Policies Grow on Trees: Model Checking Families of MDPs*

Filip Macák

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

VERIFIT

*See the paper: Andriushchenko R., Češka M., Junges S. and Macák F.: **Policies Grow on Trees: Model Checking Families of MDPs.** (accepted to ATVA'24)

Motivation

Previous work: exploring families of discrete-time Markov chains (DTMCs)

- synthesis of discrete-time probabilistic programs
- synthesis of Markov decision process (MDP) controllers wrt. hyperproperties
- synthesis of finite-state controllers for POMDPs

The family can be viewed as a DTMC with controllable or uncontrollable parameters

- controllable choice of the strategy of the agent
- uncontrollable choice of the environment or the adversary strategy

Motivation

Often, we need to reason about controllable and uncontrollable choices

- planning in multiple controllable environments
- we don't know the exact environment

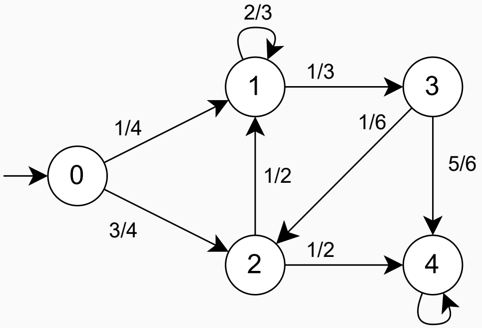
Case study examples:

- dynamic power management of a request processing device
  - parameters affect the device components and the client behavior
- virus attack on a computer network
  - parameters affect network topology and node vulnerabilities
- agent navigating in a grid-like environment
  - parameters affect the environment and the behavior of adversary agents

Markov decision processes recap

DTMC = discrete-time state transition system that evolves stochastically

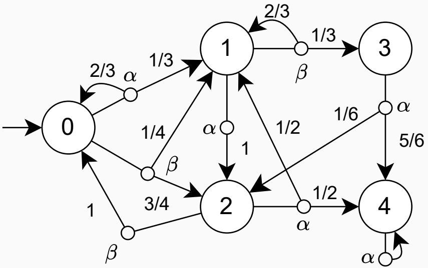
- typical query: $P(\mathrm{F}\ T) \geq 0.9 \equiv$ verify whether the probability $P(\mathrm{F}\ T)$ of reaching the set T of target states is at least 90%



Markov decision processes recap

MDP = DTMC + nondeterministic actions

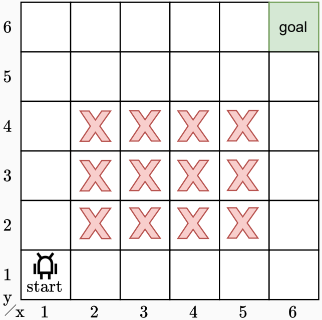
- (memoryless & deterministic) controller (scheduler, policy) $\sigma : S \to Act$ resolves the nondeterminism
- MDP M + controller σ = DTMC M^σ
- typical query: $\max_\sigma P(M^\sigma \models \mathrm{F}\ T) \equiv$ find controller σ that maximizes the probability of reaching T



Family of MDPs

Family $\{M_i\}_{i \in \mathcal{I}}$ of MDPs = MDP with parameters

- parameters affect MDP topology
- $i \in \mathcal{I}$ is a parameter assignment, $|\mathcal{I}| < \infty$
- choice of parameter assignment $i \in \mathcal{I}$ represents uncontrollable nondeterminism (adversary, environment)
- choice of action $\alpha \in Act$ represents controllable nondeterminism



- parameters: $OX = \{2, 3, 4, 5\}$ and $OY = \{2, 3, 4\}$

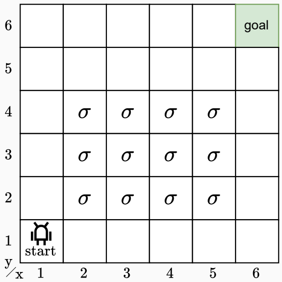
Robustness problem

input: family $\{M_i\}_{i\in\mathcal{I}}$ of MDPs

input: PCTL reachability property $P(\mathrm{F}\ T) \bowtie \lambda$

output: robust controller σ s.t. $\forall i \in \mathcal{I}$: $P(M_i^\sigma \models \mathrm{F}\ T) \bowtie \lambda$

