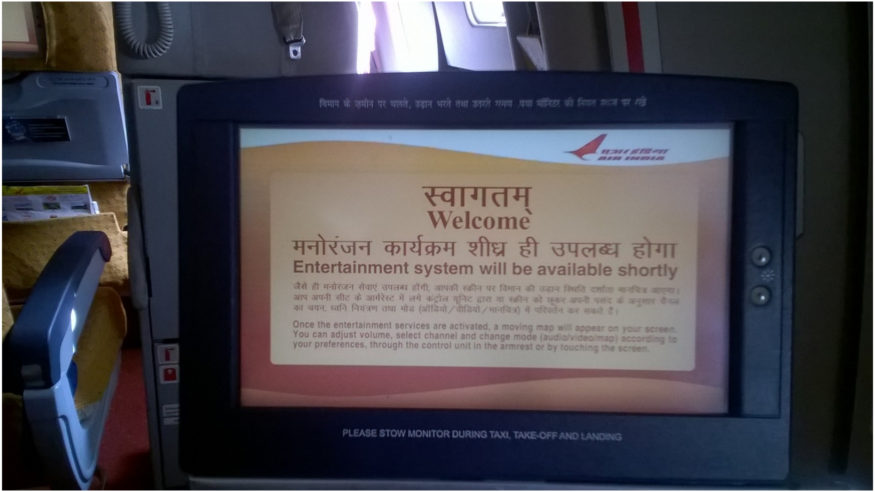
Learning to Use Learning in Verification

Jan Křetínský

Technische Universität München, Germany

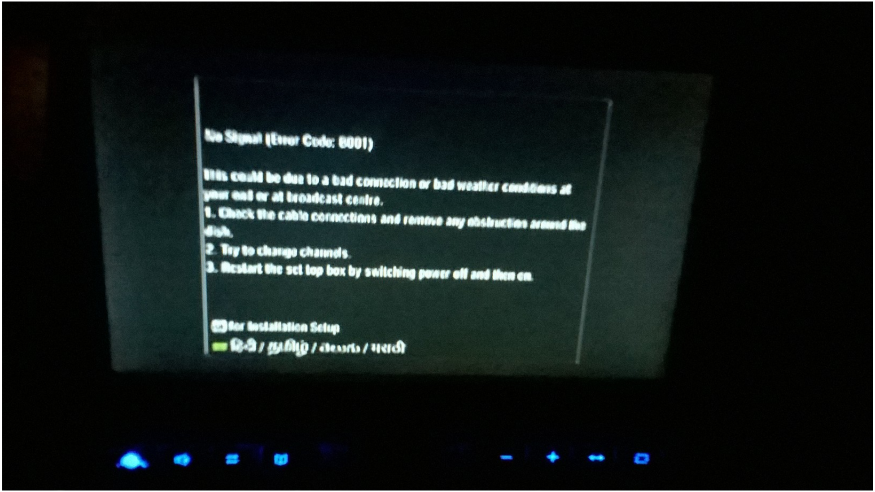
joint work with T. Brázdil (Masaryk University Brno),
K. Chatterjee, M. Chmelík, P. Daca, A. Fellner, T. Henzinger, T. Petrov (IST Austria),
V. Forejt, M. Kwiatkowska, M. Ujma (Oxford University)
D. Parker (University of Birmingham)
published at ATVA 2014, CAV 2015, TACAS 2016

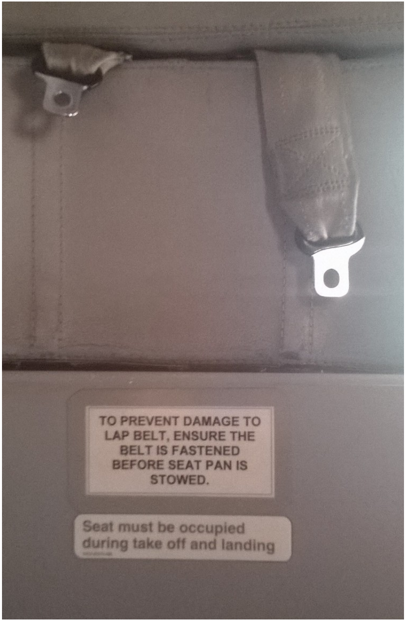
Mysore Park Workshop
Trends and Challenges in Quantitative Verification
February 3, 2016


