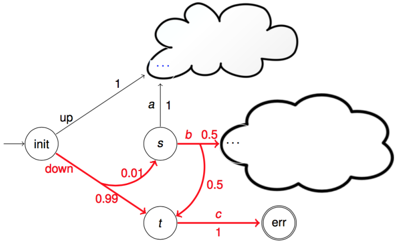


Small representation of strategies

$\sigma : S \to A$
$\sigma = \{(s, \sigma(s)) \mid s \in S\}$

How to make it more readable?

- Encoding?
- Smaller?

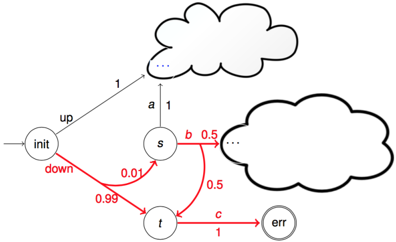


Small representation of strategies

$\sigma : S \to A$
$\sigma = \{(s, \sigma(s)) \mid s \in S\}$

How to make it more readable?

- Encoding?
- Smaller?



- Cut off states with zero importance (unreachable or useless)

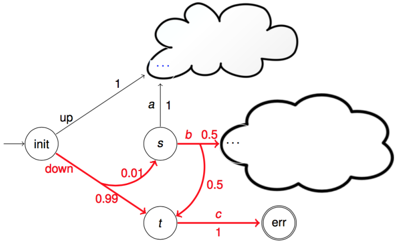
Small representation of strategies

$\sigma : S \to A$

$\sigma = \big\{(s, \sigma(s)) \mid s \in S\big\}$

How to make it more readable?

- Encoding?
- Smaller?



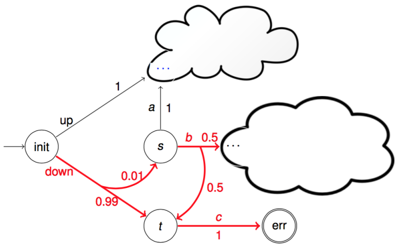
- Cut off states with zero importance (unreachable or useless)
- Cut off states with low importance (small error, ε-optimal strategy)

Small representation of strategies

$\sigma : S \to A$
$\sigma = \{(s, \sigma(s)) \mid s \in S\}$

How to make it more readable?

- Encoding?
- Smaller?



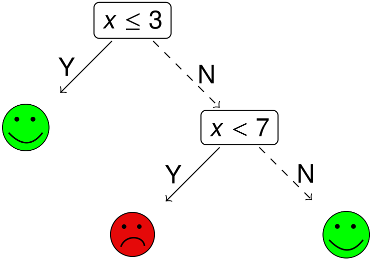
- Cut off states with zero importance (unreachable or useless)
- Cut off states with low importance (small error, ε-optimal strategy)
- How to make use of the exact quantities?

Supervised learning is the machine learning task of learning a function $f : X \rightarrow Y$ that maps an input to an output based on example input-output pairs $\{(\vec{x}_1, \vec{y}_1), \ldots, (\vec{x}_n, \vec{y}_n)\}$.

Decision trees

Supervised learning is the machine learning task of learning a function $f : X \to Y$ that maps an input to an output based on example input-output pairs $\{(\vec{x}_1, \vec{y}_1), \dots, (\vec{x}_n, \vec{y}_n)\}$.

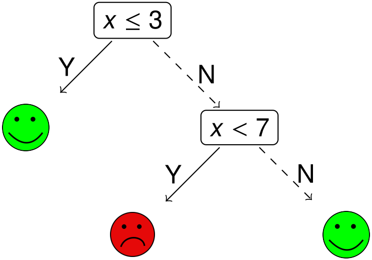
Example: A decision tree for $\{1, 2, 3, 7\} \subseteq \{1, \dots, 7\}$



Decision trees

Supervised learning is the machine learning task of learning a function $f : X \to Y$ that maps an input to an output based on example input-output pairs $\{(\vec{x}_1, \vec{y}_1), \ldots, (\vec{x}_n, \vec{y}_n)\}$.