- requires non-memoryless controllers
- related to solving POMDPs

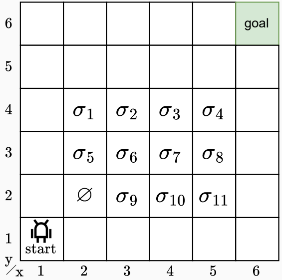


Problem statement

input: family $\{M_i\}_{i \in \mathcal{I}}$ of MDPs

input: PCTL reachability property $P(\mathrm{F}\ T) \bowtie \lambda$

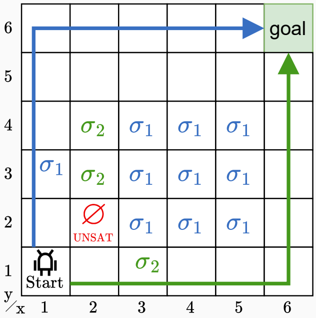
output: for each parameter assignment $i \in \mathcal{I}$ a controller σ_i s.t. $P\left(M_i^{\sigma_i} \models \mathrm{F}\ T\right) \bowtie \lambda$ (if such σ_i exists)

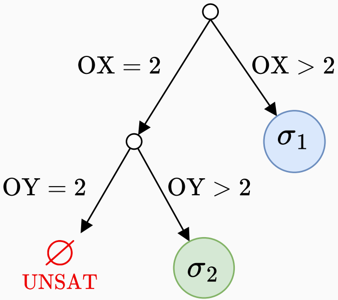


Problem statement

Additional requirement: produce a decision tree of controllers

- nodes of the tree reason about a single parameter
- leaves of the tree (describing sub-families) contain controllers (or ∅)
- space-efficient, fast lookup, more understandable for engineers





Naive approaches

One-by-one enumeration

- computationally-intensive
- produces a list of controllers
- unsuitable for large families

All-in-one abstraction + BDD encoding

- computationally- and memory-intensive
- produces a more compact decision tree
    - export is not supported by existing tools
- not all problems can be efficiently encoded

Proposed solution

Algorithm 1 Policy tree synthesis

Input: family $\mathcal{M} = \{M_i\}_{i \in \mathcal{I}}$ of MDPs, PCTL property φ

Output: policy tree for $\mathcal{M}$ wrt. φ

function $\textsc{buildTree}(\mathcal{M}, \varphi)$
    $\sigma \leftarrow$ try to find a robust controller for $\mathcal{M}$ wrt. φ
    if succeeded **then**
        return $\textsc{LeafNode}(\mathcal{M}, \sigma)$
    try to prove that no $M_i \in \mathcal{M}$ can satisfy φ
    if succeeded **then**
        return $\textsc{LeafNode}(\mathcal{M}, \emptyset)$
    $\mathcal{M}', \mathcal{M}'' \leftarrow \text{split}(\mathcal{M})$
    return $\textsc{InnerNode}(\mathcal{M}, \textsc{buildTree}(\mathcal{M}', \varphi), \textsc{buildTree}(\mathcal{M}'', \varphi))$

- gist: given a family of MDPs, try to find a robust controller or try to prove that no satisfying MDP exists, split the family if a conclusive result was not obtained

Proposed solution

Algorithm 1 Policy tree synthesis

Input: family $\mathcal{M} = \{M_i\}_{i \in \mathcal{I}}$ of MDPs, PCTL property φ

