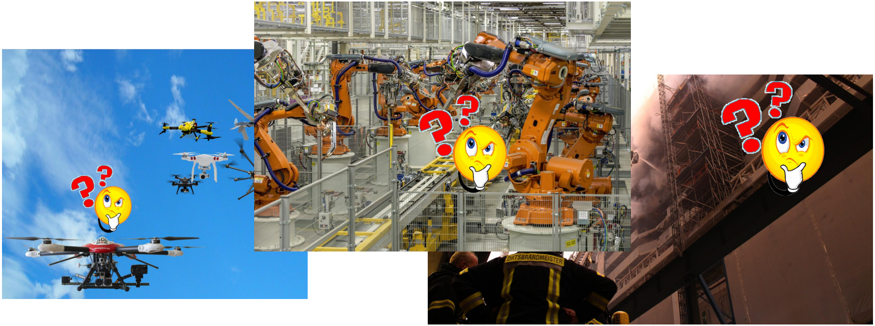
Debugging a Policy: A Framework for Automatic Action Policy Testing

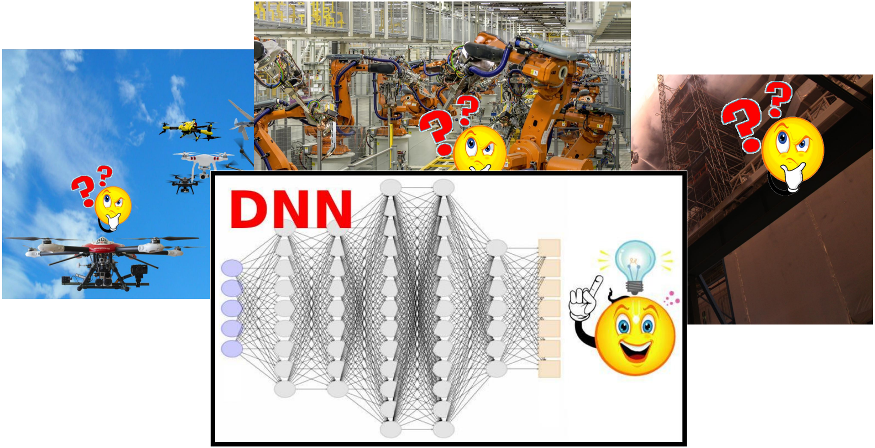
Marcel Steinmetz, Timo P. Gros, Philippe Heim,
Daniel Höller, Jörg Hoffmann