Example: A decision tree for $\{1, 2, 3, 7\} \subseteq \{1, \ldots, 7\}$



Positive examples:
Good $= \{1, 3, 7\}$

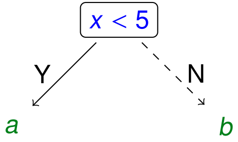
Negative examples:
Bad $= \{6\}$

Decision trees for MDP strategy representation

Assumption: a state of S is given by a valuation of integer variables

Task: Encode (memoryless deterministic) strategy as DT

$\sigma : S \to A$



Decision trees for MDP strategy representation

Assumption: a state of S is given by a valuation of integer variables

Task: Encode (memoryless deterministic) strategy as DT

$\sigma : S \to A$

$\sigma : S \times A \to 2$ (permissive vs. liberal)

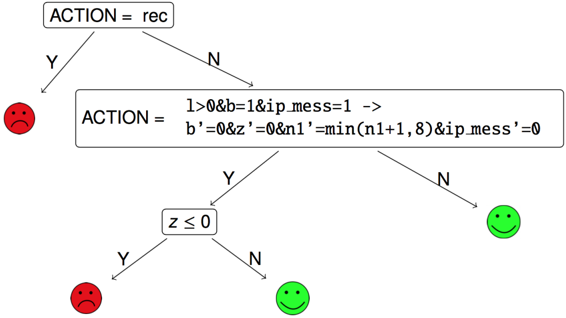
Decision trees for MDP strategy representation

Assumption: a state of S is given by a valuation of integer variables

Task: Encode (memoryless deterministic) strategy as DT

$\sigma : S \to A$

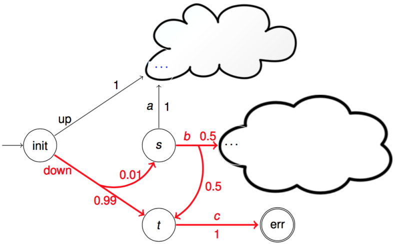
$\sigma : S \times A \to 2$ (permissive vs. liberal)



Decision trees for MDP strategy representation

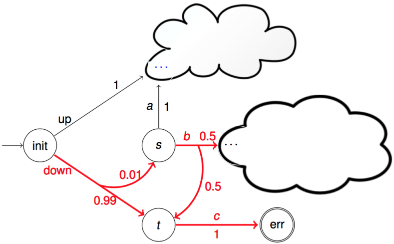
Algorithm

1. generate *Good* $\subseteq \{(s, a) \mid \sigma(s, a)\}$
   and *Bad* $\subseteq \{(s, a) \mid a \in A(s), \neg\sigma(s, a)\}$
2. learn a DT τ for *Good*, *Bad*
3. evaluate strategy τ
4. if good enough then terminate
   else goto 1

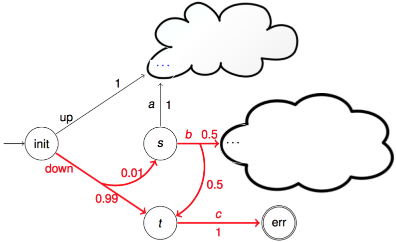


Algorithm

1. generate *Good* $\subseteq \{(s,a) \mid \sigma(s,a)\}$
   and *Bad* $\subseteq \{(s,a) \mid a \in A(s), \neg\sigma(s,a)\}$
2. learn a DT τ for *Good*, *Bad*
3. evaluate strategy τ
4. if good enough then terminate
   else goto 1

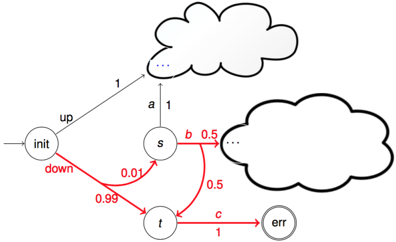


Data for decision tree



What to put in *Good* and *Bad* (and how many times)?

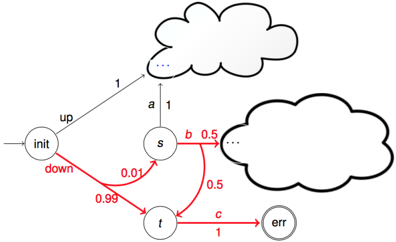
Data for decision tree



What to put in *Good* and *Bad* (and how many times)?

