

Explaining Symbolic Action Policies using Facet Reasoning

Jennifer Santos, Johannes K. Fichte and Daniel Gnad

Linköping University, Heidelberg University

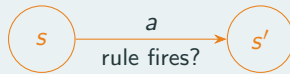
June 28, 2026

Symbolic Action Policies

What is a policy?

Given a policy and a planning problem:

- **Input:** a state + a candidate action
- **Output:** a verdict
 - Does this action lead towards the goal?



- **In this work:** transition classifiers (Musayev et al. 2025), built on general policies (Frances, Bonet, and Geffner 2021).
- Policies are **symbolic** and in principle human-readable.

Readable, but understandable?

```
(:policy
(:Booleans
(b43 "b_empty(c_and(c_primitive(holding,0),c_some(r_primitive(on_g,0,1),c_primitive(clear,0))))")
(b138 "b_empty(c_and(c_primitive(holding,0),c_not(c_all(r_primitive(on_g,0,1),c_primitive(clear,0))))")
(b1132 "b_empty(c_and(c_primitive(holding,0),c_not(c_all(r_primitive(on_g,0,1),c_equal(r_primitive(on,0,1),r_primitive(on_g,0,1))))))")
(b1157 "b_empty(c_and(c_primitive(holding,0),c_some(r_primitive(on_g,0,1),c_some(r_transitive_closure(r_primitive(on_g,0,1),c_primitive(clear,0))))))")
(b1205 "b_empty(c_and(c_primitive(holding,0),c_all(r_transitive_closure(r_primitive(on_g,0,1),c_equal(r_primitive(on,0,1),r_primitive(on_g,0,1))))))")

(:numericals
(n1886 "n_count(c_and(c_primitive(on-table_g,0),c_all(r_inverse(r_primitive(on,0,1),c_equal(r_primitive(on,0,1),r_primitive(on_g,0,1))))))")
(n2637 "n_count(c_and(c_equal(r_primitive(on,0,1),r_primitive(on_g,0,1)),c_all(r_inverse(r_primitive(on_g,0,1),c_primitive(holding,0))))))")
(n229 "n_count(c_and(c_primitive(clear_g,0),c_some(r_transitive_closure(r_primitive(on_g,0,1),c_primitive(clear,0))))))")
(n785 "n_count(c_and(c_not(c_primitive(holding,0)),c_some(r_transitive_closure(r_primitive(on_g,0,1),c_primitive(clear,0))))))")
(n1143 "n_count(c_some(r_transitive_closure(r_primitive(on,0,1),c_and(c_not(c_primitive(on-table,0)),c_primitive(on-table_g,0))))))")
(n2431 "n_count(c_and(c_not(c_primitive(on-table,0)),c_not(c_some(r_transitive_closure(r_primitive(on_g,0,1),c_primitive(on-table,0))))))")
(n305 "n_count(c_some(r_transitive_closure(r_primitive(on,0,1),c_equal(r_primitive(on,0,1),r_primitive(on_g,0,1))))))")
(n1358 "n_count(c_all(r_transitive_closure(r_primitive(on,0,1),c_all(r_inverse(r_primitive(on_g,0,1),c_primitive(clear_g,0))))))")
(n3445 "n_count(c_some(r_transitive_closure(r_primitive(on,0,1),c_some(r_primitive(on_g,0,1),c_some(r_primitive(on_g,0,1),c_primitive(on-table_g,0))))))")
(n3534 "n_count(c_some(r_transitive_closure(r_primitive(on_g,0,1),c_all(r_primitive(on,0,1),c_some(r_primitive(on,0,1),c_primitive(on-table_g,0))))))")
(n3563 "n_count(c_some(r_inverse(r_transitive_closure(r_primitive(on,0,1))),c_all(r_transitive_closure(r_primitive(on_g,0,1),c_primitive(clear,0))))))")
(:rule (:conditions ) (:effects (:e_n_inc n9) (:e_n_dec n122)))

(:rule (:conditions ) (:effects (:e_n_dec_bot n9) (:e_b_pos b43) (:e_n_inc n2637) (:e_n_dec n785)))
(:rule (:conditions ) (:effects (:e_n_dec_bot n9) (:e_b_neg b43) (:e_n_inc n2637) (:e_n_dec n785) (:e_b_pos b1205)))
(:rule (:conditions ) (:effects (:e_n_dec_bot n9) (:e_n_dec_bot n1886) (:e_n_inc n2637) (:e_n_dec_bot n229) (:e_n_inc_bot n785)))
(:rule (:conditions (:c_b_neg b138) (:c_b_pos b1132)) (:effects (:e_n_dec_bot n9) (:e_n_dec_bot n2637) (:e_n_inc_bot n1143) (:e_n_inc_bot n2431) (:e_n_inc n305)))
(:rule (:conditions (:c_b_neg b1132)) (:effects (:e_n_dec_bot n9) (:e_n_dec_bot n2637) (:e_n_inc_bot n2431) (:e_n_dec n1358) (:e_n_inc_bot n3534) (:e_n_dec_bot n3563)))
)
```