Output: policy tree for $\mathcal{M}$ wrt. φ

function $\textsc{buildTree}(\mathcal{M}, \varphi)$
    $\sigma \leftarrow$ try to find a robust controller for $\mathcal{M}$ wrt. φ
    if succeeded **then**
        return $\textsc{LeafNode}(\mathcal{M}, \sigma)$
    try to prove that no $M_i \in \mathcal{M}$ can satisfy φ
    if succeeded **then**
        return $\textsc{LeafNode}(\mathcal{M}, \emptyset)$
    $\mathcal{M}', \mathcal{M}'' \leftarrow$ split$(\mathcal{M})$
    return $\textsc{InnerNode}(\mathcal{M}, \textsc{buildTree}(\mathcal{M}', \varphi), \textsc{buildTree}(\mathcal{M}'', \varphi))$

- gist: given a family of MDPs, try to find a robust controller or try to prove that no satisfying MDP exists, split the family if a conclusive result was not obtained

Proposed solution

Algorithm 1 Policy tree synthesis

Input: family $\mathcal{M} = \{M_i\}_{i \in \mathcal{I}}$ of MDPs, PCTL property φ

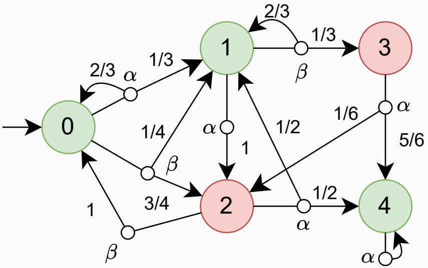
Output: policy tree for $\mathcal{M}$ wrt. φ

function $\textsc{buildTree}(\mathcal{M}, \varphi)$
    $\sigma \leftarrow$ try to find a robust controller for $\mathcal{M}$ wrt. φ
    if succeeded **then**
        return $\textsc{LeafNode}(\mathcal{M}, \sigma)$
    try to prove that no $M_i \in \mathcal{M}$ can satisfy φ
    if succeeded **then**
        return $\textsc{LeafNode}(\mathcal{M}, \varnothing)$
    $\mathcal{M}', \mathcal{M}'' \leftarrow \text{split}(\mathcal{M})$
    return $\textsc{InnerNode}(\mathcal{M}, \textsc{buildTree}(\mathcal{M}', \varphi), \textsc{buildTree}(\mathcal{M}'', \varphi))$

How to find a robust controller?

Stochastic game

Stochastic game $\mathcal{G}$ = MDP with its states partitioned into Player 1 and Player 2 states

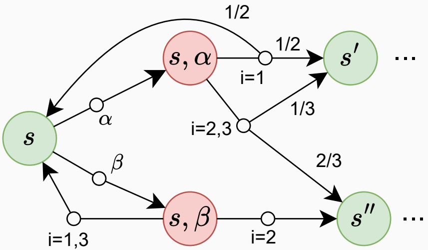


- controller is a pair $\sigma = (\sigma_1, \sigma_2)$ of Player 1 & Player 2 controllers
- Player 1 maximizes, Player 2 minimizes the reachability probability:

$$\max_{\sigma_1} \min_{\sigma_2} P(\mathcal{G}^{\sigma_1 \sigma_2} \models \mathrm{F}\ T)$$

Stochastic game abstraction

- Player 1 picks an action, Player 2 picks a parameter assignment



the above is an over-approximation since Player 2 is too powerful:

- Player 2 can pick parameter assignments inconsistently
    - consistent abstraction would mimic the all-in-one abstraction
- Player 2 acts second
    - this order avoids the abstraction blow-up

Robust policy heuristic

- assume a family $\mathcal{M}$ of MDPs and a specification $P(\mathrm{F}\ T) \geq 0.9$
- construct game abstraction $\mathcal{G}(\mathcal{M})$
- the following is a sufficient (but not necessary) condition for σ_1 to be a robust controller for $\mathcal{M}$:

$$\max_{\sigma_1} \min_{\sigma_2} P(\mathcal{G}(\mathcal{M})^{\sigma_1 \sigma_2} \models \mathrm{F}\ T) \geq 0.9$$

