Learning Small Strategies Fast

Jan Křetínský

Technical University of Munich, Germany

joint work with
P. Ashok, E. Kelmendi, J. Krämer, T. Meggendorfer, M. Weininger (TUM)
T. Brázdil (Masaryk University Brno),
K. Chatterjee, M. Chmelík, P. Daca, A. Fellner, T. Henzinger, T. Petrov, V. Toman (IST Austria),
V. Forejt, M. Kwiatkowska, M. Ujma (Oxford University)
D. Parker (University of Birmingham)

Logic and Learning

The Alan Turing Institute
January 12, 2018

Controller synthesis and verification

Controller synthesis and verification

Formal methods

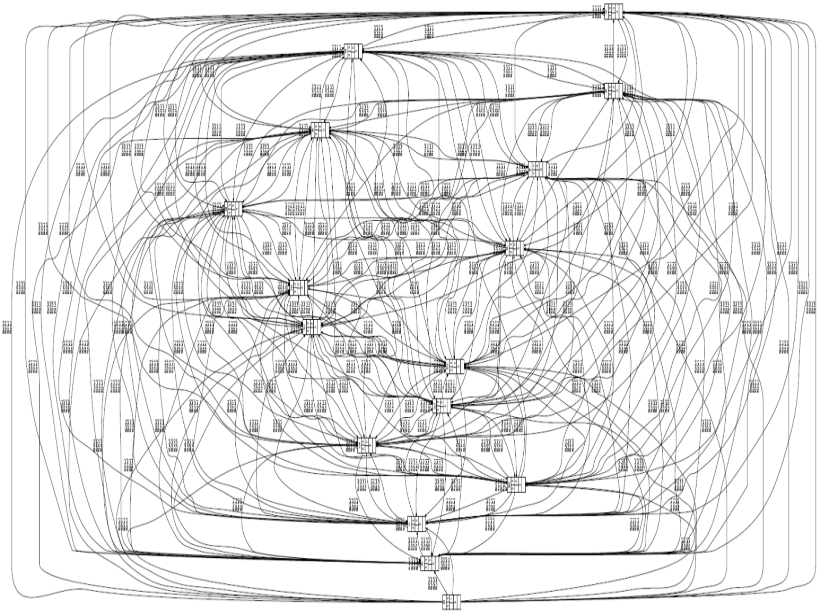
- \+ precise
- – scalability issues

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

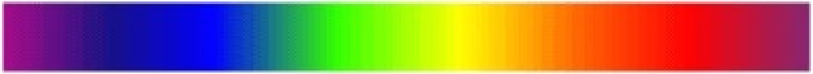
Formal methods and machine learning

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

Examples

- 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

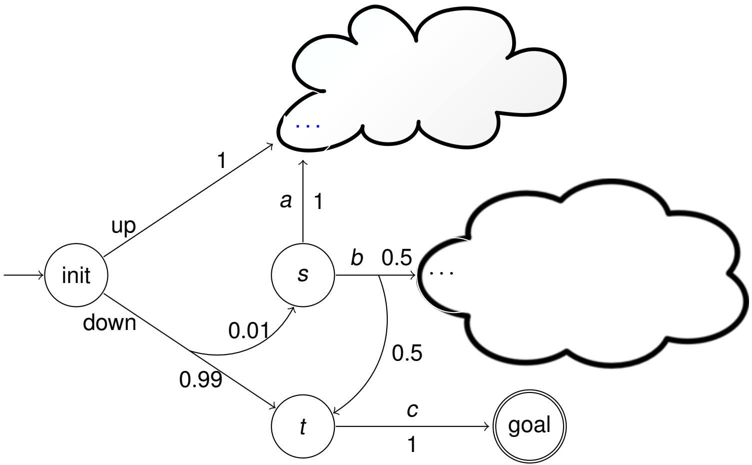
[4] K., Meggendorfer: Efficient Strategy Iteration for Mean Payoff in Markov Decision Processes. ATVA 2017