(Musayev et al. 2025)

- Ideally a learned policy is perfect — learning is not perfect.
- A policy might fail or **prune necessary plans**.
- When it does: which rule is responsible, and why?

Previous methods

- Relied on **manual interpretation** of hand-crafted symbolic policies.
- For learned symbolic policies, this is **practically impossible**.

1. A framework to debug, analyze and explain symbolic action policies.
2. An **automatic** analysis that pinpoints which rules break the policy.
3. An **interactive** analysis where the user can inspect how individual rules affect the plan space.
4. We can answer questions such as:
 - Why does a policy fail?
 - Why does a working policy prune plans?



- The remaining plan space **is** the consequence of the policy.
- Tells how restrictive the policy is.
- We measure this with **facets**.

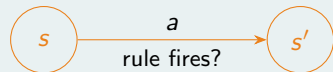
Background

Policies in Classical Planning

Planning problem $\Pi = \langle D, I \rangle$: find a plan from s_0 to G (PDDL / SAS⁺).

Policies

- A set of **rules** over **description-logic (DL) features**
- **Domain-general**: generalizes across instances



Transition classes (Musayev et al. 2025):

good

decreases goal distance

bad

does not decrease it

unsolvable

leads to an unsolvable state

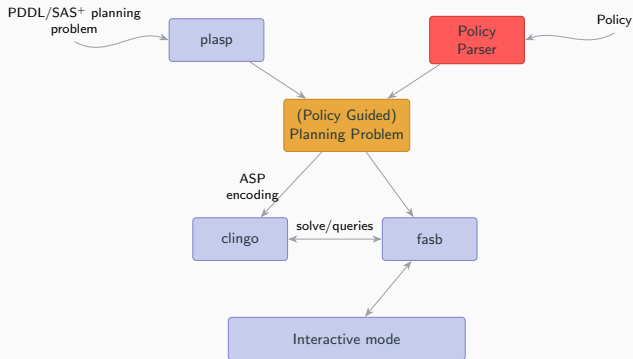
Faceted reasoning

- A **facet** is an atom that is present in *some* solutions but *not in all* (Fichte, Gaggl, and Rusovac 2022).
 - **Enforce** → keep solutions where it holds.
 - **Prohibit** → keep where it is false.

Why it scales:

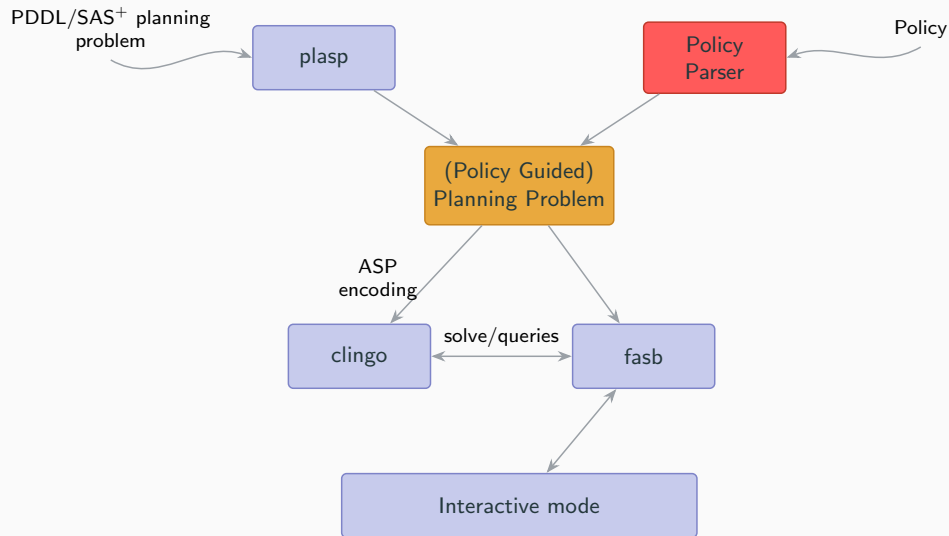
Facet reasoning is in **NP**, which is much cheaper than *counting* or *enumerating* ($\#P$) plans (Speck et al. 2025).

