Motion Planning under Uncertainty and Partial Observability

Nils Jansen
Aachen, Dec 7, 2017
joint work with:

Autonomous Systems

- How to affordably build trustworthy systems?
- Use a reliable **system model**
- Operate along uncontrollable agents, in uncertain, or partially observable environments

Formal verification needs to account for these factors.
Autonomous (Cyber-physical) Systems

- Safety specification
- Performance specification
- System model
- Real system
- Formal verification
- Model-based Testing
- Controller synthesis
- Machine learning

Solutions at the interfaces of domains
Help the Mouse

Find the best way to the cheese

While moving mouse discovers exhausting surfaces

Find cost-optimal strategy to get to the cheese

Underlying Model: Markov Decision Process

Obtained via Reinforcement Learning
MDPs for Motion Planning
Reinforcement Learning

- robot
- state, reward
- action

\[
Q: S \times Act \rightarrow \mathbb{R}
\]

- episodic exploration of the state space
- collect information about environment by interaction
- Q-learning approximates optimum
- Q-matrix contains values of taking each action at each state

MDP
Help the Mouse

Find the best way to the cheese

While moving mouse discovers exhausting surfaces

Avoid randomly moving cat

Cost is not known prior to exploring the grid

Repair the obtained strategy intermediately

multi-objective model checking for MDPs

Find safe and cost-optimal strategy to get to the cheese
Story - Safe Reinforcement Learning

nondeterministic behavior inside stochastic environment

Two main ingredients: Model Repair and Reinforcement Learning
Model Repair

- Given a model not satisfying specifications, change the model minimally such that specs are satisfied

- Nonlinear programming: Bartocci et al., TACAS 2011
- PSO based approach: Chen et al., TASE 2013
- Heuristic greedy algorithm: NFM 2014
- Convex optimization: TACAS 2017

- Take strategy and robot MDP
- Repair induced DTMC “preserving” robot MDP
- $\rightarrow$ Repaired strategy
Help the Mouse with an Unknown Environment

- Find the best way to the cheese
- While moving mouse discovers exhausting surfaces
- Avoid randomly moving cat
- Try all safe ways to the cheese (for future mice)

Multi-objective model checking for MDPs

- Find safe and cost-optimal strategy to get to the cheese
- Cost is not known prior to exploring the grid
- Deploy multiple strategies for safe exploration (permissive strategy)
Two main ingredients: Permissive strategies and Reinforcement Learning
Permissive Strategies

\[ s_0 \rightarrow a, \ s_1 \rightarrow d \]
\[ s_0 \rightarrow b, \ s_1 \rightarrow c \]
\[ s_0 \rightarrow b, \ s_1 \rightarrow d \]

\[ s_0 \rightarrow a, \ s_1 \rightarrow c \] unsafe

Find maximal set of safe strategies
Exclude conflicting choices

Exponentially many

Find set of safe strategies

Probability of reaching shall be less than 0.3
Permissive Strategies: SMT encoding

\[ p_s \leq \lambda \quad \text{probability smaller than threshold} \]
\[ \forall s \in S. \quad \bigvee_{a \in \text{Act}(s)} y_{s,a} \quad \text{at least one action per state} \]
\[ \forall s \in T. \quad p_s = 1 \quad \text{probability of target states is one} \]
\[ \forall s \in S. \forall a \in \text{Act}(s). \quad y_{s,a} \to p_s \geq \sum_{s' \in S} P(s, a, s') \cdot p_{s'} \quad \text{probability of each state is assigned the maximum under the permissive strategy} \]

Variables

- \( y_{s,a} \) action is chosen at state
- \( p_s \) probability of state

Solution induces safe permissive strategy

MILP quantifies permissiveness
Story - Safe Reinforcement Learning

- **Soundness**: Correct computation of permissive strategy
- **Completeness**: Optimal safe strategy is computed
- No efficient representation for ‘maximally’ permissive strategy
- **Heuristics**: Significant speedup for costly computation
Further Research

- Less assumptions about the system
- PAC safe learning?
- Richer properties
- Data consistency
Help the Mouse with Partial Observability