[5] draft

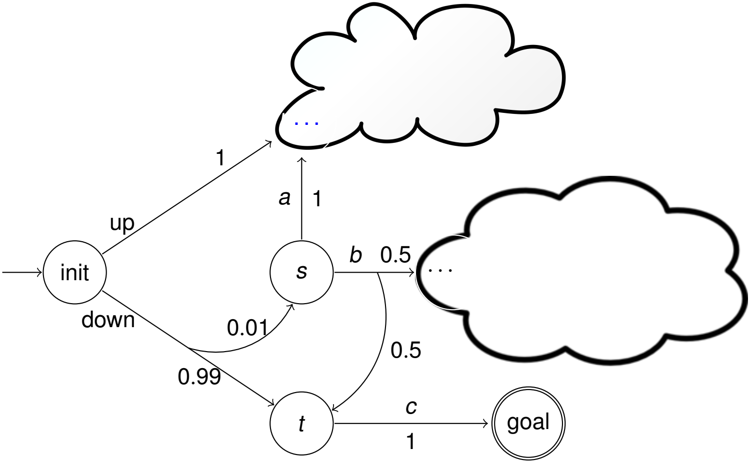
[6] Brazdil, Chatterjee, Chmelik, Fellner, K.: Counterexample Explanation by Learning Small Strategies in Markov Decision Processes. CAV 2015

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

Example: Markov decision processes

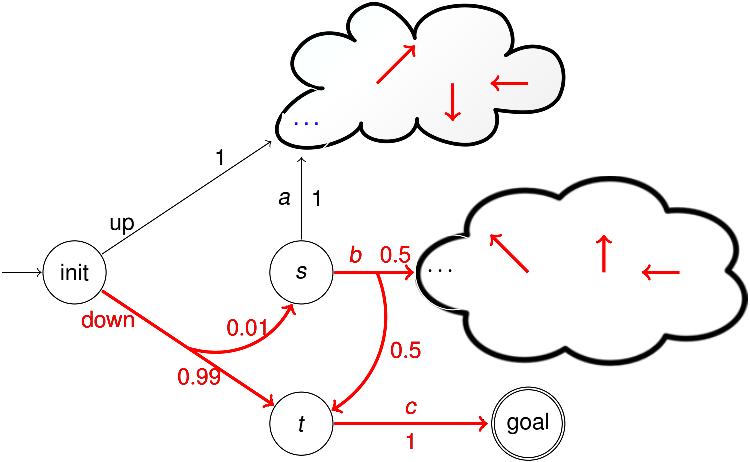


Example: Markov decision processes



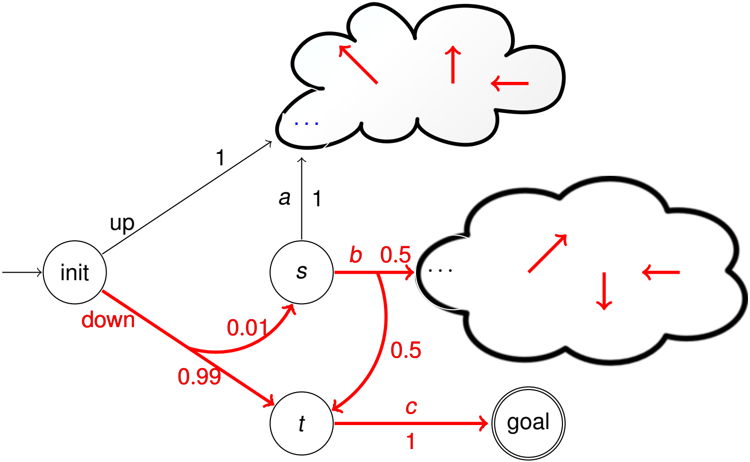
$$\max_{\text{strategy } \sigma} \mathbb{P}^{\sigma}[\Diamond \text{ goal}]$$

Example: Markov decision processes



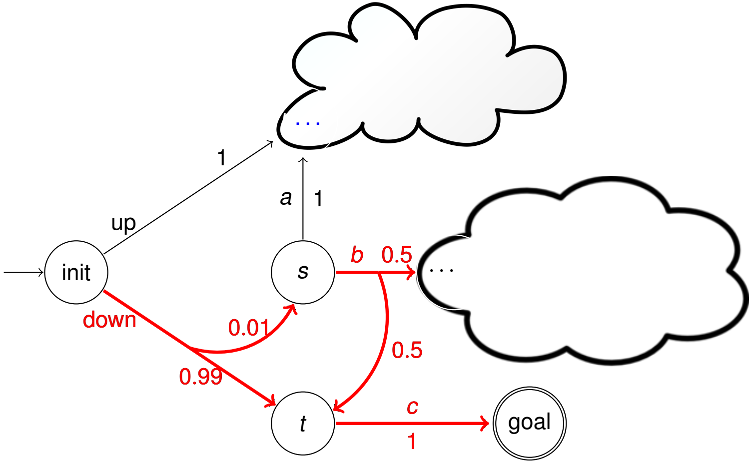
$$\max_{\text{strategy } \sigma} \mathbb{P}^{\sigma}[\Diamond \text{ goal}]$$