स्वागतम्
Welcome
मनोरंजन कार्यक्रम शीघ्र ही उपलब्ध होगा
Entertainment system will be available shortly
Once the entertainment services are activated, a moving map will appear on your screen. You can adjust volume, select channel and change mode (audio/video/map) according to your preferences, through the control unit in the armrest or by touching the screen.
PLEASE STOW MONITOR DURING TAXI, TAKE-OFF AND LANDING


सेवा उपलब्ध नहीं है
Service not available
RICOH


सेवा उपलब्ध नहीं है
Service not available


No Signal (Error Code: 8001)
This could be due to a bad connection or bad weather conditions at your end or at broadcast centre.
1. Check the cable connections and remove any obstruction around the dish.
2. Try to change channels.
3. Restart the set top box by switching power off and then on.
for Installation Setup


TO PREVENT DAMAGE TO LAP BELT, ENSURE THE BELT IS FASTENED BEFORE SEAT PAN IS STOWED.
Seat must be occupied during take off and landing

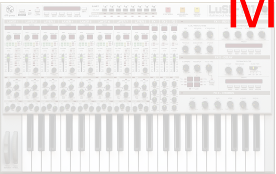
Approaches and their interaction

Formal methods

- precise
- scalability issues

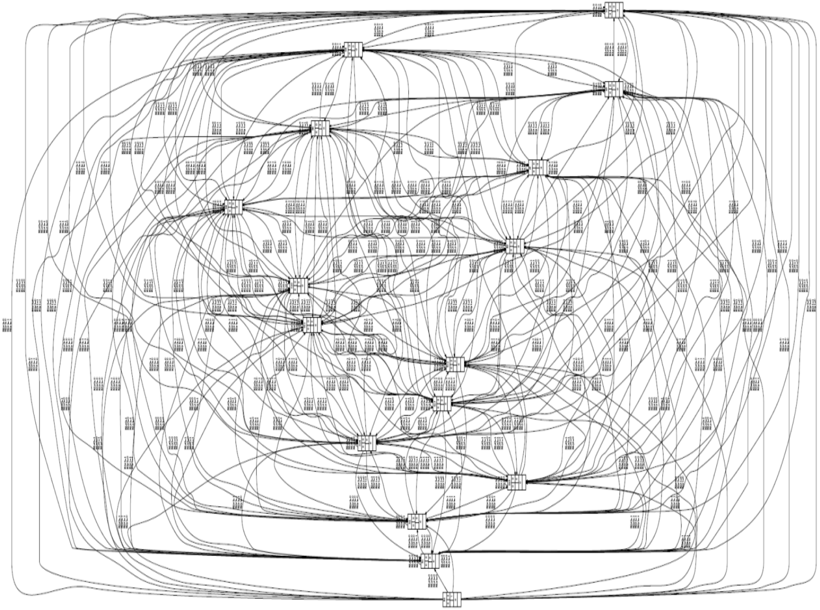
Formal methods

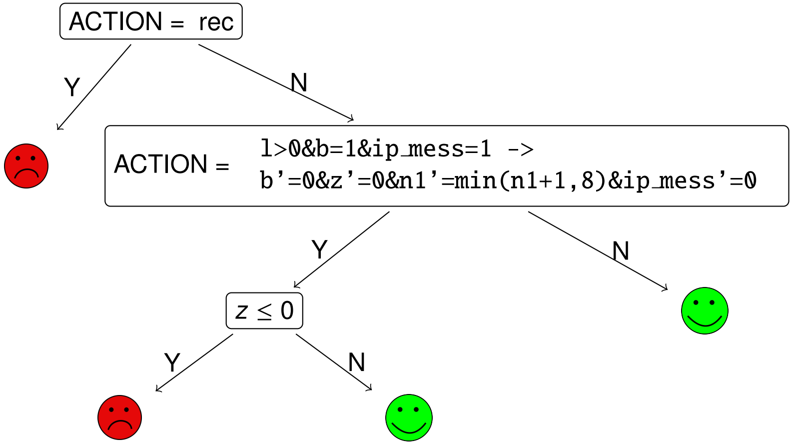
- precise
- scalability issues



MEM-OUT

Approaches and their interaction





ACTION = rec
Y
N
ACTION = l>0&b=1&ip_mess=1 -> b'=0&z'=0&n1'=min(n1+1,8)&ip_mess'=0
Y
N
z ≤ 0
Y
N