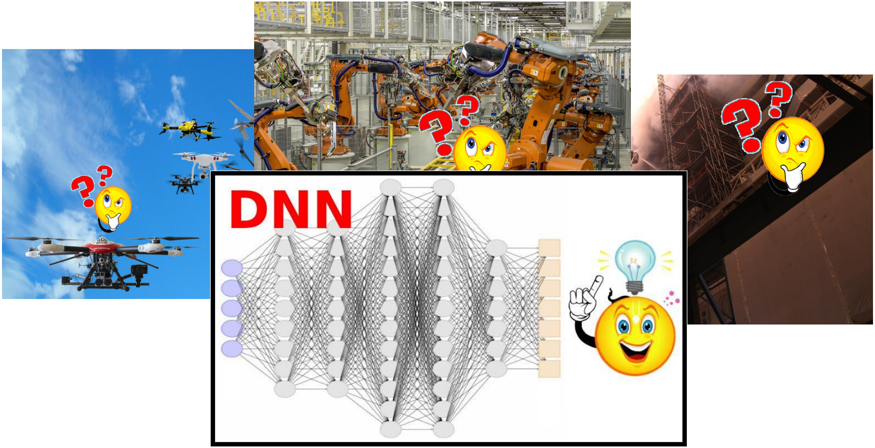
SAARLAND UNIVERSITY
COMPUTER SCIENCE

July 5, 2021

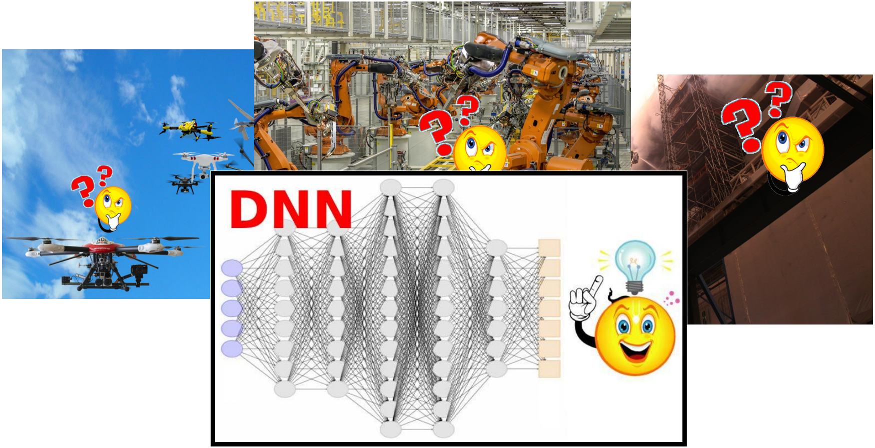
ORTSBRANDMEISTER


ORTSBRANDMEISTER


DNN



e.g. PRL, e.g. [Toyer *et al.* (2018); Issakkimuthu *et al.* (2018); Groshev *et al.* (2018); Garg *et al.* (2019); Rivlin *et al.* (2020); Toyer *et al.* (2020)]



e.g. PRL, e.g. [Toyer *et al.* (2018); Issakkimuthu *et al.* (2018); Groshev *et al.* (2018); Garg *et al.* (2019); Rivlin *et al.* (2020); Toyer *et al.* (2020)]

But what about trust in a learned neural action policy?

- Explanation, e.g. [Chakraborti *et al.* (2019); Agogino *et al.* (2019)]
- Visualization, e.g. [Gros *et al.* (2020)]
- Shielding, e.g. [Könighofer *et al.* (2017); Alshiekh *et al.* (2017); Fulton and Platzer (2018)]
- Verification, e.g. [Katz *et al.* (2017); Gehr *et al.* (2018); Akintunde *et al.* (2018); Vinzent and Hoffmann (2021)]

- Explanation, e.g. [Chakraborti *et al.* (2019); Agogino *et al.* (2019)]
- Visualization, e.g. [Gros *et al.* (2020)]
- Shielding, e.g. [Könighofer *et al.* (2017); Alshiekh *et al.* (2017); Fulton and Platzer (2018)]
- Verification, e.g. [Katz *et al.* (2017); Gehr *et al.* (2018); Akintunde *et al.* (2018); Vinzent and Hoffmann (2021)]
- Testing

- Explanation, e.g. [Chakraborti *et al.* (2019); Agogino *et al.* (2019)]
- Visualization, e.g. [Gros *et al.* (2020)]
- Shielding, e.g. [Könighofer *et al.* (2017); Alshiekh *et al.* (2017); Fulton and Platzer (2018)]
- Verification, e.g. [Katz *et al.* (2017); Gehr *et al.* (2018); Akintunde *et al.* (2018); Vinzent and Hoffmann (2021)]
- Testing

  e.g. [Julian *et al.* (2020)] in model-free (blackbox environment) setting for image-based NN controllers.

  But in ICAPS community? None that I know of.

- Explanation, e.g. [Chakraborti *et al.* (2019); Agogino *et al.* (2019)]
- Visualization, e.g. [Gros *et al.* (2020)]
- Shielding, e.g. [Könighofer *et al.* (2017); Alshiekh *et al.* (2017); Fulton and Platzer (2018)]
- Verification, e.g. [Katz *et al.* (2017); Gehr *et al.* (2018); Akintunde *et al.* (2018); Vinzent and Hoffmann (2021)]
- Testing

  e.g. [Julian *et al.* (2020)] in model-free (blackbox environment) setting for image-based NN controllers.

  But in ICAPS community? None that I know of.

→ "Is this planning?"