- An iterative and interactive planning tool (Gnad et al. 2025).
- A facet is an action that occurs in some plans but not in all.
- Uses facets to explore the plan space of a planning task.
- Some features:
 - Count facets/plans
 - Enumerate facets/plans
 - Enforce/prohibit facets



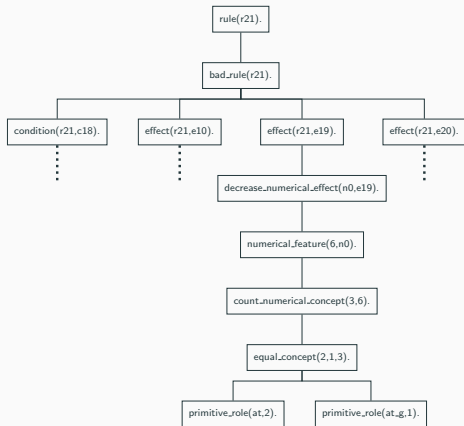
Method: Encoding Policies in ASP

Encode Policies



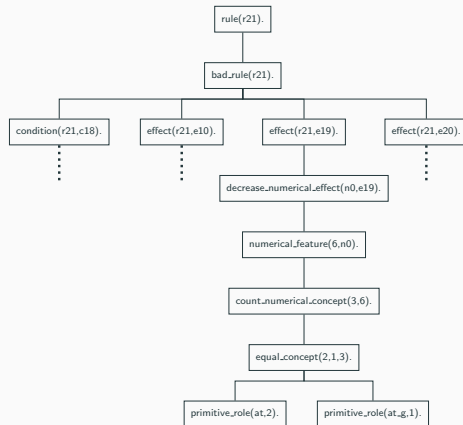
Translation to ASP

```
(:rule
  (:conditions (:c_b_pos b3))
  (:effects (:e_n_dec n0)
            (:e_n_inc_bot n1)
            (:e_b_pos b4)))
(Boolean:
  (b3 "b_empty(c_some(
    r_primitive(at_g,0,1),
    c_primitive(at_robot,0)))")
  (b4 "b_empty(c_not(c_all(
    r_primitive(at_g,0,1),
    c_primitive(clear,0)))")
  (numerical:
    (n0 "n_count(c_equal(
      r_primitive(at,0,1),
      r_primitive(at_g,0,1)))")
    (n1 "n_count(c_some(
      r_primitive(at_g,0,1),
      c_primitive(clear,0)))")
```



Translation to ASP

```
(:rule
  (:conditions (:c_b_pos b3))
  (:effects (:e_n_dec n0)
            (:e_n_inc_bot n1)
            (:e_b_pos b4)))
(Boolean:
  (b3 "b_empty(c_some(
    r_primitive(at_g,0,1),
    c_primitive(at-robot,0)))")
  (b4 "b_empty(c_not(c_all(
    r_primitive(at_g,0,1),
    c_primitive(clear,0))))")
  (numerical:
    (n0 "n_count(c_equal(
      r_primitive(at,0,1),
      r_primitive(at_g,0,1)))")
    (n1 "n_count(c_some(
      r_primitive(at_g,0,1),
      c_primitive(clear,0)))")
```

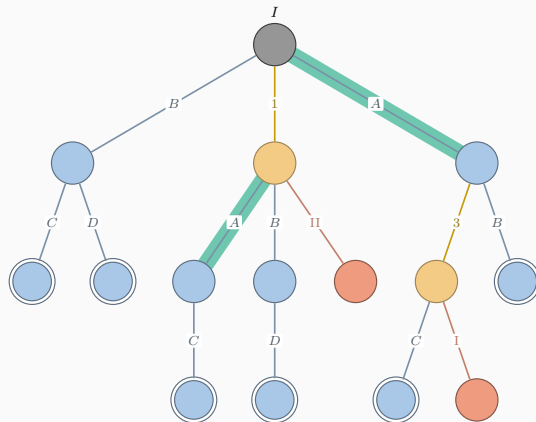


Each feature is evaluated by domain-independent rules, e.g.:

```
effect_satisfied(E,T,T1) :- decrease_numerical_effect(N,E),
    T1 = T+1, numerical(N,V0,T), numerical(N,V1,T1), V0 > V1.
```