Approaches and their interaction

Formal methods

- precise
- scalability issues

Approaches and their interaction

Formal methods

- precise
- scalability issues

Learning

- weaker guarantees
- scalable

different objectives

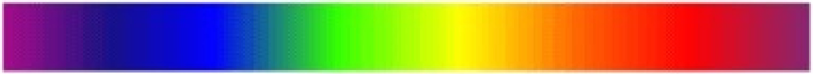
Approaches and their interaction

Formal methods

- precise
- scalability issues

Learning

- weaker guarantees
- scalable



Approaches and their interaction

Formal methods

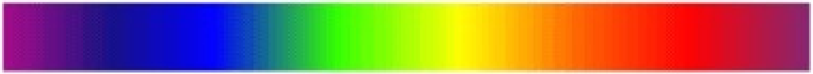
- precise
- scalability issues

Learning

- weaker guarantees
- scalable

precise computation

focus on important stuff

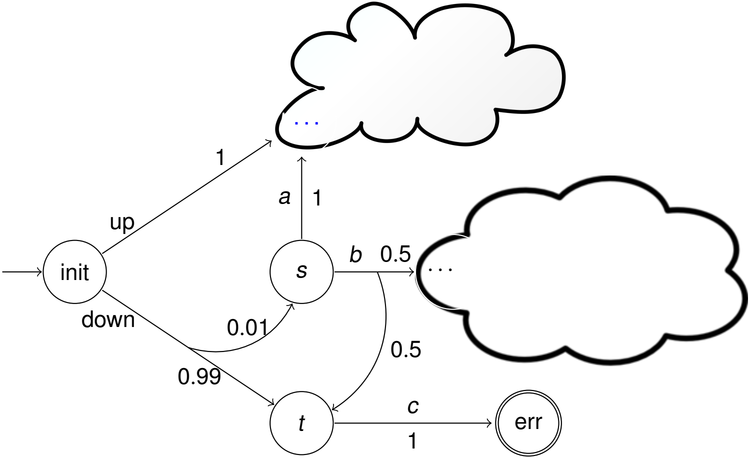


What problems?

- Verification
  - (ε)-optimality $\stackrel{?}{\rightarrow}$ PAC
  - hard guarantees $\stackrel{?}{\rightarrow}$ probably correct
- Controller synthesis
  - convergence is preferable
  - at least probably correct?
- Synthesis