- Explanation, e.g. [Chakraborti *et al.* (2019); Agogino *et al.* (2019)]
- Visualization, e.g. [Gros *et al.* (2020)]
- Shielding, e.g. [Könighofer *et al.* (2017); Alshiekh *et al.* (2017); Fulton and Platzer (2018)]
- Verification, e.g. [Katz *et al.* (2017); Gehr *et al.* (2018); Akintunde *et al.* (2018); Vinzent and Hoffmann (2021)]
- Testing

  e.g. [Julian *et al.* (2020)] in model-free (blackbox environment) setting for image-based NN controllers.

  But in ICAPS community? None that I know of.

→ "Is this planning?" Some of it surely is (you'll see).

- Explanation, e.g. [Chakraborti *et al.* (2019); Agogino *et al.* (2019)]
- Visualization, e.g. [Gros *et al.* (2020)]
- Shielding, e.g. [Könighofer *et al.* (2017); Alshiekh *et al.* (2017); Fulton and Platzer (2018)]
- Verification, e.g. [Katz *et al.* (2017); Gehr *et al.* (2018); Akintunde *et al.* (2018); Vinzent and Hoffmann (2021)]
- Testing

  e.g. [Julian *et al.* (2020)] in model-free (blackbox environment) setting for image-based NN controllers.

  But in ICAPS community? None that I know of.

→ "Is this planning?" Some of it surely is (you'll see).

→ "Is this PRL?"

- Explanation, e.g. [Chakraborti *et al.* (2019); Agogino *et al.* (2019)]
- Visualization, e.g. [Gros *et al.* (2020)]
- Shielding, e.g. [Könighofer *et al.* (2017); Alshiekh *et al.* (2017); Fulton and Platzer (2018)]
- Verification, e.g. [Katz *et al.* (2017); Gehr *et al.* (2018); Akintunde *et al.* (2018); Vinzent and Hoffmann (2021)]
- Testing

  e.g. [Julian *et al.* (2020)] in model-free (blackbox environment) setting for image-based NN controllers.

  But in ICAPS community? None that I know of.

→ "Is this planning?" Some of it surely is (you'll see).

→ "Is this PRL?" You tell me :-)

- Explanation, e.g. [Chakraborti *et al.* (2019); Agogino *et al.* (2019)]
- Visualization, e.g. [Gros *et al.* (2020)]
- Shielding, e.g. [Könighofer *et al.* (2017); Alshiekh *et al.* (2017); Fulton and Platzer (2018)]
- Verification, e.g. [Katz *et al.* (2017); Gehr *et al.* (2018); Akintunde *et al.* (2018); Vinzent and Hoffmann (2021)]
- Testing

  e.g. [Julian *et al.* (2020)] in model-free (blackbox environment) setting for image-based NN controllers.

  But in ICAPS community? None that I know of.

→ "Is this planning?" Some of it surely is (you'll see).

→ "Is this PRL?" You tell me :-) New workshop Trusted AIP?

Agenda

1. Context & Notation
2. What is a “Bug”?
3. Bug Confirmation
4. Outlook

Agenda

1. Context & Notation
2. What is a “Bug”?
3. Bug Confirmation
4. Outlook

Planning Models Addressed

Everything.

Planning Models Addressed

Everything.

- Classical planning
- Contingent planning
- Oversubscription planning
- Discounted-reward/MaxProb MDPs
- ⟨InsertYourFavoriteModelHere⟩

Planning Models Addressed

Everything.

- Classical planning
- Contingent planning
- Oversubscription planning
- Discounted-reward/MaxProb MDPs
- ⟨InsertYourFavoriteModelHere⟩

→ All we assume is that learning a policy $\pi : states \mapsto actions$ makes sense, and that a value function $V^\pi : states \mapsto \mathbb{R}$ can be defined which captures the quality of π run on s.