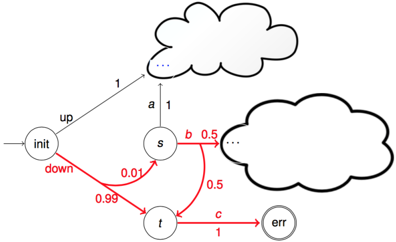
More important decision $\Longrightarrow$ more frequent data

Data for decision tree



What to put in *Good* and *Bad* (and how many times)?
More important decision $\Longrightarrow$ more frequent data

Importance of a decision in *s* with respect to $\Diamond$*goal* and controller σ:

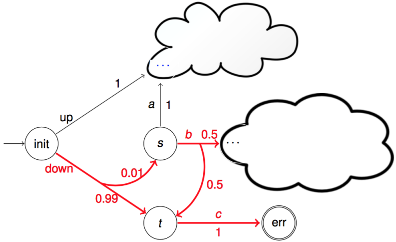


What to put in *Good* and *Bad* (and how many times)?
More important decision $\implies$ more frequent data

Importance of a decision in s with respect to $\Diamond goal$ and controller σ:

$$\mathbb{P}^{\sigma}[\Diamond s \qquad\qquad]$$

Data for decision tree

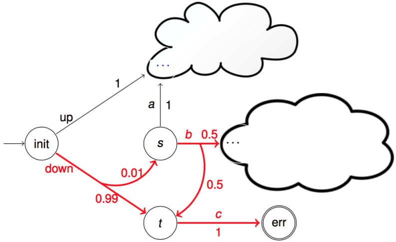


What to put in *Good* and *Bad* (and how many times)?
More important decision $\Longrightarrow$ more frequent data

Importance of a decision in s with respect to $\Diamond goal$ and controller σ:

$$\mathbb{P}^{\sigma}[\Diamond s \mid \Diamond goal]$$

Data for decision tree



What to put in *Good* and *Bad* (and how many times)?
More important decision $\implies$ more frequent data

Importance of a decision in s with respect to $\Diamond goal$ and controller σ:

$$\mathbb{P}^{\sigma}[\Diamond s \mid \Diamond goal] \approx \frac{\#\text{simulations visiting } s, goal}{\#\text{simulations visiting } goal}$$

$\implies$ take states on successful simulations

Experimental results (MDP strategy representation)

Example	#states	Value	Explicit	BDD	DT	Rel.err(DT) %
firewire	481,136	1.0	479,834	4233	1	0.0
investor	35,893	0.958	28,151	783	27	0.886
mer	1,773,664	0.200016	———	MEM-OUT	———	*
zeroconf	89,586	0.00863	60,463	409	7	0.106

Experimental results (MDP strategy representation)

Example	#states	Value	Explicit	BDD	DT	Rel.err(DT) %
firewire	481,136	1.0	479,834	4233	1	0.0
investor	35,893	0.958	28,151	783	27	0.886
mer	1,773,664	0.200016	———	MEM-OUT	———	*
zeroconf	89,586	0.00863	60,463	409	7	0.106

* MEM-OUT in PRISM,
whereas RL yields: 1887 619 13 0.00014

Decision trees vs. BDD

Diasadvantage: no subgraph merging (BDD are dags)

Advantage: can choose different predicates on the same level (BDD have fixed variable ordering)

Decision trees vs. BDD

Diasadvantage: no subgraph merging (BDD are dags)

Advantage: can choose different predicates on the same level (BDD have fixed variable ordering)

Learning advantages:

- wider class of predicates (not just bit representation)

Decision trees vs. BDD

Diasadvantage: no subgraph merging (BDD are dags)

Advantage: can choose different predicates on the same level (BDD have fixed variable ordering)

Learning advantages:

- wider class of predicates (not just bit representation)
- entropy-based heuristic (vs. variable ordering)

Decision trees vs. BDD

Diasadvantage: no subgraph merging (BDD are dags)

Advantage: can choose different predicates on the same level (BDD have fixed variable ordering)

Learning advantages:

- wider class of predicates (not just bit representation)
- entropy-based heuristic (vs. variable ordering)
- don’t-care inputs ($\overline{\mathit{Good} \cup \mathit{Bad}}$ can be resolved either way)

Decision trees vs. BDD

Diasadvantage: no subgraph merging (BDD are dags)

Advantage: can choose different predicates on the same level (BDD have fixed variable ordering)

Learning advantages:

- wider class of predicates (not just bit representation)
- entropy-based heuristic (vs. variable ordering)
- don't-care inputs ($\overline{\mathit{Good} \cup \mathit{Bad}}$ can be resolved either way)
- imprecise outputs (not exactly $\mathit{Good} \mapsto$ ☺, $\mathit{Bad} \mapsto$ ☹)

Decision trees for games strategy representation

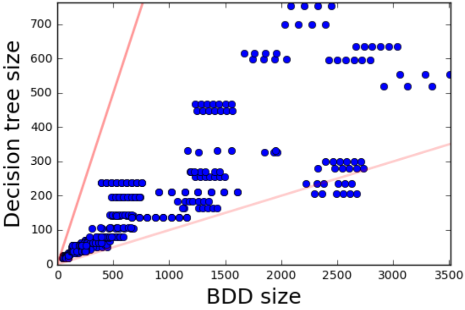
Non-deterministic adversary $\Longrightarrow$ capture almost all decisions

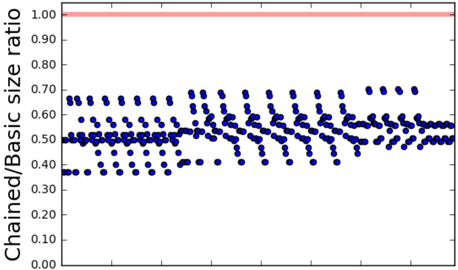
- overfitting $\Longrightarrow$ unfold until leaves decided
- unfold even under no information gain $\Longrightarrow$ look-ahead

Additional issue for synthesis for I/O signals: only Boolean structure

Experimental results (game strategy representation)

Safety

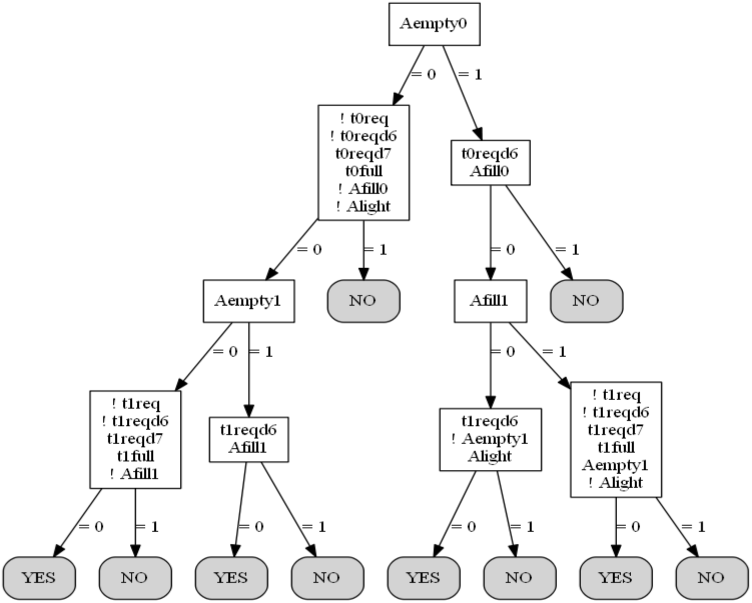




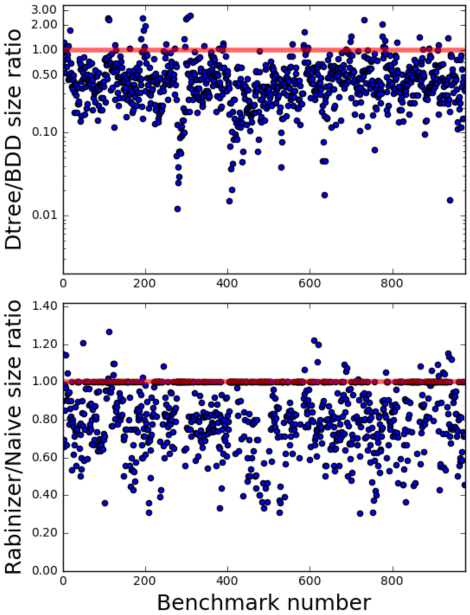
Experimental results (reachability)