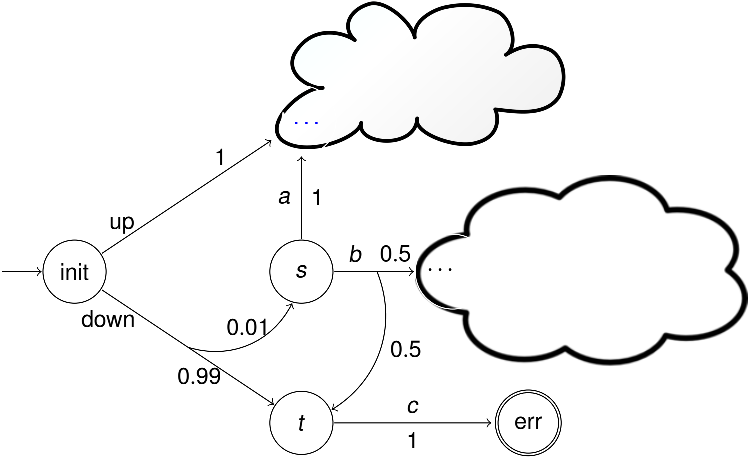
Markov decision processes

$$(S, s_0 \in S, A, \Delta : S \to A \to \mathcal{D}(S))$$



Markov decision processes

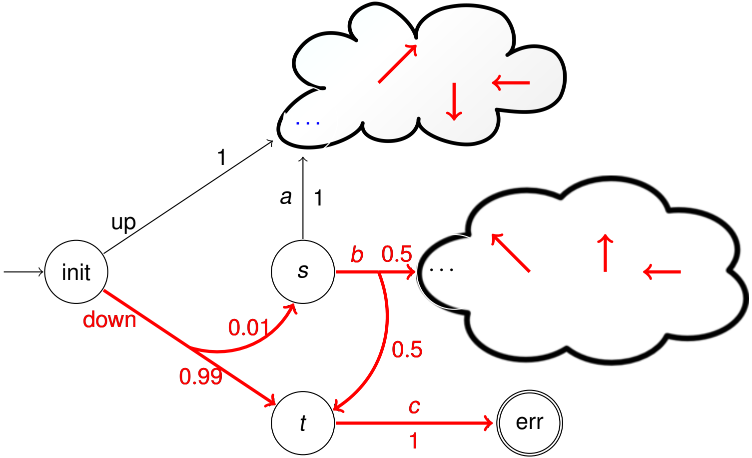
$(S, s_0 \in S, A, \Delta : S \to A \to \mathcal{D}(S))$



$\max_{\text{strategy } \sigma} \mathbb{P}^{\sigma}[\text{Reach err}]$

Markov decision processes

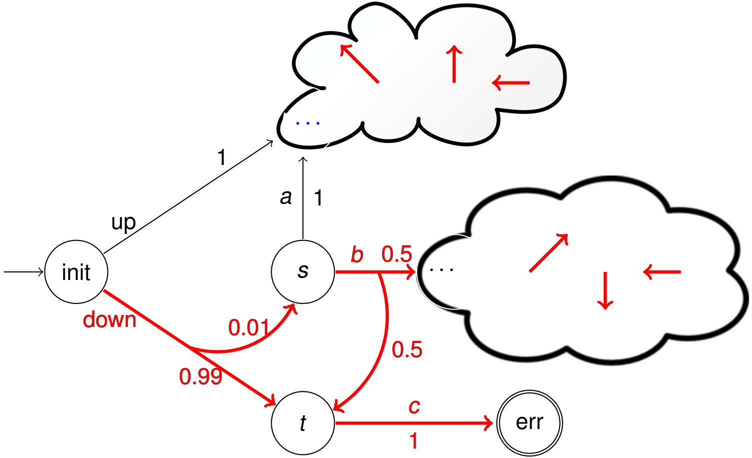
$$(S, s_0 \in S, A, \Delta : S \to A \to \mathcal{D}(S))$$



$$\max_{\text{strategy } \sigma} \mathbb{P}^{\sigma}[\text{Reach err}]$$

Markov decision processes

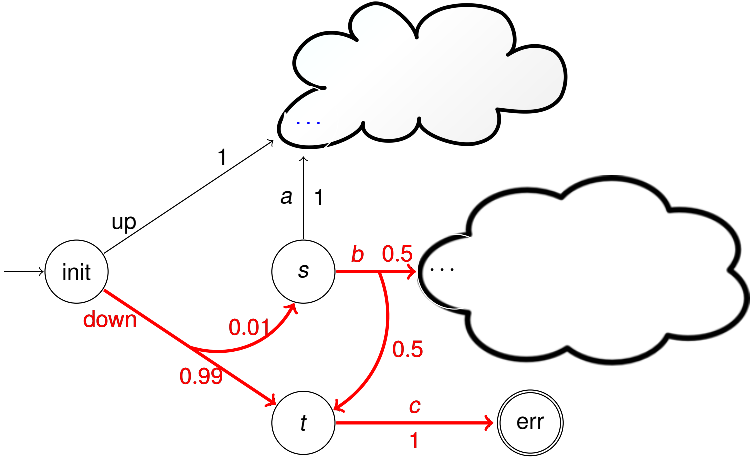
$$(S, s_0 \in S, A, \Delta : S \to A \to \mathcal{D}(S))$$