Generic (Cross-Planning-Model) Notation

Qualitative value function:

$$V^\pi(s) := \begin{cases} 0 & \text{no run of } \pi \text{ on } s \text{ reaches the goal} \\ 0.5 & \text{some runs of } \pi \text{ on } s \text{ reach the goal} \\ 1 & \text{all runs of } \pi \text{ on } s \text{ reach the goal} \end{cases}$$

Generic (Cross-Planning-Model) Notation

Qualitative value function:

$$V^\pi(s) := \begin{cases} 0 & \text{no run of } \pi \text{ on } s \text{ reaches the goal} \\ 0.5 & \text{some runs of } \pi \text{ on } s \text{ reach the goal} \\ 1 & \text{all runs of } \pi \text{ on } s \text{ reach the goal} \end{cases}$$

Optimal value function:

$$V^*(s) := \begin{cases} \min_\pi V^\pi(s) & \text{objective is minimization} \\ \max_\pi V^\pi(s) & \text{objective is maximization} \end{cases}$$

Generic (Cross-Planning-Model) Notation

Qualitative value function:

$$V^\pi(s) := \begin{cases} 0 & \text{no run of } \pi \text{ on } s \text{ reaches the goal} \\ 0.5 & \text{some runs of } \pi \text{ on } s \text{ reach the goal} \\ 1 & \text{all runs of } \pi \text{ on } s \text{ reach the goal} \end{cases}$$

Optimal value function:

$$V^*(s) := \begin{cases} \min_\pi V^\pi(s) & \text{objective is minimization} \\ \max_\pi V^\pi(s) & \text{objective is maximization} \end{cases}$$

Generic "is better than" notation: (for the record)

$$V(s') \prec V(s) :iff \begin{cases} V(s') < V(s) & \text{objective is minimization} \\ V(s') > V(s) & \text{objective is maximization} \end{cases}$$

Agenda

1 Context & Notation

2 What is a “Bug”?

3 Bug Confirmation

4 Outlook

Definition: Bug

Definition (Bug)

A state s is a **bug** in policy π if $\Delta := |V^{\pi}(s) - V^{*}(s)| > 0$.

Definition: Bug

Definition (Bug)

A state s is a **bug** in policy π if $\Delta := |V^{\pi}(s) - V^{*}(s)| > 0$.

- Classical planning, qualitative: $\Delta = 1 \equiv \pi$ does not reach the goal on solvable state.
- Contingent planning, qualitative: $\Delta = 0.5 \equiv \pi$ does not reach the goal on some solvable states.
- Oversubscription planning/rewards: Δ rewards less than possible.
- MaxProb MDPs: reach goal with Δ less probability than possible.

Definition: Bug

Definition (Bug)

A state s is a **bug** in policy π if $\Delta := |V^{\pi}(s) - V^{*}(s)| > 0$.

- Classical planning, qualitative: $\Delta = 1 \equiv \pi$ does not reach the goal on solvable state.
- Contingent planning, qualitative: $\Delta = 0.5 \equiv \pi$ does not reach the goal on some solvable states.
- Oversubscription planning/rewards: Δ rewards less than possible.
- MaxProb MDPs: reach goal with Δ less probability than possible.

Notes:

- Bug-free $\Rightarrow$ optimal.

Definition: Bug

Definition (Bug)

A state s is a **bug** in policy π if $\Delta := |V^\pi(s) - V^*(s)| > 0$.

- Classical planning, qualitative: $\Delta = 1 \equiv \pi$ does not reach the goal on solvable state.
- Contingent planning, qualitative: $\Delta = 0.5 \equiv \pi$ does not reach the goal on some solvable states.
- Oversubscription planning/rewards: Δ rewards less than possible.
- MaxProb MDPs: reach goal with Δ less probability than possible.

Notes:

- Bug-free $\Rightarrow$ optimal.
- This would *not* be the case for bug := action starting optimal policy.

Definition: Fuzzing Bug

Definition (Fuzzing Bug)

A state s' is a **fuzzing-bug** relative to s if
$\Delta := |V^\pi(s') - V^*(s')| - |V^\pi(s) - V^*(s)| > 0$.

Definition: Fuzzing Bug

Definition (Fuzzing Bug)

A state s' is a **fuzzing-bug** relative to s if
$\Delta := |V^\pi(s') - V^*(s')| - |V^\pi(s) - V^*(s)| > 0$.

Observe: (trivial)

1. If s' is a fuzzing-bug relative to some s, then s' is a bug.

Definition: Fuzzing Bug

Definition (Fuzzing Bug)

A state s' is a **fuzzing-bug** relative to s if
$\Delta := |V^\pi(s') - V^*(s')| - |V^\pi(s) - V^*(s)| > 0$.