Name	$\|S\|$	$\|I\|$	$\|O\|$	$\|Train\|$	$\|BDD\|$	$\|DT\|$	$\|DT^+\|$
wash_3_1_1_3	102	3	7	40	45	3	1
wash_4_1_1_3	466	4	9	144	76	4	1
wash_4_1_1_4	346	4	9	96	78	4	1
wash_4_2_1_4	958	4	9	432	157	4	1
wash_4_2_2_4	3310	4	9	432	301	4	1
wash_5_1_1_3	1862	5	11	416	127	5	1
wash_5_1_1_4	1630	5	11	352	121	5	1
wash_5_2_1_4	5365	5	11	2368	255	5	1
wash_5_2_2_4	27919	5	11	2368	554	5	1
wash_6_1_1_3	6962	6	13	1088	193	6	1
wash_6_1_1_4	6622	6	13	1024	172	6	1
wash_6_2_1_4	27412	6	13	10432	419	6	1

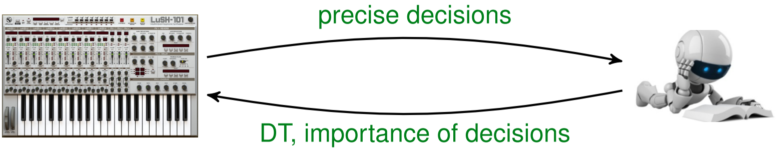
Experimental results (parametric solutions)

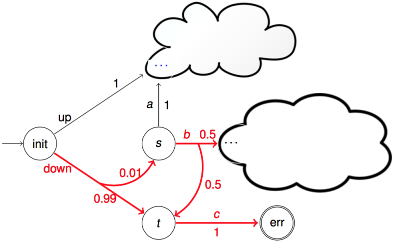


Experimental results (LTL synthesis)



Summary: Strategy representation





Cut off states with zero importance (unreachable or useless)

Cut off states with low importance (if possible)

Making use of the exact quantities

Importance of a decision in s with respect to $\Diamond goal$ and strategy σ:

e.g. $\mathbb{P}^{\sigma}[\Diamond s \mid \Diamond goal]$ or “losing action”

Literature

- Reinforcement learning for efficient strategy synthesis
  - MDP with functional spec (reachability, LTL)[1] [2]
  - MDP with performance spec (mean payoff/average reward)[3] [4]
  - Simple stochastic games (reachability)[5]
- Decision tree learning for efficient strategy representation
  - MDP[6]
  - Games[7]

---

[1] Brazdil, Chatterjee, Chmelik, Forejt, K., Kwiatkowska, Parker, Ujma: Verification of Markov Decision Processes Using Learning Algorithms. ATVA 2014

[2] Daca, Henzinger, K., Petrov: Faster Statistical Model Checking for Unbounded Temporal Properties. TACAS 2016

[3] Ashok, Chatterjee, Daca, K., Meggendorfer: Value Iteration for Long-run Average Reward in Markov Decision Processes. CAV 2017

[4] K., Meggendorfer: Efficient Strategy Iteration for Mean Payoff in Markov Decision Processes. ATVA 2017

[5] Kelmedi, Krämer, K., Weininger: Value Iteration for Simple Stochastic Games: Stopping Criterion and Learning Algorithm. CAV 2018

[6] Brazdil, Chatterjee, Chmelik, Fellner, K.: Counterexample Explanation by Learning Small Strategies in Markov Decision Processes. CAV 2015

[7] Brazdil, Chatterjee, K., Toman: Strategy Representation by Decision Trees in Reactive Synthesis. TACAS 2018

Table of Contents

1. Introduction
2. Strategy computation using reinforcement learning
3. Strategy representation using decision-tree learning
4. Further examples

Further examples

1. Reinforcement learning in verification
2. Decision-tree learning
3. Automata learning
4. Other domains, meta-domains

Reinforcement learning in verification

David, Jensen, Larsen, Legay, Lime, Sorensen, Taankvist: On Time with Minimal Expected Cost! ATVA 2014

- priced timed MDP: worst case time-bounds + minimal expected cost
- 1. $\sigma \leftarrow$ uniform strategy
  2. simulate σ
  3. $\sigma \leftarrow$ learn a better strategy from the best runs
     (covarinace / logisitic regression / trees)
  4. go to 2. or output the best currently known (safe) strategy

Reinforcement learning in verification

David, Jensen, Larsen, Legay, Lime, Sorensen, Taankvist: On Time with Minimal Expected Cost! ATVA 2014