$$\max_{\text{strategy } \sigma} \mathbb{P}^{\sigma}[\text{Reach err}]$$

Markov decision processes

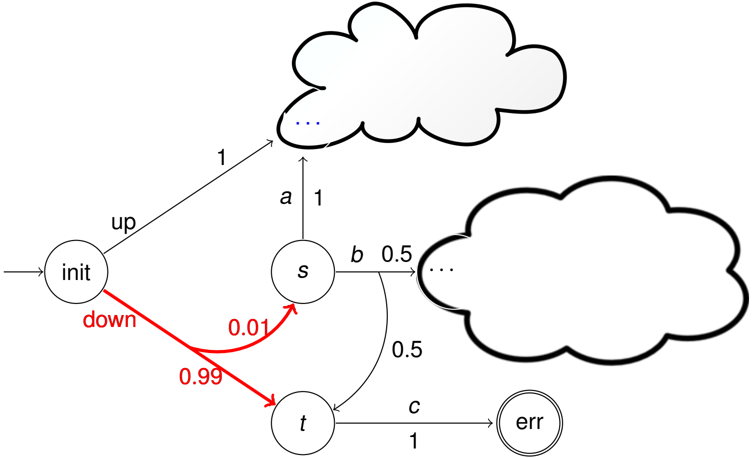
$$(S, s_0 \in S, A, \Delta : S \to A \to \mathcal{D}(S))$$



$$\max_{\text{strategy } \sigma} \mathbb{P}^{\sigma}[\text{Reach err}]$$

Markov decision processes

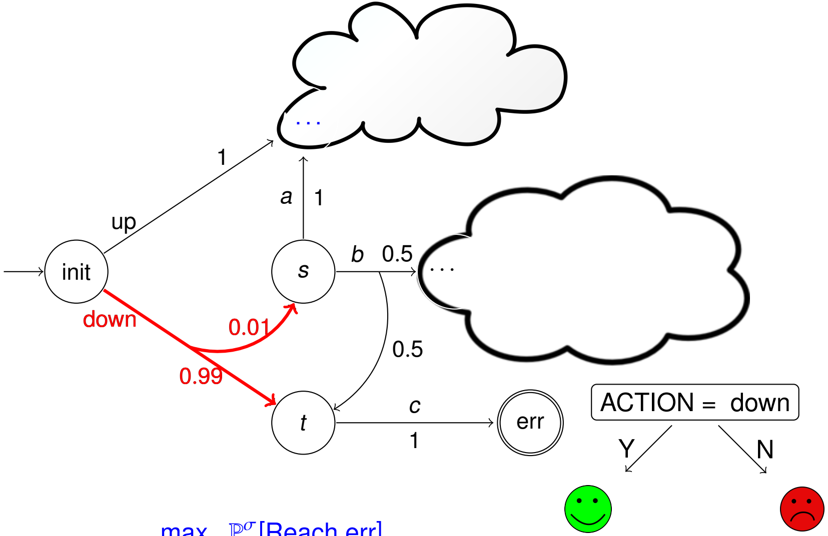
$(S, s_0 \in S, A, \Delta : S \to A \to \mathcal{D}(S))$



$\max_{\text{strategy } \sigma} \mathbb{P}^{\sigma}[\text{Reach err}]$

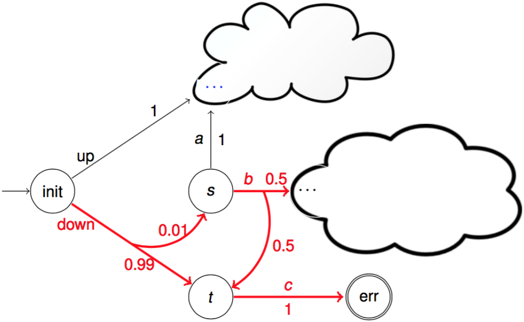
Markov decision processes

$$(S, s_0 \in S, A, \Delta : S \to A \to \mathcal{D}(S))$$



$$\max_{\text{strategy } \sigma} \mathbb{P}^{\sigma}[\text{Reach err}]$$

Ex.1: Computing strategies faster: How?

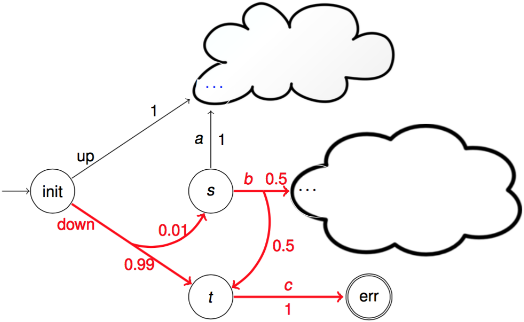


Fixed-point computation

$$V(s) := \max_{a \in \Delta(s)} V(s, a)$$

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

Ex.1: Computing strategies faster: How?