Example: Markov decision processes



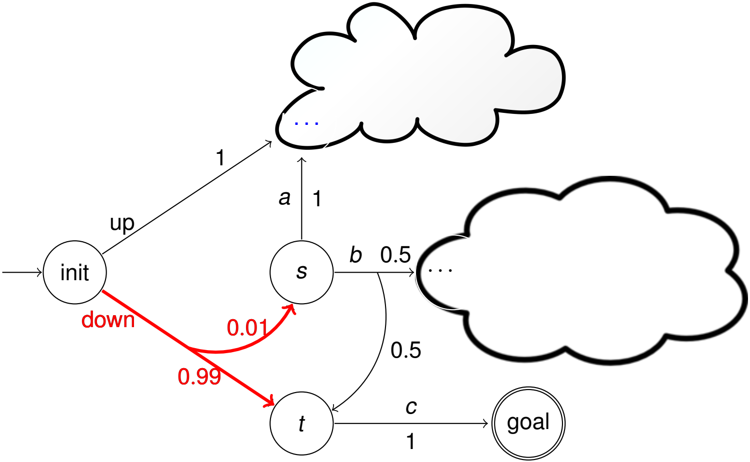
$$\max_{\text{strategy } \sigma} \mathbb{P}^{\sigma}[\Diamond \text{ goal}]$$

Example: Markov decision processes



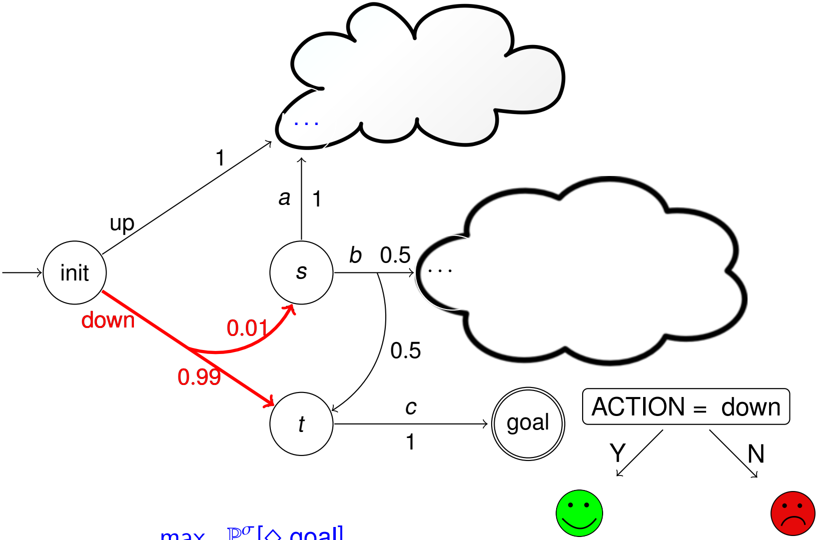
$$\max_{\text{strategy } \sigma} \mathbb{P}^{\sigma}[\Diamond \text{ goal}]$$

Example: Markov decision processes



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

Example: Markov decision processes



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

Example 1: Computing strategies faster

repeat

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

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

until $\mathit{UpBound}(s_{\mathrm{init}}) - \mathit{LoBound}(s_{\mathrm{init}}) < \epsilon$

---

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

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

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

    $\mathit{UpBound}(s) := \max_{a \in A} \mathit{UpBound}(s,a)$

    $\mathit{LoBound}(s) := \max_{a \in A} \mathit{LoBound}(s,a)$

Example 1: Computing strategies faster

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

---

repeat

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

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

until $\mathit{UpBound}(s_{\mathrm{init}}) - \mathit{LoBound}(s_{\mathrm{init}}) < \epsilon$