- if the above condition does *not* hold and σ_2 is consistent in its parameter assignment, then this assignment is unsatisfiable

Proposed solution

Algorithm 1 Policy tree synthesis

Input: family $\mathcal{M} = \{M_i\}_{i \in \mathcal{I}}$ of MDPs, PCTL property φ

Output: policy tree for $\mathcal{M}$ wrt. φ

```
function buildTree(𝓜, φ)
    σ ← try to find a robust controller for 𝓜 wrt. φ
    if succeeded then
        return LeafNode(𝓜, σ)
    try to prove that no M_i ∈ 𝓜 can satisfy φ
    if succeeded then
        return LeafNode(𝓜, ∅)
    𝓜′, 𝓜″ ← split(𝓜)
    return InnerNode(𝓜, buildTree(𝓜′, φ), buildTree(𝓜″, φ))
```

How to prove a family is unsatisfiable?

Proving unsatisfiability heuristic

- assume a family $\mathcal{M}$ of MDPs and a specification $P(\mathrm{F}\ T) \geq 0.9$
- the following is a sufficient (but not necessary) condition for no MDP in $\mathcal{M}$ being satisfiable:

$$\max_{\sigma_1} \max_{\sigma_2} P(\mathcal{G}(\mathcal{M})^{\sigma_1 \sigma_2} \models \mathrm{F}\ T) < 0.9$$

- such "game" abstraction is simply an MDP
- if the above condition does *not* hold and σ_2 is consistent in its parameter assignment, then this assignment is satisfiable

Proposed solution

Algorithm 1 Policy tree synthesis

Input: family $\mathcal{M} = \{M_i\}_{i \in \mathcal{I}}$ of MDPs, PCTL property φ

Output: policy tree for $\mathcal{M}$ wrt. φ

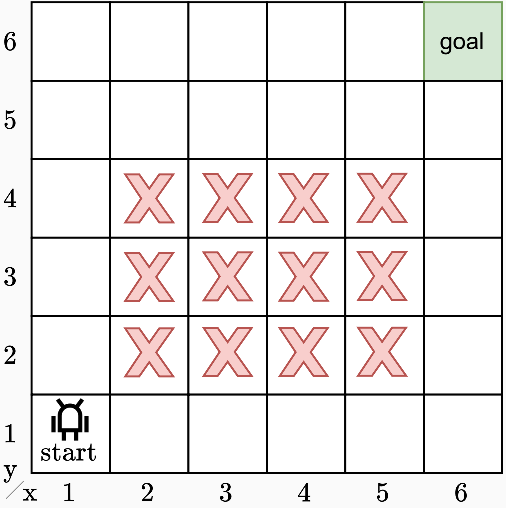
function $\textsc{buildTree}(\mathcal{M}, \varphi)$
    $\sigma \leftarrow$ try to find a robust controller for $\mathcal{M}$ wrt. φ
    if succeeded **then**
        return $\textsc{LeafNode}(\mathcal{M}, \sigma)$
    try to prove that no $M_i \in \mathcal{M}$ can satisfy φ
    if succeeded **then**
        return $\textsc{LeafNode}(\mathcal{M}, \varnothing)$
    $\mathcal{M}', \mathcal{M}'' \leftarrow \text{split}(\mathcal{M})$
    return $\textsc{InnerNode}(\mathcal{M}, \textsc{buildTree}(\mathcal{M}', \varphi), \textsc{buildTree}(\mathcal{M}'', \varphi))$

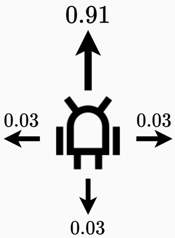
How to split a family?