Observe: (trivial)

1. If s' is a fuzzing-bug relative to some s, then s' is a bug.
2. Every bug s' with non-minimal optimality gap $|V^\pi(s) - V^*(s)|$ is a fuzzing-bug relative to some s.

Definition: Fuzzing Bug

Definition (Fuzzing Bug)

A state s' is a **fuzzing-bug** relative to s if
$\Delta := |V^\pi(s') - V^*(s')| - |V^\pi(s) - V^*(s)| > 0$.

Observe: (trivial)

1. If s' is a fuzzing-bug relative to some s, then s' is a bug.
2. Every bug s' with non-minimal optimality gap $|V^\pi(s) - V^*(s)|$ is a fuzzing-bug relative to some s.

Why?

- Natural situation in fuzzing algorithms.
- 2. does not hold under restrictions on reachability of s' from s by such algorithms.

Definition: Fuzzing Bug

Definition (Fuzzing Bug)

A state s' is a **fuzzing-bug** relative to s if
$\Delta := |V^\pi(s') - V^*(s')| - |V^\pi(s) - V^*(s)| > 0$.

Observe: (trivial)

1. If s' is a fuzzing-bug relative to some s, then s' is a bug.
2. Every bug s' with non-minimal optimality gap $|V^\pi(s) - V^*(s)|$ is a fuzzing-bug relative to some s.

Why?

- Natural situation in fuzzing algorithms.
- 2. does not hold under restrictions on reachability of s' from s by such algorithms.
- Can this definition help in bug confirmation?

Agenda

1. Context & Notation
2. What is a “Bug”?
3. Bug Confirmation
4. Outlook

Bug Confirmation

Definition (Bug Confirmation)

Bug confirmation is the problem of deciding, given a state s, whether or not s is a bug.

→ Obviously, solving this problem exactly involves solving s optimally. (I told you some of it is planning, didn't I?)

Bug Confirmation

Definition (Bug Confirmation)

Bug confirmation is the problem of deciding, given a state s, whether or not s is a bug.

→ Obviously, solving this problem exactly involves solving s optimally. (I told you some of it is planning, didn't I?)

So we approximate . . . [Patrik Haslum, AIPS'00]

Bug Confirmation

Definition (Bug Confirmation)

Bug confirmation is the problem of deciding, given a state s, whether or not s is a bug.

→ Obviously, solving this problem exactly involves solving s optimally. (I told you some of it is planning, didn't I?)

So we approximate . . . [Patrik Haslum, AIPS'00]

With $H_* \succeq V^*(s)$ and $h_\pi(s) \preceq V^\pi(s)$ pessimistic approximation of V^* and optimistic approximation of V^π respectively:

Proposition (Bug Confirmation)

Say that $V^*(s) \preceq H_*(s)$ and $h_\pi(s) \preceq V^\pi(s)$. Say that $h_\pi(s) \succeq V^*(s)$ and $H_*(s) \preceq V^\pi(s)$. Then $|h_\pi(s) - H_*(s)| \leq |V^\pi(s) - V^*(s)|$.

Bug Confirmation

Definition (Bug Confirmation)

Bug confirmation is the problem of deciding, given a state s, whether or not s is a bug.

→ Obviously, solving this problem exactly involves solving s optimally. (I told you some of it is planning, didn't I?)

So we approximate . . . [Patrik Haslum, AIPS'00]

With $H_* \succeq V^*(s)$ and $h_\pi(s) \preceq V^\pi(s)$ pessimistic approximation of V^* and optimistic approximation of V^π respectively:

Proposition (Bug Confirmation)

Say that $V^*(s) \preceq H_*(s)$ *and* $h_\pi(s) \preceq V^\pi(s)$. *Say that* $h_\pi(s) \succeq V^*(s)$ *and* $H_*(s) \preceq V^\pi(s)$. *Then* $|h_\pi(s) - H_*(s)| \leq |V^\pi(s) - V^*(s)|$.