Mouse has restricted range of vision

Cat is only observable when near

For mouse, cat is either near or far

Find strategy that induces $Pr_{max}(\neg B U G)$

Belief state: Likelihood of the actual position of the cat

Find safest strategy to get to the cheese

Model checking undecidable

quantitative model checking for POMDPs
Nondeterministic Choices

Mouse has restricted range of vision

Cat is only observable when near

For mouse, cat is either near or far

Find strategy that induces $Pr_{max}(\neg B U G)$

Abstract possible positions into nondeterministic choices

Instead of infinitely many distributions, finite number of choices

Probabilistic Two-Player Game
Game-based Abstraction for POMDPs

- **Merge states** that share an observation into an abstract state
  - probabilistic movements of cat outside of the visible area

- **Introduce choice** over those states
  - position of cat is now determined nondeterministically

- Additional level of nondeterminism: **2-Player game**

- **Worst case** analysis
  - opponent can jump → cat is strengthened, spurious movements
Story - Game-based Abstraction for POMDPs

Safety Specification → POMDP → abstract → PG

Model checking → Optimal PG strategy

[\[p, p + \tau, Pr_{\max}(\neg B U G), u] \]

full observability

Nils Jansen

Radboud University
Refinement

- Usual state splitting as for MDPs is not possible
  - we need a one-to-one correspondence with the POMDP

- Remove spurious movements

- History-based refinement
  - 1-step, multi-step
  - region-based, magnifying lens abstraction

- Refine environment -> increase range of vision
## Experiments

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3 × 3</td>
<td>299</td>
<td>515</td>
<td>739</td>
<td>0.8323</td>
<td>0.063</td>
<td>0.26</td>
<td>400</td>
<td>645</td>
<td>1053</td>
<td>0.8323</td>
<td>0.142</td>
<td>0.036</td>
<td>0.8323</td>
</tr>
<tr>
<td>4 × 4</td>
<td>983</td>
<td>1778</td>
<td>2705</td>
<td>0.9556</td>
<td>0.099</td>
<td>1.81</td>
<td>1348</td>
<td>2198</td>
<td>3897</td>
<td>0.9556</td>
<td>0.353</td>
<td>0.080</td>
<td>0.9556</td>
</tr>
<tr>
<td>5 × 5</td>
<td>2835</td>
<td>5207</td>
<td>8148</td>
<td>0.9882</td>
<td>0.144</td>
<td>175.94</td>
<td>6124</td>
<td>10700</td>
<td>19248</td>
<td>0.9740</td>
<td>0.188</td>
<td>0.649</td>
<td>0.9882</td>
</tr>
<tr>
<td>5 × 6</td>
<td>4390</td>
<td>8126</td>
<td>12890</td>
<td>0.9945</td>
<td>0.228</td>
<td>4215.056</td>
<td>8058</td>
<td>14383</td>
<td>26079</td>
<td>0.9785</td>
<td>0.242</td>
<td>0.518</td>
<td>0.9945</td>
</tr>
<tr>
<td>6 × 6</td>
<td>6705</td>
<td>20086</td>
<td>12501</td>
<td>??</td>
<td>0.377</td>
<td>– MO –</td>
<td>10592</td>
<td>19286</td>
<td>35226</td>
<td>0.9830</td>
<td>0.322</td>
<td>1.872</td>
<td>0.9970</td>
</tr>
<tr>
<td>8 × 8</td>
<td>24973</td>
<td>47413</td>
<td>78338</td>
<td>??</td>
<td>1.735</td>
<td>– MO –</td>
<td>23128</td>
<td>81090</td>
<td>43790</td>
<td>0.9897</td>
<td>0.527</td>
<td>6.349</td>
<td>0.9998</td>
</tr>
<tr>
<td>10 × 10</td>
<td>66297</td>
<td>127829</td>
<td>214094</td>
<td>??</td>
<td>9.086</td>
<td>– MO –</td>
<td>40464</td>
<td>14582</td>
<td>78054</td>
<td>0.9914</td>
<td>0.904</td>
<td>6.882</td>
<td>0.9999</td>
</tr>
</tbody>
</table>