Abstraction refinement

Abstraction refinement step: if neither of the tests was successful, we split family $\mathcal{M}$ into smaller subfamilies based on the controller (σ_1, σ_2) for the game abstraction $\mathcal{G}(\mathcal{M})$

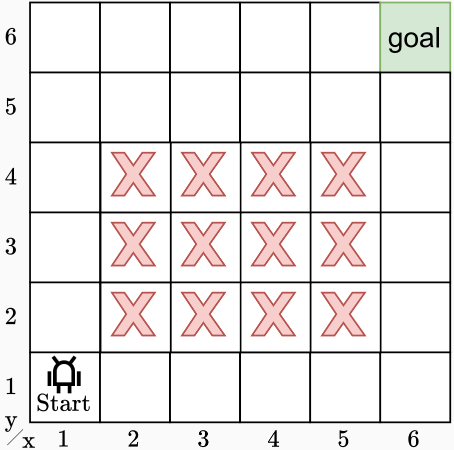
- if σ_2 is not consistent i.e. in parameter X, we split wrt. X to disallow such an inconsistency in the subfamilies
- if σ_2 is consistent, representing some satisfiable assignment i, we try to separate i (and other assignments in which σ_2 is consistent) into a smaller subfamily





- parameters: $OX = \{2, 3, 4, 5\}$ and $OY = \{2, 3, 4\}$
- crashing = going into sink state
- specification: $P(\mathrm{F}\, goal) \geq 0.99$

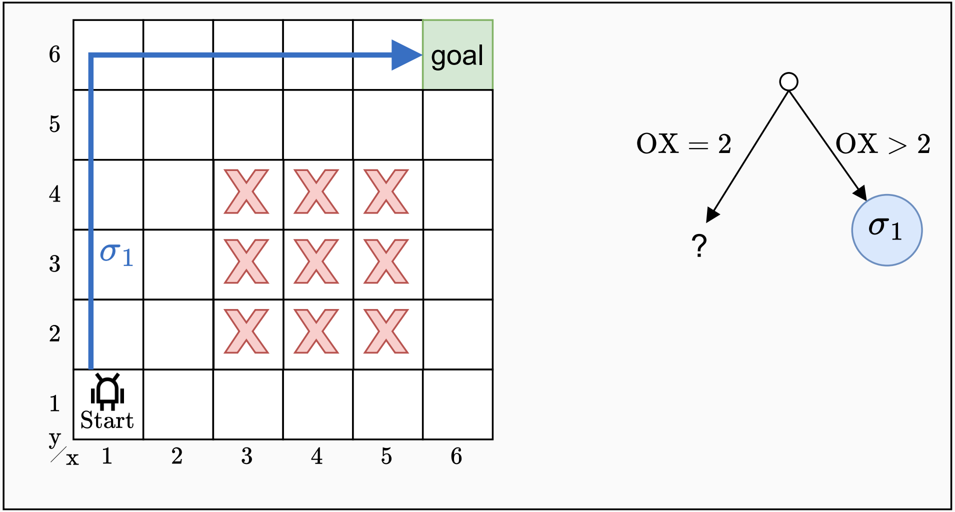
Example



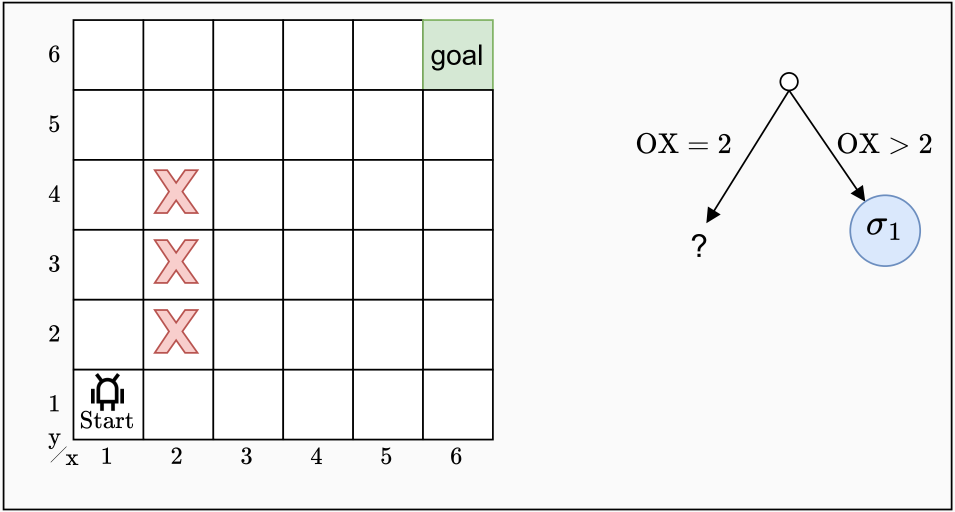
○

?