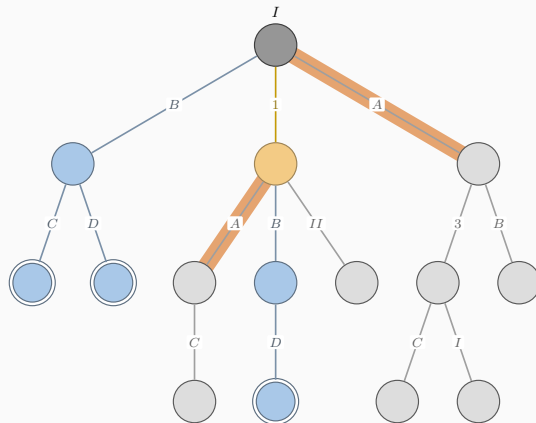
Policy Rules as Facets

- In PlanPilot, a facet is an optional action in the plan space.
- In our framework, rule activation is also treated as a facet.
- **Enforce:** keep plans where **rule A** fires.



Policy Rules as Facets

- In PlanPilot, a facet is an optional action in the plan space.
- In our framework, rule activation is also treated as a facet.
- **Enforce**: keep plans where **rule A** fires.
- **Prohibit**: keep plans where **rule A** never fires.



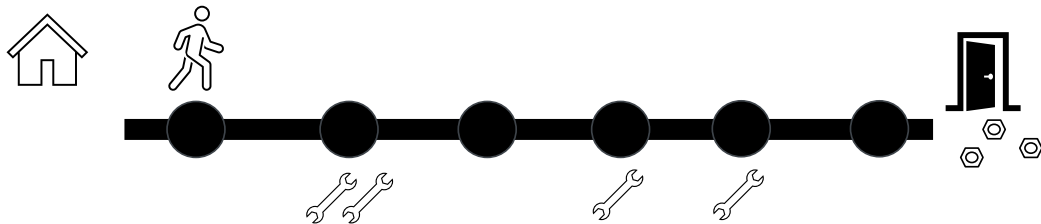
Framework

Configuration

Name	Good rules	Bad rules	Unsolvable rules	Optimization mode
ASP				
ASP _g	✓			
ASP _{g,b}	✓	✓		
ASP _{g, $\tilde{b}$}	✓	✓		✓
ASP _{g,b,u}	✓	✓	✓	

Case Study

Case Study: Spanner



Case Study: Spanner

Instance	$\sum f_{\text{ASP}}$	$\sum f_{\text{ASP}_{g,b}}$
easy p05	22	12
easy p10	108	80
easy p12	110	102
easy p15	228	216
easy p16	308	278
easy p18	310	292
easy p20	498	426
easy p21	508	430
easy p22	634	534
easy p23	660	628
easy p24	662	542

Case Study: Spanner

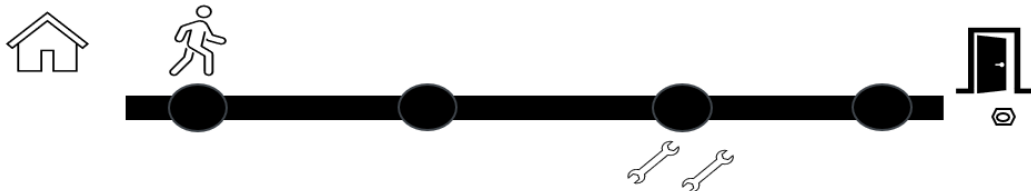
Instance	$\sum f_{\text{ASP}}$	$\sum f_{\text{ASP}_{g,b}}$	Prohibited rule
easy p05	22	12	r126
easy p10	108	80	r143
easy p12	110	102	r126
easy p15	228	216	r126
easy p16	308	278	r145
easy p18	310	292	-
easy p20	498	426	r143
easy p21	508	430	r143
easy p22	634	534	r143
easy p23	660	628	r126
easy p24	662	542	r143