Example 1: Computing strategies faster

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

---

repeat
    sample a path from s_{init}
    for all visited transitions $s \xrightarrow{a}$ **do**
        $\textsc{Update}(s \xrightarrow{a})$
until $\mathit{UpBound}(s_{\text{init}}) - \mathit{LoBound}(s_{\text{init}}) < \epsilon$

Example 1: Computing strategies faster

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

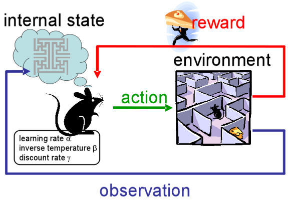
1: **repeat**
2: sample a path from s_{init}
3: **for all** visited transitions $s \xrightarrow{a}$ **do**
4: $\textsc{Update}(s \xrightarrow{a})$
5: **until** $\mathit{UpBound}(s_{\text{init}}) - \mathit{LoBound}(s_{\text{init}}) < \epsilon$

Example 1: Computing strategies faster

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

---

repeat
    sample a path from s_{init}
    for all visited transitions $s \xrightarrow{a}$ **do**
        $\textsc{Update}(s \xrightarrow{a})$
until $\mathit{UpBound}(s_{\text{init}}) - \mathit{LoBound}(s_{\text{init}}) < \epsilon$



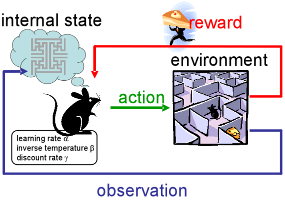
Example 1: Computing strategies faster

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

---

repeat
  sample a path from s_{init} ▷ pick action $\arg\max_a \mathit{UpBound}(s \xrightarrow{a})$
  for all visited transitions $s \xrightarrow{a}$ **do**
    $\textsc{Update}(s \xrightarrow{a})$

until $\mathit{UpBound}(s_{\mathrm{init}}) - \mathit{LoBound}(s_{\mathrm{init}}) < \epsilon$



Example 1: Computing strategies faster

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

---

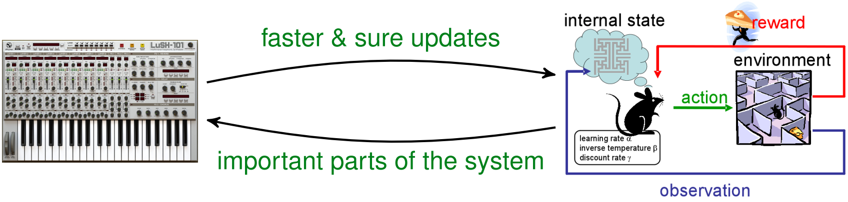
repeat

  sample a path from s_{init} ▷ pick action $\arg\max_a \mathit{UpBound}(s \xrightarrow{a})$

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

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

until $\mathit{UpBound}(s_{\mathit{init}}) - \mathit{LoBound}(s_{\mathit{init}}) < \epsilon$

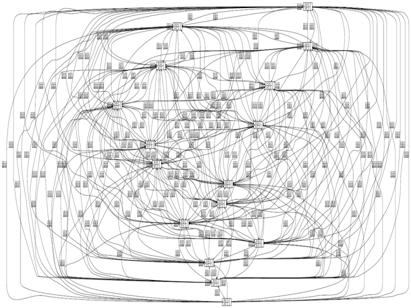


Example 1: Experimental results

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

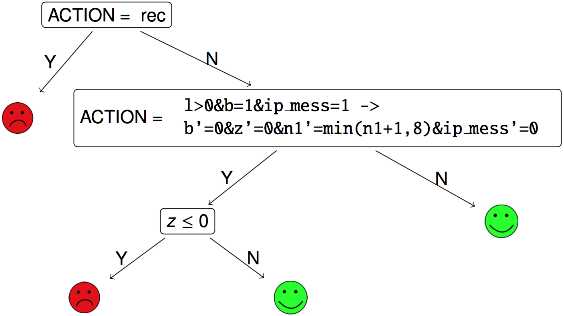
Example 2: Computing small strategies

- explicit map $\sigma : S \to A$
- BDD (binary decision diagrams) encoding its bit representation
- DT (decision tree)