- priced timed MDP: worst case time-bounds + minimal expected cost
- 1. $\sigma \leftarrow$ uniform strategy
  2. simulate σ
  3. $\sigma \leftarrow$ learn a better strategy from the best runs
     (covarinace / logisitic regression / trees)
  4. go to 2. or output the best currently known (safe) strategy

Junges, Jansen, Dehnert, Topcu, Katoen: Safety-Constrained Reinforcement Learning for MDPs. TACAS 2016

- 1. compute safe actions
  2. there run RL

Reinforcement learning in verification

David, Jensen, Larsen, Legay, Lime, Sorensen, Taankvist: On Time with Minimal Expected Cost! ATVA 2014

- priced timed MDP: worst case time-bounds + minimal expected cost
- 1. $\sigma \leftarrow$ uniform strategy
  2. simulate σ
  3. $\sigma \leftarrow$ learn a better strategy from the best runs
     (covarinace / logisitic regression / trees)
  4. go to 2. or output the best currently known (safe) strategy

Junges, Jansen, Dehnert, Topcu, Katoen: Safety-Constrained Reinforcement Learning for MDPs. TACAS 2016

- 1. compute safe actions
  2. there run RL

Hasanbeig, Abate, Kroening: Logically-Correct Reinforcement Learning.

- like BRTDP, but with limit-deterministic Büchi automaton

Reinforcement learning in verification

David, Jensen, Larsen, Legay, Lime, Sorensen, Taankvist: On Time with Minimal Expected Cost! ATVA 2014

- priced timed MDP: worst case time-bounds + minimal expected cost
- 1. $\sigma \leftarrow$ uniform strategy
  2. simulate σ
  3. $\sigma \leftarrow$ learn a better strategy from the best runs
     (covarinace / logisitic regression / trees)
  4. go to 2. or output the best currently known (safe) strategy

Junges, Jansen, Dehnert, Topcu, Katoen: Safety-Constrained Reinforcement Learning for MDPs. TACAS 2016

- 1. compute safe actions
  2. there run RL

Hasanbeig, Abate, Kroening: Logically-Correct Reinforcement Learning.

- like BRTDP, but with limit-deterministic Büchi automaton

K., Pérez, Raskin: Learning-Based Mean-Payoff Optimization in an Unknown MDP under Omega-Regular Constraints. CONCUR 2018

- RL for long-run average reward, while satisfying a parity condition

David, Jensen, Larsen, Legay, Lime, Sorensen, Taankvist: On Time with Minimal Expected Cost! ATVA 2014

- priced timed MDP: worst case time-bounds + minimal expected cost
- 1. $\sigma \leftarrow$ uniform strategy
  2. simulate σ
  3. $\sigma \leftarrow$ learn a better strategy from the best runs
     (covarinace / logisitic regression / trees)
  4. go to 2. or output the best currently known (safe) strategy

Junges, Jansen, Dehnert, Topcu, Katoen: Safety-Constrained Reinforcement Learning for MDPs. TACAS 2016

- 1. compute safe actions
  2. there run RL

Hasanbeig, Abate, Kroening: Logically-Correct Reinforcement Learning.

- like BRTDP, but with limit-deterministic Büchi automaton

K., Pérez, Raskin: Learning-Based Mean-Payoff Optimization in an Unknown MDP under Omega-Regular Constraints. CONCUR 2018

- RL for long-run average reward, while satisfying a parity condition

Ashok, Brázdil, K., Slámečka: Monte Carlo Tree Search for Verifying Reachability in Markov Decision Processes.

Decision-tree learning

Invariant generation

- 1. from sample runs learn candidates for invariants
  2. check candidates
  3. refine incorrect candidates / return a correct one

Krishna, Puhrsch, Wies: Learning invariants using decision trees. 2015

Garg, Neider, Madhusudan, Roth: Learning invariants using decision trees and implication counterexamples. POPL 2016

Automata learning

Neider, Topcu: An Automaton Learning Approach to Solving Safety Games over Infinite Graphs. TACAS 2016