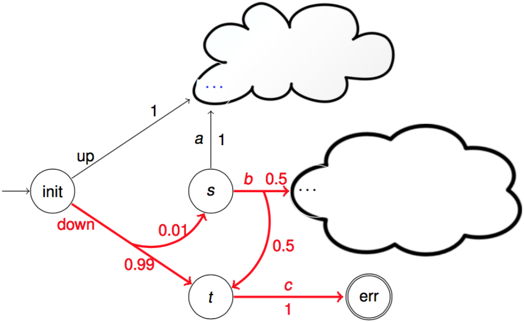
Fixed-point computation

$$V(s) := \max_{a \in \Delta(s)} V(s, a)$$

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

Order of evaluation?

Ex.1: Computing strategies faster: How?



Fixed-point computation

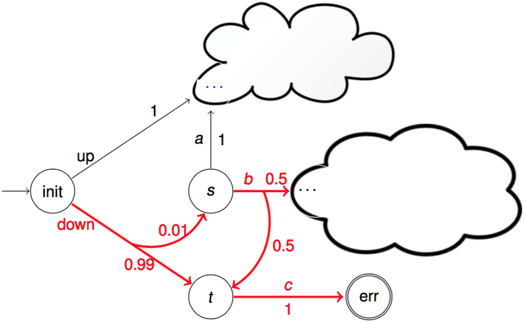
$$V(s) := \max_{a \in \Delta(s)} V(s,a)$$

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

Order of evaluation? [ATVA'14]

More frequently evaluate those states that are **visited** more frequently

Ex.1: Computing strategies faster: How?



Fixed-point computation

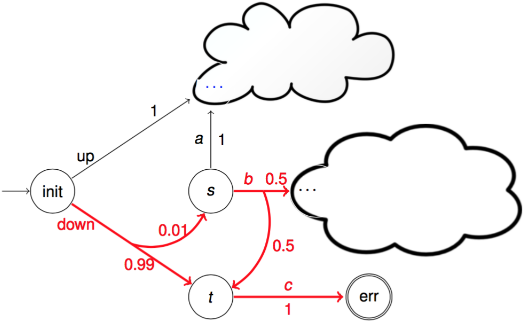
$$V(s) := \max_{a \in \Delta(s)} V(s,a)$$

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

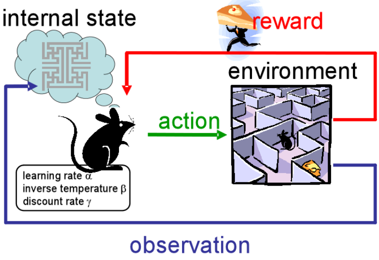
Order of evaluation? [ATVA'14]

More frequently evaluate those states that are **visited** more frequently by reasonably good schedulers

Ex.1: Computing strategies faster: How?



Reinforcement learning



Fixed-point computation

$$V(s) := \max_{a \in \Delta(s)} V(s,a)$$

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

Order of evaluation? [ATVA'14]

More frequently evaluate those states that are **visited** more frequently by reasonably good schedulers

Ex.1: Computing strategies faster: Algorithm

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

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

repeat

until $U(s_0) - L(s_0) < \epsilon$

Ex.1: Computing strategies faster: Algorithm

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

repeat
    sample a path from s_0 to $\{\mathbf{1},\mathbf{0}\}$
        ▹ actions uniformly from $\arg\max_a U(s,a)$
        ▹ states according to $\Delta(s,a,s')$

until $U(s_0) - L(s_0) < \epsilon$

Ex.1: Computing strategies faster: Algorithm

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

repeat
    sample a path from s_0 to $\{\mathbf{1},\mathbf{0}\}$
        ▹ actions uniformly from $\arg\max_a U(s,a)$
        ▹ states according to $\Delta(s,a,s')$

    for all visited transitions (s,a,s') **do**
        UPDATE(s,a,s')

until $U(s_0) - L(s_0) < \epsilon$

Ex.1: Computing strategies faster: Algorithm

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

repeat
    sample a path from s_0 to $\{\mathbf{1},\mathbf{0}\}$
        ▹ actions uniformly from $\arg\max_a U(s,a)$
        ▹ states according to $\Delta(s,a,s')$