Case Study: Spanner

Instance	$\sum f_{\text{ASP}}$	$\sum f_{\text{ASP}_{g,b}}$	Prohibited rule	Rule impact
easy p05	22	12	r126	100 %
easy p10	108	80	r143	≈ 14 %
easy p12	110	102	r126	100 %
easy p15	228	216	r126	100 %
easy p16	308	278	r145	≈ 27 %
easy p18	310	292	-	-
easy p20	498	426	r143	≈ 11 %
easy p21	508	430	r143	≈ 18 %
easy p22	634	534	r143	10 %
easy p23	660	628	r126	100 %
easy p24	662	542	r143	≈ 23 %

Case Study: Spanner

Easy p05



$w(sh, l1) \rightarrow w(l1, l2) \rightarrow w(l2, l3) \rightarrow pu(s1) \rightarrow pu(s2) \rightarrow w(l3, l4) \rightarrow w(l4, gt) \rightarrow tn(s1, n1)$

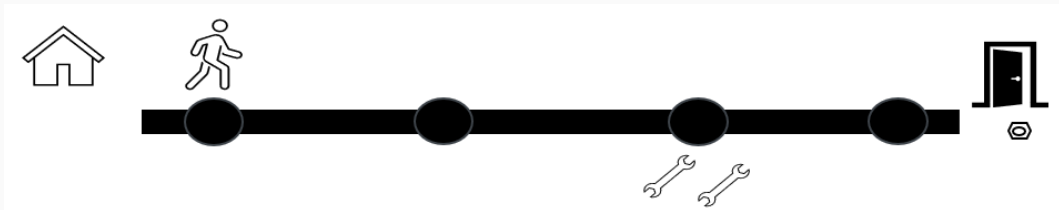
$w(sh, l1) \rightarrow w(l1, l2) \rightarrow w(l2, l3) \rightarrow pu(s1) \rightarrow pu(s2) \rightarrow w(l3, l4) \rightarrow w(l4, gt) \rightarrow tn(s2, n1)$

$w(sh, l1) \rightarrow w(l1, l2) \rightarrow w(l2, l3) \rightarrow pu(s2) \rightarrow pu(s1) \rightarrow w(l3, l4) \rightarrow w(l4, gt) \rightarrow tn(s1, n1)$

$w(sh, l1) \rightarrow w(l1, l2) \rightarrow w(l2, l3) \rightarrow pu(s2) \rightarrow pu(s1) \rightarrow w(l3, l4) \rightarrow w(l4, gt) \rightarrow tn(s2, n1)$

Case Study: Spanner

Easy p05



$w(sh, l1) \rightarrow w(l1, l2) \rightarrow w(l2, l3) \rightarrow pu(s1) \rightarrow pu(s2) \rightarrow w(l3, l4) \rightarrow w(l4, gt) \rightarrow tn(s1, n1)$

$w(sh, l1) \rightarrow w(l1, l2) \rightarrow w(l2, l3) \rightarrow pu(s1) \rightarrow pu(s2) \rightarrow w(l3, l4) \rightarrow w(l4, gt) \rightarrow tn(s2, n1)$

$w(sh, l1) \rightarrow w(l1, l2) \rightarrow w(l2, l3) \rightarrow pu(s2) \rightarrow pu(s1) \rightarrow w(l3, l4) \rightarrow w(l4, gt) \rightarrow tn(s1, n1)$

$w(sh, l1) \rightarrow w(l1, l2) \rightarrow w(l2, l3) \rightarrow pu(s2) \rightarrow pu(s1) \rightarrow w(l3, l4) \rightarrow w(l4, gt) \rightarrow tn(s2, n1)$

..... *pruned plans*

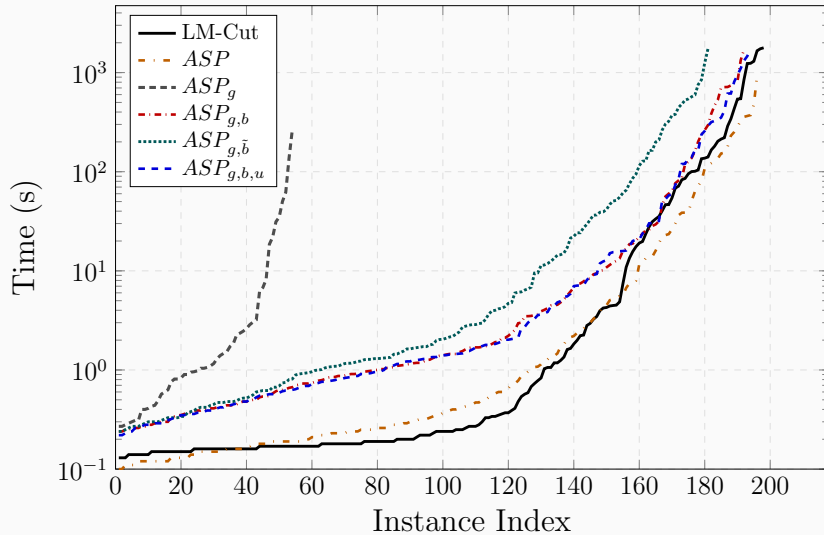
$w(sh, l1) \rightarrow w(l1, l2) \rightarrow w(l2, l3) \rightarrow pu(s1) \rightarrow w(l3, l4) \rightarrow w(l4, gt) \rightarrow tn(s1, n1)$