- strategy representation

Learn a model of a system and check the learnt model

- *Fitrau-Brostean, Janssen, Vaandrager: Combining model learning and model checking to analyze TCP implementations. CAV 2016*
- *Santolucito, Zhai, Piskac: Probabilistic automated language learning for configuration files. CAV 2016*
- *Chen, Hsieh, Lengál, Lii, Tsai, Wang, and Wang: PAC learning-based verification and model synthesis. ICSE 2016*

Anything will do…

Guidance of theorem provers:

- *Kaliszyk, Mamane,Urban: Machine learning of Coq proof guidance: First experiments. SCSS 2014*
- *Blanchette, Greenaway, Kaliszyk, Kühlwein, Urban: A learning-based fact selector for Isabelle/HOL. J. Autom. Reasoning 2016*

Anything will do…

Guidance of theorem provers:

- *Kaliszyk, Mamane,Urban: Machine learning of Coq proof guidance: First experiments. SCSS 2014*
- *Blanchette, Greenaway, Kaliszyk, Kühlwein, Urban: A learning-based fact selector for Isabelle/HOL. J. Autom. Reasoning 2016*

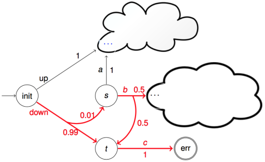
Meta-usage: Choice of model checker

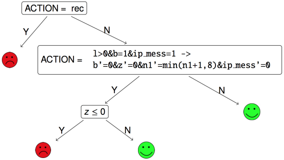
- *Demyanova, Pani, Veith, Zuleger: Empirical software metrics for benchmarking of verification tools. CAV 2015*
- *Czech, Hüllermeier, Jakobs, Wehrheim. Predicting rankings of software verification tools. FSE 2017*

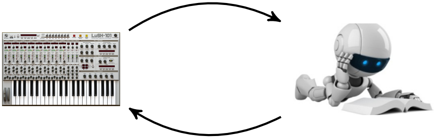
Summary

Machine learning in verification

- Heuristics to improve usability, e.g., **scalability and explainability**
- Example 1: **Speeding up** value iteration
  - TECHNIQUE: reinforcement learning
  - IDEA: focus on updating "most important parts" = most often visited by good strategies
- Example 2: **Small and readable strategies**
  - TECHNIQUE: decision tree learning
  - IDEA: based on the importance of states, feed the decisions to the learning algorithm

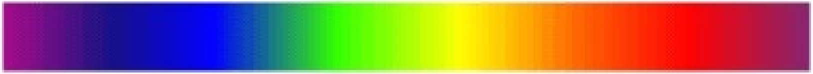






Discussion

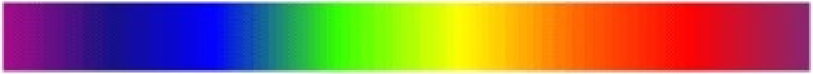
Verification using machine learning



- How far do we want to compromise?
- Do we have to compromise?
  - BRTDP, invariant generation, strategy representation don't
- Don't we want more than ML?
  - (ε-)optimal controllers?
  - arbitrary controllers – is it still verification?
- What do we actually want?
  - scalability shouldn't overrule guarantees?
  - oracle usage seems fine
  - when is PAC enough?

Discussion

Verification using machine learning

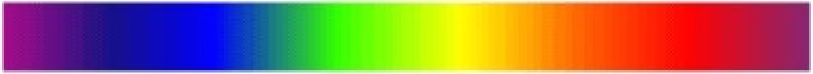


- How far do we want to compromise?
- Do we have to compromise?
  - BRTDP, invariant generation, strategy representation don't
- Don't we want more than ML?
  - (ε-)optimal controllers?
  - arbitrary controllers – is it still verification?
- What do we actually want?
  - scalability shouldn't overrule guarantees?
  - oracle usage seems fine
  - when is PAC enough?

- 3rd Workshop on Learning in Verification (LiVe) @ ETAPS (April 2019)

Discussion

Verification using machine learning



- How far do we want to compromise?
- Do we have to compromise?
  - BRTDP, invariant generation, strategy representation don't
- Don't we want more than ML?
  - (ε-)optimal controllers?
  - arbitrary controllers – is it still verification?
- What do we actually want?
  - scalability shouldn't overrule guarantees?
  - oracle usage seems fine
  - when is PAC enough?

- 3rd Workshop on Learning in Verification (LiVe) @ ETAPS (April 2019)

Thank you