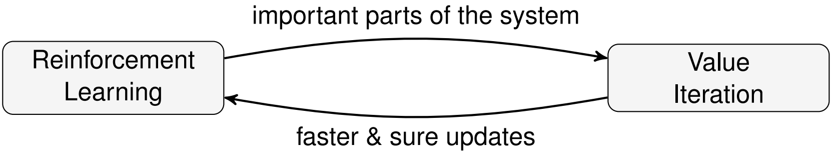
    for all visited transitions (s,a,s') **do**
        $\textsc{Update}(s,a,s')$

until $U(s_0) - L(s_0) < \epsilon$

---

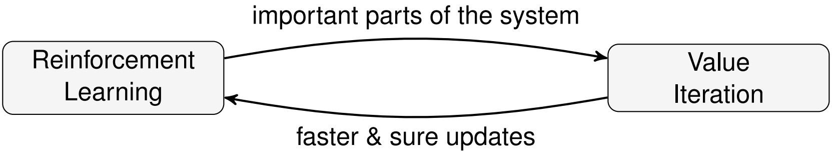
procedure $\textsc{Update}(s,a,\cdot)$
    $U(s,a) := \sum_{s' \in S} \Delta(s,a,s') \cdot U(s')$
    $L(s,a) := \sum_{s' \in S} \Delta(s,a,s') \cdot L(s')$

Ex.1: Computing strategies faster



Guaranteed upper & lower bounds at all times + practically fast convergence

Ex.1: Computing strategies faster



Guaranteed upper & lower bounds at all times + practically fast convergence

Remark:

- PAC SMC for MDP and (unbounded) LTL [ATVA'14]: $|S|, p_{\min}$
- practical PAC SMC for MC and (unbounded) LTL + mean payoff [TACAS'16]: $p_{\min}$

Ex.1: Experimental results

Example	Visited states	
	PRISM	BRTDP
zeroconf	3,001,911	760
	4,427,159	977
	5,477,150	1411
wlan	345,000	2018
	1,295,218	2053
	5,007,548	1995
firewire	6,719,773	26,508
	13,366,666	25,214
	19,213,802	32,214
mer	17,722,564	1950
	17,722,564	2902
	26,583,064	1950
	26,583,064	2900

Further examples on reinforcement learning

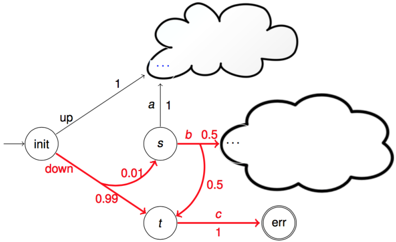
Sebastian Junges, Nils Jansen, Christian Dehnert, Ufuk Topcu, Joost-Pieter Katoen:
Safety-Constrained Reinforcement Learning for MDPs. TACAS 2016.

- safe and cost-optimizing strategies
- (1) compute safe, permissive strategies
- (2) learn cost-optimal strategies (convergence) among them

Alexandre David, Peter Gjl Jensen, Kim Guldstrand Larsen, Axel Legay, Didier Lime, Mathias Grund Srensen, Jakob Haahr Taankvist:
On Time with Minimal Expected Cost! ATVA 2014.

- priced timed games $\rightarrow$ priced timed MDPs
- time-bounded cost-optimal (convergence) strategies
- (1) Uppaal TiGa for safe strategies
- (2) Uppaal SMC and learning for cost-optimal strategies

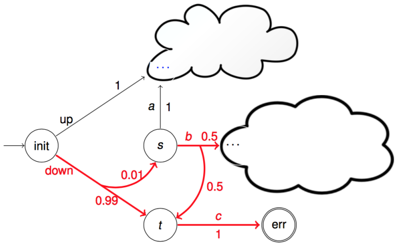
Ex.2: Computing small strategies: Which decisions?



Importance of a node *s* with respect to $\Diamond$*target* and strategy σ:

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

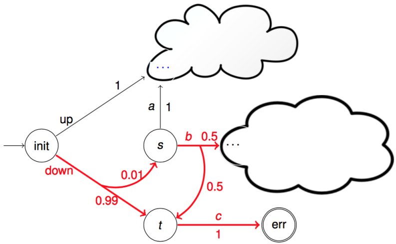
Ex.2: Computing small strategies: Which decisions?



Importance of a node s with respect to $\Diamond target$ and strategy σ:

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

Ex.2: Computing small strategies: Which decisions?