$w(sh, l1) \rightarrow w(l1, l2) \rightarrow w(l2, l3) \rightarrow pu(s2) \rightarrow w(l3, l4) \rightarrow w(l4, gt) \rightarrow tn(s2, n1)$

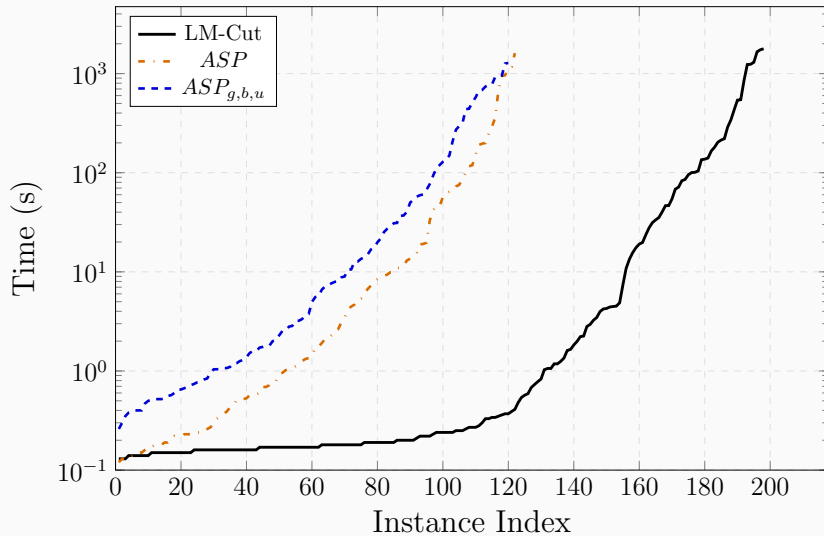
Experiments

- Benchmarks: **IPC 2023** – 10 domains $\times$ 90 instances.
- Baseline: A^* + **LM-Cut**.
- Horizon $H = \lfloor 1.5 \times \text{optimal length} \rfloor$.
- 30 minutes and 4 GiB per instance.
- Transition classifiers trained with default parameters.

Runtime: solved instances



Runtime: facet count



Summary

Takeaways

- A framework that allows for a fully automatic or interactive analysis of symbolic action policies.
- With our framework, it is possible to answer:
 - Why does a policy fail?
 - Why does a working policy prune plans?
- A method to translate symbolic action policies to ASP.

Future work

Scalable encoding | *features* as facets.

Thank you!

- A framework that allows for a fully automatic or interactive analysis of symbolic action policies.
- With our framework, it is possible to answer:
 - Why does a policy fail?
 - Why does a working policy prune plans?
- A method to translate symbolic action policies to ASP.

Questions?

Future work

Scalable encoding | *features* as facets.

References

- Fichte, Johannes Klaus, Sarah Alice Gaggl, and Dominik Rusovac (2022). “Rushing and strolling among answer sets–navigation made easy”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 36. 5, pp. 5651–5659.
- Frances, Guillem, Blai Bonet, and Hector Geffner (2021). “Learning general planning policies from small examples without supervision”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 13, pp. 11801–11808.
- Gnad, Daniel et al. (2025). “Interactive exploration of plan spaces”. In: *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*. Vol. 22. 1, pp. 599–609.
- Musayev, Farid et al. (2025). “Combining Heuristics and Transition Classifiers in Classical Planning”. In: *Proceedings of the 28th European Conference on Artificial Intelligence (ECAI 2025)*. Ed. by Inês Lynce and Aniello Murano. IOS Press, pp. 4694–4701.
- Speck, David et al. (2025). “Counting and reasoning with plans”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 39. 25, pp. 26688–26696.