→ Boils down to: "evaluate $V^\pi(s)$, and try to find a better policy for s".

Bug Confirmation, ctd.

Proposition (Fuzzing Bug Confirmation)

(a) If $I_*(s) \cap I_*(s') = \emptyset$, s' *is a fuzzing-bug relative to* s *if* $H_*(s') \prec h_*(s)$ *and either* $V^\pi(s') \succeq V^\pi(s)$ *or* $|V^\pi(s') - V^\pi(s)| < |H_*(s') - h_*(s)|$.
(b) s' *is a fuzzing-bug relative to* s *if* $V^\pi(s') \succeq V^\pi(s)$ *and* $|V^\pi(s') - V^\pi(s)| > U_*(s, s')$.

Bug Confirmation, ctd.

Proposition (Fuzzing Bug Confirmation)

(a) If $I_*(s) \cap I_*(s') = \emptyset$, s' is a fuzzing-bug relative to s if $H_*(s') \prec h_*(s)$ and either $V^\pi(s') \succeq V^\pi(s)$ or $|V^\pi(s') - V^\pi(s)| < |H_*(s') - h_*(s)|$.
(b) s' is a fuzzing-bug relative to s if $V^\pi(s') \succeq V^\pi(s)$ and $|V^\pi(s') - V^\pi(s)| > U_*(s, s')$.

Theorem (It's All in Vain)

Boils down to "evaluate $V^\pi(s)$, and try to find a better policy for s".

Bug Confirmation, ctd.

Proposition (Fuzzing Bug Confirmation)

(a) If $I_*(s) \cap I_*(s') = \emptyset$, s' is a fuzzing-bug relative to s if $H_*(s') \prec h_*(s)$ and either $V^\pi(s') \succeq V^\pi(s)$ or $|V^\pi(s') - V^\pi(s)| < |H_*(s') - h_*(s)|$.
(b) s' is a fuzzing-bug relative to s if $V^\pi(s') \succeq V^\pi(s)$ and $|V^\pi(s') - V^\pi(s)| > U_*(s, s')$.

Theorem (It's All in Vain)

Boils down to "evaluate $V^\pi(s)$, and try to find a better policy for s".

So what?

- Many special cases with "V^* oracle" (e.g. all states known to be solvable; enough time during at testing to run symbolic planner).
- In general case, plug in plan-quality improvement algorithms [Bäckström (1998); Do and Kambhampati (2003); Nakhost and Müller (2010); Siddiqui and Haslum (2015)].

Agenda

1 Context & Notation

2 What is a “Bug”?

3 Bug Confirmation

4 Outlook

Outlook

Ok, so now let's actually do this!

Outlook

Ok, so now let's actually do this!

- Develop fuzzing methods!
- Develop bug confirmation paradigms (metamorphosic testing etc)!
- See what all this does in all your favorite planning and learning scenarios!

Last Slide

Thanks for listening.

Questions?

References I

Adrian Agogino, Ritchie Lee, and Dimitra Giannakopoulou. Challenges of explaining control. In *2nd ICAPS Workshop on Explainable Planning (XAIP'19)*, 2019.

Michael Akintunde, Alessio Lomuscio, Lalit Maganti, and Edoardo Pirovano. Reachability analysis for neural agent-environment systems. In *16th International Conference on Principles of Knowledge Representation and Reasoning (KR'18)*, pages 184–193, 2018.

Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. Safe reinforcement learning via shielding. *CoRR*, abs/1708.08611, 2017.

Christer Bäckström. Computational aspects of reordering plans. *Journal of Artificial Intelligence Research*, 9:99–137, 1998.

Tathagata Chakraborti, Anagha Kulkarni, Sarath Sreedharan, David E. Smith, and Subbarao Kambhampati. Explicability? legibility? predictability? transparency? privacy? security? the emerging landscape of interpretable agent behavior. In *Proceedings of the 29th International Conference on Automated Planning and Scheduling (ICAPS'19)*, pages 86–96. AAAI Press, 2019.

References II

Minh. B. Do and Subbarao Kambhampati. Improving the temporal flexibility of position constrained metric temporal plans. In *Proceedings of the 13th International Conference on Automated Planning and Scheduling (ICAPS'03)*, pages 42–51, 2003.