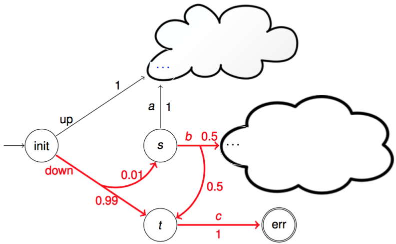


Importance of a node s with respect to $\Diamond$*target* and strategy σ:

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

Cut off states with zero importance (unreachable or useless)

Ex.2: Computing small strategies: Which decisions?



Importance of a node s with respect to $\Diamond$*target* and strategy σ:

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

Cut off states with zero importance (unreachable or useless)

Cut off states with low importance (small error, ε-optimal strategy)

Ex.2: Small strategies: Which representation?

How to make use of the exact importance?

Ex.2: Small strategies: Which representation?

How to make use of the exact importance?

Learn decisions in s in proportion to importance of s

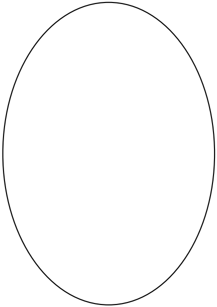
Ex.2: Small strategies: Which representation?

How to make use of the exact importance?

Learn decisions in s in proportion to importance of s

Advantages of decision trees over BDDs:

- more readable: predicates
- smaller due to good ordering: entropy
- yet smaller at a cost of an error: pruning



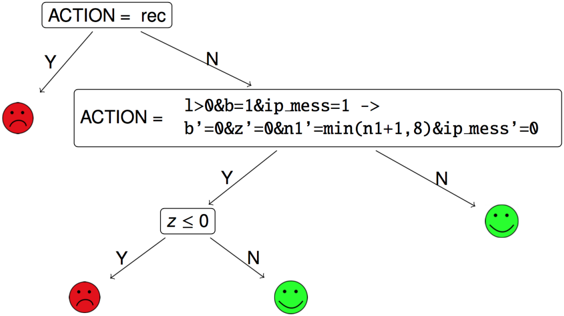
Ex.2: Small strategies: Which representation?

How to make use of the exact importance?

Learn decisions in s in proportion to importance of s

Advantages of decision trees over BDDs:

- more readable: predicates
- smaller due to good ordering: entropy
- yet smaller at a cost of an error: pruning



Ex.2: Experimental results

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

Ex.2: Experimental results

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 BRTDP yields: 1887 619 13 0.00014

Further examples on decision trees

Pranav Garg, Daniel Neider, P. Madhusudan, Dan Roth:
Learning Invariants using Decision Trees and Implication Counterexamples. POPL 2016.

- positive examples from runs of the program
- negative examples from modifications
- implication examples

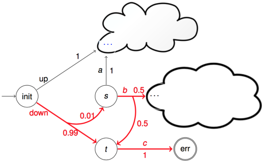
Siddharth Krishna, Christian Puhrsch, Thomas Wies:
Learning Invariants Using Decision Trees.

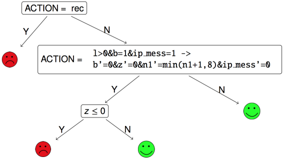
- positive examples: states reachable when preconditions holds
- negative examples: states leaving loop and violating a postcondition

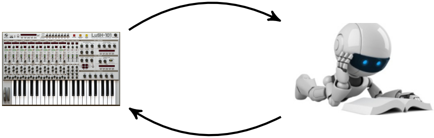
Summary

Machine learning in verification

- **Scalable heuristics**
- Example 1: **Speeding up** value iteration
  - TECHNIQUE: reinforcement learning, BRTDP
  - 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

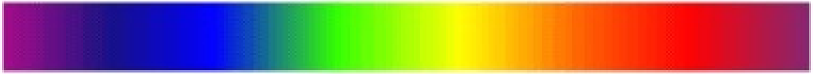






Discussion

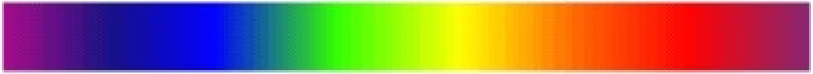
Verification using machine learning



- 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 shouldnt overrule guarantees?
    - SMC should be PAC; when is it enough?
- Oracle usage seems fine

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 shouldnt overrule guarantees?
    - SMC should be PAC; when is it enough?
- Oracle usage seems fine

Thank you