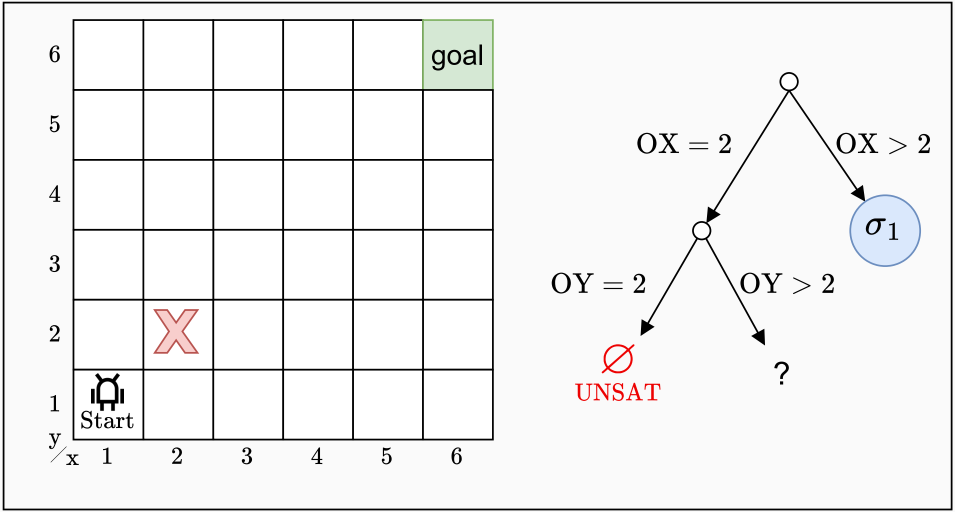
Example



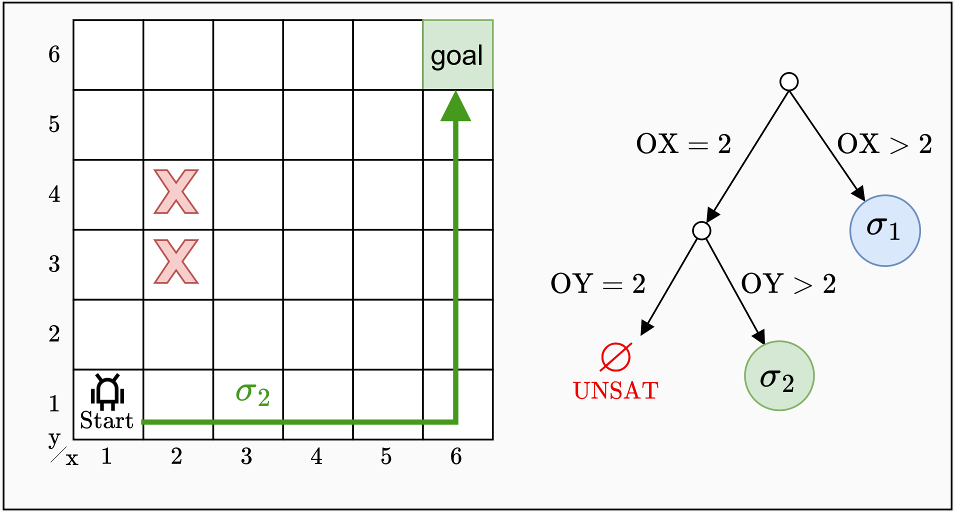
Example

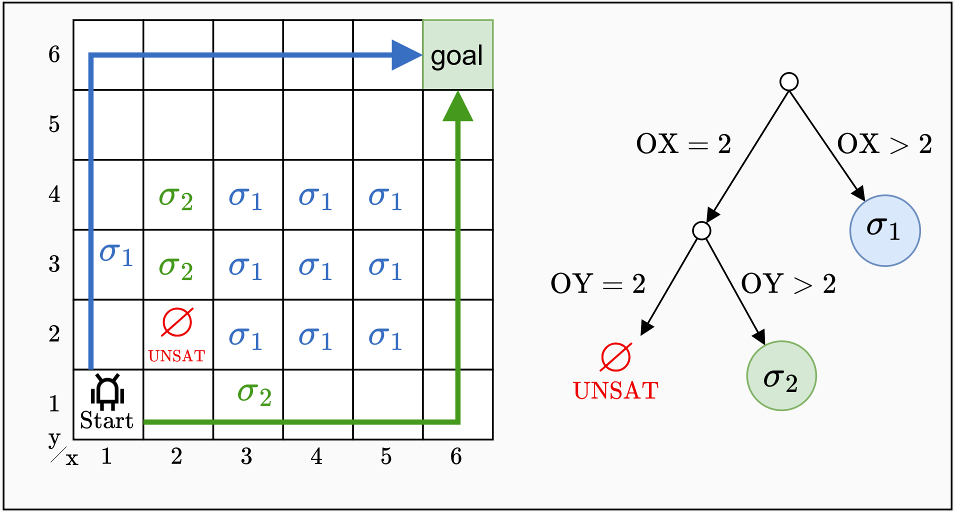


Example



Example

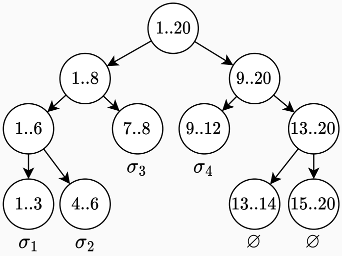


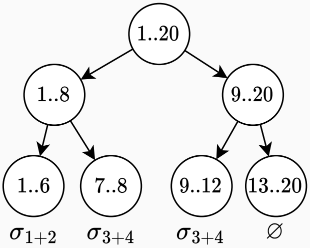

goal
6
5
4
3
2
1
y
x
1
2
3
4
5
6
Start
UNSAT
σ_1
σ_2
OX = 2
OX > 2
OY = 2
OY > 2
UNSAT
σ_1
σ_2

Tree post processing