Nathan Fulton and Andreé Platzer. Safe reinforcement learning via formal methods: Toward safe control through proof and learning. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI'18)*, pages 6485–6492. AAAI Press, 2018.

Sankalp Garg, Aniket Bajpai, and Mausam. Size independent neural transfer for RDDL planning. In *Proceedings of the 29th International Conference on Automated Planning and Scheduling (ICAPS'19)*, pages 631–636. AAAI Press, 2019.

Timon Gehr, Matthew Mirman, Dana Drachsler-Cohen, Petar Tsankov, Swarat Chaudhuri, and Martin T. Vechev. AI2: Safety and robustness certification of neural networks with abstract interpretation. In *Proceedings of the IEEE Symposium on Security and Privacy 2018*, pages 3–18. IEEE Computer Society, 2018.

Timo P. Gros, David Groß, Stefan Gumhold, Jrg Hoffmann, Michaela Klauck, and Marcel Steinmetz. TraceVis: Towards visualization for deep statistical model checking. In *Proceedings of the 9th International Symposium On Leveraging Applications of Formal Methods, Verification and Validation (ISoLA'20)*, 2020.

References III

Edward Groshev, Maxwell Goldstein, Aviv Tamar, Siddharth Srivastava, and Pieter Abbeel. Learning generalized reactive policies using deep neural networks. In *Proceedings of the 28th International Conference on Automated Planning and Scheduling (ICAPS'18)*, pages 408–416. AAAI Press, 2018.

Murugeswari Issakkimuthu, Alan Fern, and Prasad Tadepalli. Training deep reactive policies for probabilistic planning problems. In *Proceedings of the 28th International Conference on Automated Planning and Scheduling (ICAPS'18)*, pages 422–430. AAAI Press, 2018.

Kyle D. Julian, Ritchie Lee, and Mykel J. Kochenderfer. Validation of image-based neural network controllers through adaptive stress testing. In *23rd IEEE International Conference on Intelligent Transportation Systems (ITSC'20)*, pages 1–7, 2020.

Guy Katz, Clark W. Barrett, David L. Dill, Kyle Julian, and Mykel J. Kochenderfer. Reluplex: An efficient SMT solver for verifying deep neural networks. In *Proceedings of the 29th International Conference Computer Aided Verification (CAV'17)*, volume 10426 of *Lecture Notes in Computer Science*, pages 97–117. Springer, 2017.

References IV

Bettina Könighofer, Mohammed Alshiekh, Roderick Bloem, Laura Humphrey, Robert Könighofer, Ufuk Topcu, and Chao Wang. Shield synthesis. *Formal Methods in System Design*, 51(2):332–361, 2017.

Hootan Nakhost and Martin Müller. Action elimination and plan neighborhood graph search: Two algorithms for plan improvement. In *Proceedings of the 20th International Conference on Automated Planning and Scheduling (ICAPS'10)*, pages 121–128, 2010.

Or Rivlin, Tamir Hazan, and Erez Karpas. Generalized planning with deep reinforcement learning. In *ICAPS 2020 Workshop on Bridging the Gap Between AI Planning and Reinforcement Learning (PRL)*, pages 16–24, 2020.

Fazlul Hasan Siddiqui and Patrik Haslum. Continuing plan quality optimisation. *Journal of Artificial Intelligence Research*, 54:369–435, 2015.

Sam Toyer, Felipe Trevizan, Sylvie Thiebaux, and Lexing Xie. Action schema networks: Generalised policies with deep learning. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI'18)*, 2018.

Sam Toyer, Sylvie Thiébaux, Felipe W. Trevizan, and Lexing Xie. Asnets: Deep learning for generalised planning. *Journal of Artificial Intelligence Research*, 68:1–68, 2020.

References V

Marcel Vinzent and Jörg Hoffmann. Neural network action policy verification via predicate abstraction. In *Proceedings of the ICAPS Workshop on Bridging the Gap Between AI Planning and Reinforcement Learning (PRL'21)*, 2021.