20 × 20
- Time out during model construction –

30 × 30
- Time out during model construction –

40 × 40
- Time out during model construction –

50 × 50
- Time out during model construction –

### MDP

<table>
<thead>
<tr>
<th>States</th>
<th>Choices</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1053</td>
<td>0.8323</td>
<td>0.142</td>
</tr>
<tr>
<td>3897</td>
<td>0.9556</td>
<td>0.353</td>
</tr>
<tr>
<td>19248</td>
<td>0.9740</td>
<td>0.188</td>
</tr>
<tr>
<td>26079</td>
<td>0.9785</td>
<td>0.242</td>
</tr>
<tr>
<td>35226</td>
<td>0.9830</td>
<td>0.322</td>
</tr>
<tr>
<td>43790</td>
<td>0.9897</td>
<td>0.527</td>
</tr>
<tr>
<td>78054</td>
<td>0.9914</td>
<td>0.904</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>States</th>
<th>Choices</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>199144</td>
<td>745362</td>
<td>395774</td>
</tr>
<tr>
<td>477824</td>
<td>1808442</td>
<td>957494</td>
</tr>
<tr>
<td>876504</td>
<td>3334722</td>
<td>1763214</td>
</tr>
<tr>
<td>1395184</td>
<td>5324202</td>
<td>2812934</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>States</th>
<th>Choices</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>280.079</td>
<td>3129.577</td>
<td>– MO –</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>States</th>
<th>Choices</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>– Time out during model construction –</td>
<td>– Time out during model construction –</td>
<td>– Time out during model construction –</td>
</tr>
</tbody>
</table>
Further Research

- automatic refinement
- synthesize sufficiently safe environment
NATIONAL ACADEMY OF SCIENCES

Computers do not deal well with ambiguity. We have to tell them PRECISELY what we want them to do. Thus, computer science requires precise thinking from us.

The challenge of precise thinking attracted me to study computer science.

Moshe Vardi
NAS Member
Human-in-the-loop Synthesis for POMDPs

Turn scenario into an arcade game

Collect data of human playing

From data, compute a strategy

Put human in critical situations

Underlying (family of) POMDPs

Applying strategy yields DTMC, efficient verification

Counterexamples point to critical parts
Story: HiL Synthesis for POMDPs

1. Safety Specification
2. POMDP
3. Training Environment
4. Collect data
5. Gamify
6. Refinement
7. Model Checking
8. Strategy Computation
9. Apply strategy
10. UNSAT
11. SAT
12. Under submission
13. Under construction
Data Augmentation

- Strategy is trained on randomly generated environments
- Training set needs samples until further environments wouldn’t likely change the strategy

- To reduce training set, similar observations are handled similar
(Further) Research

• algorithmic improvement of human strategy (model repair?)
• memory
• in general: how to employ humans’ power of perception to reduce ambiguity for verification/synthesis?
Deterministic vs. Randomization vs. Memory

\[ \Pr_{\text{max}}(\diamondsuit s_7) \]

Choices at observation ‘blue’:

- Choose ‘up’ at each state: prob 2/3 to reach \( s_7 \)  
  memoryless deterministic

- Choose ‘up’ with prob \( 0 < p < 1 \) and ‘down’ with prob \( 1 - p = \frac{2}{3} + \frac{1}{3}p < 1 \)  
  memoryless randomized

- Choose ‘up’ if predecessor is ‘yellow’. Otherwise, choose ‘up’ if ‘blue’ observed even number of times, ‘down’ otherwise: prob 1  
  deterministic with memory
Randomized Strategies for POMDPs

- Can often trade off memory
- Still hard to compute: NP-hard, SQRT-SUM-hard, in PSPACE
- Encode finite memory:
Randomized Strategies using Parameter Synthesis for Markov Chains

\[ s_0 \xrightarrow{a_1} s_1 \xrightarrow{1} a_1 \xrightarrow{1} s_2 \xrightarrow{a_2} 0.5 \xrightarrow{1} s_1 \xrightarrow{a_1} s_0 \]

\[ s_0 \xrightarrow{a_2} 0.5 \xrightarrow{0.5} s_2 \xrightarrow{a_2} 1 \xrightarrow{1} s_3 \xrightarrow{a_1} s_0 \]

\[ s_0 \xrightarrow{a_3} 0.5 \xrightarrow{0.5} s_2 \xrightarrow{a_2} 1 \xrightarrow{1} s_3 \xrightarrow{a_1} s_0 \]
Further Research

- Efficient methods for randomized strategies for POMDPs
- Permissive strategies for POMDPs (and applications)
- POMDP active learning
- Utilization of convex optimization
Conclusion

- Safe motion planning under uncertainty and partial observability
- Randomization vs. Memory
- Integrate humans into verification and learning
- More Efficient Model Repair