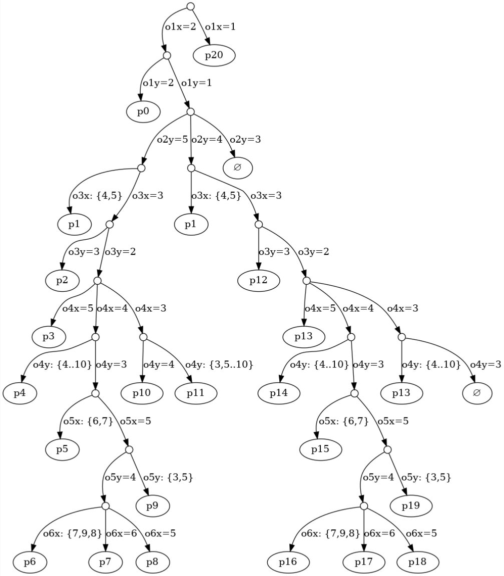
Three post-processing steps:

1. for each leaf siblings pair with policies σ_l and σ_r for subfamilies $\mathcal{M}_l$ and $\mathcal{M}_r$, verify robustness of σ_{l+r} and σ_{r+l}. If one of them is robust, join the two leaves.
2. combine each pair of compatible policies
3. join all sibling leaves which are denoted by the same policy (or are unsat)