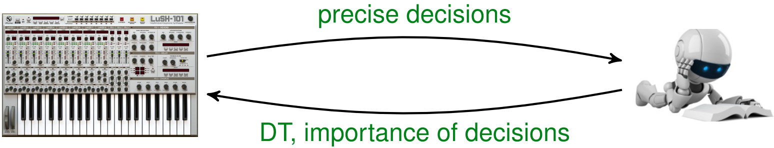


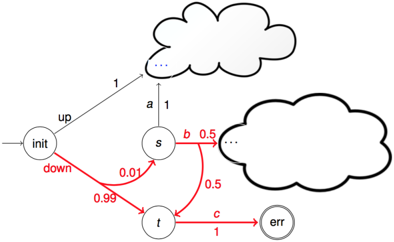
Example 2: Computing small strategies

- explicit map $\sigma : S \to A$
- BDD (binary decision diagrams) encoding its bit representation
- DT (decision tree)



Example 2: Computing small strategies





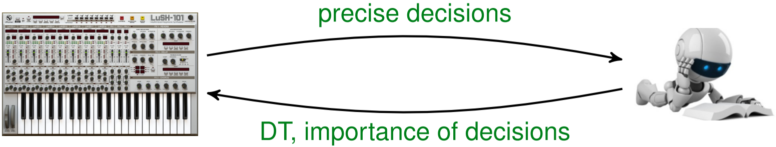
Cut off states with zero importance (unreachable or useless)

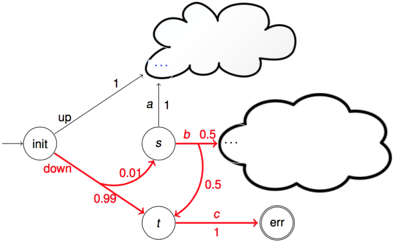
Cut off states with low importance (small error, ε-optimal strategy)

How to make use of the exact quantities?

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

Example 2: Computing small strategies





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?

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

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

Example 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

Example 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 RL yields: 1887 619 13 0.00014

Some related work

Reinforcement learning in verification

- Junges, Jansen, Dehnert, Topcu, Katoen: *Safety-Constrained Reinforcement Learning for MDPs.* TACAS 2016
- David, Jensen, Larsen, Legay, Lime, Sorensen, Taankvist: *On Time with Minimal Expected Cost!* ATVA 2014

Strategy representation learning

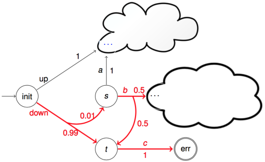
- Neider, Topcu: *An Automaton Learning Approach to Solving Safety Games over Infinite Graphs.* TACAS 2016

Invariants generation, theorem provers guidance, ...

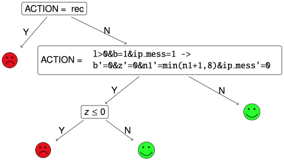
Summary

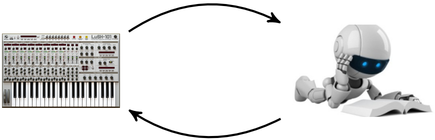
Machine learning in verification

- **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



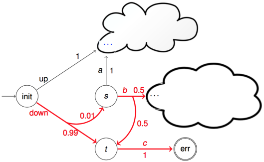


- Learning in Verification (LiVe) at ETAPS

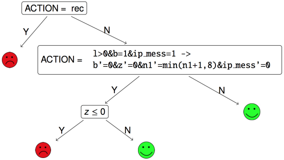
Summary

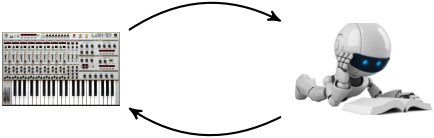
Machine learning in verification

- **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

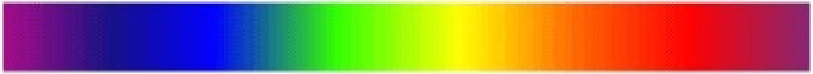




- Learning in Verification (LiVe) at ETAPS

Thank you

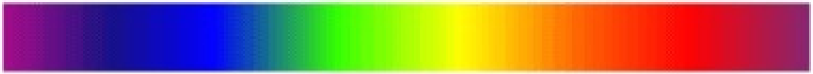
Verification using machine learning



- 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?

Thank you