Decision tree example



Experimental results

model	model info			our approach		speedup wrt.	
	$\lvert S_{\mathcal{M}} \rvert$	$\lvert \mathcal{M} \rvert$	SAT %	P/SAT %	time	1-by-1	all-in-1
dodge-2	2e5	3e4	100	0.1	**122**	8	1.1
dodge-3	2e5	9e7	100	<0.01	**1445**	†1764	MO
dpm-10-b	9e3	1e5	22	0.02	**74**	21	TO
obs-8-6	5e2	5e4	90	0.6	**6**	4	1.5
obs-10-6	8e2	3e6	98	<0.01	**5**	412	MO
obs-10-9	1e3	4e8	100	<0.01	**259**	†1661	MO
rov-1000	2e4	4e6	99	0.03	**1402**	†65	TO
uav-work	9e3	2e6	99	<0.01	**113**	55	TO
virus	2e3	7e4	83	0.9	50	**0.8**	TO
rocks-6-4	3e3	7e3	100	34	102	0.2	**0.1**

Experimental results

model	model info			our approach		speedup wrt.	
	$\lvert S_{\mathcal{M}} \rvert$	$\lvert \mathcal{M} \rvert$	SAT %	P/SAT %	time	1-by-1	all-in-1
dodge-2	2e5	3e4	100	0.1	**122**	8	1.1
dodge-3	2e5	9e7	100	<0.01	**1445**	†1764	MO
dpm-10-b	9e3	1e5	22	0.02	**74**	21	TO
obs-8-6	5e2	5e4	90	0.6	**6**	4	1.5
obs-10-6	8e2	3e6	98	<0.01	**5**	412	MO
obs-10-9	1e3	4e8	100	<0.01	**259**	†1661	MO
rov-1000	2e4	4e6	99	0.03	**1402**	†65	TO
uav-work	9e3	2e6	99	<0.01	**113**	55	TO
virus	2e3	7e4	83	0.9	50	**0.8**	TO
rocks-6-4	3e3	7e3	100	34	102	0.2	**0.1**

Experimental results

model	model info			our approach		speedup wrt.	
	$\lvert S_{\mathcal{M}} \rvert$	$\lvert \mathcal{M} \rvert$	SAT %	P/SAT %	time	1-by-1	all-in-1
dodge-2	2e5	3e4	100	0.1	**122**	8	1.1
dodge-3	2e5	9e7	100	<0.01	**1445**	†1764	MO
dpm-10-b	9e3	1e5	22	0.02	**74**	21	TO
obs-8-6	5e2	5e4	90	0.6	**6**	4	1.5
obs-10-6	8e2	3e6	98	<0.01	**5**	412	MO
obs-10-9	1e3	4e8	100	<0.01	**259**	†1661	MO
rov-1000	2e4	4e6	99	0.03	**1402**	†65	TO
uav-work	9e3	2e6	99	<0.01	**113**	55	TO
virus	2e3	7e4	83	0.9	50	**0.8**	TO
rocks-6-4	3e3	7e3	100	34	102	0.2	**0.1**

Main takeaways

Main contributions:

1. We contribute a scalable approach to policy synthesis for sets of MDPs
2. The key technique is a game-based abstraction with abstraction refinement
3. The resulting algorithm finds policies for millions of MDPs and provides a compact representation of them

On MDP similarity

- Our approach works better on families where MDPs are similar
- However, there's no good metric to determine how similar the MDPs are
- Even similar-looking MDPs can have vastly different winning policies
- We argue this approach is beneficial for the community

Code

The input format is an extended version of PRISM modelling language

- straightforward for people from the MDP verification community
- easy to iterate and change the MDP families

Artifact: https://doi.org/10.5281/zenodo.12569976

The presented approach and many more algorithms implemented in tool **PAYNT**

- PAYNT repository: https://github.com/randriu/synthesis

Conclusion

Future work:

- Investigate the robustness problem further
- Incorporate the compact representation of policies (e.g. as decision trees)
- Extend the framework to families of POMDPs

Thank you for attention!