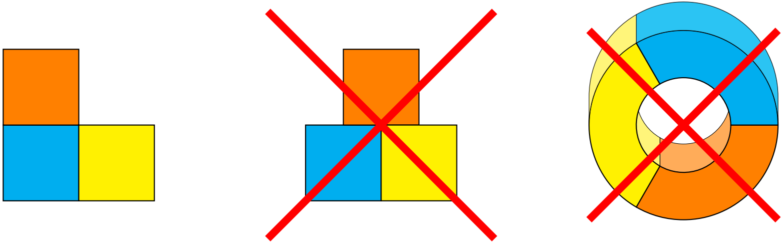
Domain-Independent Instance Generation for Classical Planning

Claudia Grundke   Gabriele Röger   Malte Helmert

University of Basel

KR, November 15, 2025



PDDL Axioms and Polynomial-Time Computations

Grundke et al. (ICAPS 2024)

With a small (but important) language extension,
PDDL axioms can describe exactly those properties
that can be computed in polynomial time (in task size).

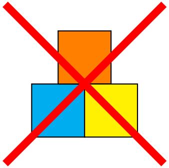
⇝ Use PDDL axioms to determine if a given set of objects and initial state defines a legal instance of the domain.

⇝ declarative definition of what constitutes a classical planning domain that is accessible to algorithms

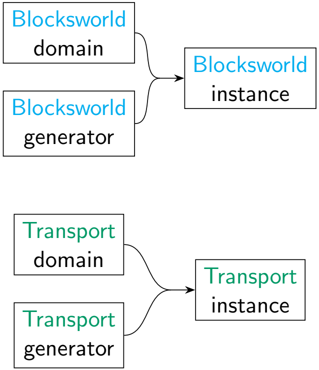
Adding Legality Constraints to PDDL Domains

An instance is legal for a domain if the PDDL axioms yield *legal*() when evaluated on the initial state.

$$
\begin{aligned}
\mathit{illegal}() &\leftarrow \exists b\ \mathit{on}(b,b)\\
\mathit{illegal}() &\leftarrow \exists b_1 \exists b_2 \exists b_3\, (\mathit{on}(b_1,b_2) \land \mathit{on}(b_1,b_3) \land b_2 \neq b_3)\\
\mathit{illegal}() &\leftarrow \exists b_1 (\neg \mathit{ontable}(b_1) \land \neg \exists b_2\ \mathit{on}(b_1,b_2))\\
\mathit{illegal}() &\leftarrow \ \ldots\\
&\ \ldots\\
\mathit{legal}() &\leftarrow \neg \mathit{illegal}()
\end{aligned}
$$



Domain-Specific vs. Domain-Independent



Blocksworld domain

Blocksworld generator

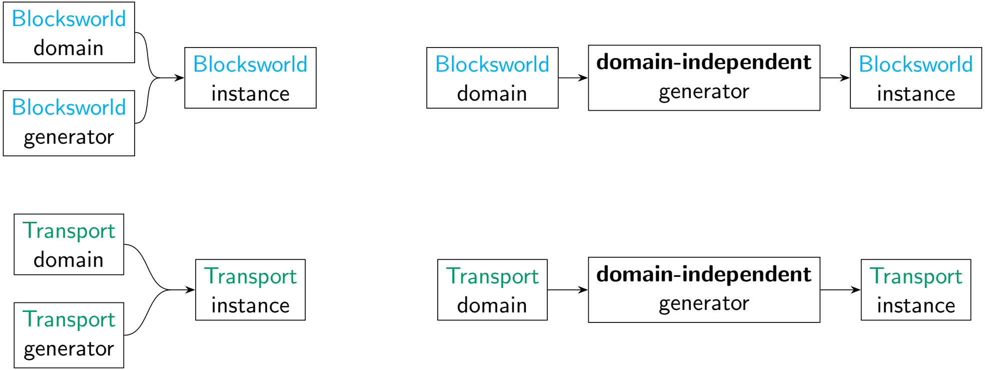
Blocksworld instance

Transport domain

Transport generator

Transport instance

Domain-Specific vs. Domain-Independent



Blocksworld domain

Blocksworld generator

Blocksworld instance

Blocksworld domain

domain-independent generator

Blocksworld instance

Transport domain

Transport generator

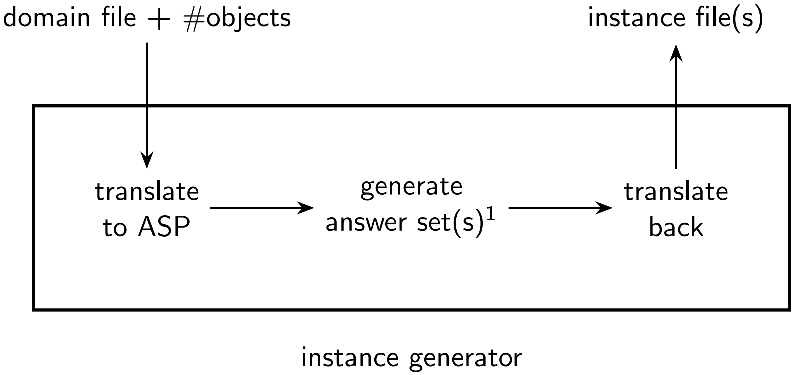
Transport instance

Transport domain

domain-independent generator

Transport instance

Instance Generation using ASP



[1] Gebser, Kaminski, Kaufmann, Schaub, Multi-shot ASP solving with clingo (TPLP 2019)

PDDL Axioms are almost ASP Rules

PDDL Axioms → **ASP Rules**

rule bodies are
first-order
logic formulas

rule bodies are
existentially quantified
conjunctions of literals

Conversion is easy except when it isn't (Röger & Grundke, PuK 2024).

Instance Generation as ASP Solving

Take number of objects as input
(optionally: number of objects per PDDL type).

Use ASP choice rules to “guess” an arbitrary initial state using these objects.

Use legality predicate to validate that the initial state is legal.
$\leftarrow \neg \mathit{legal}()$

Translating Type Information

For each object we set a type predicate.

We replicate the type hierarchy.

We ensure that each predicate is instantiated only with objects of appropriate types.

Translating Type Information

For each object we set a type predicate.

$\textit{truck}(\texttt{t}_1).$

We replicate the type hierarchy.

$$\textit{vehicle}(x) \leftarrow \textit{truck}(x)$$
$$\textit{object}(x) \leftarrow \textit{vehicle}(x)$$

We ensure that each predicate is instantiated only with objects of appropriate types.

Diversity

ASP solver clingo returns all answer sets
or a fixed number of (often similar) answer sets.

But we want a diverse subset of all answer sets.

➡ Böhl, Gaggl, Rusovac: collection of answer sets with explicit diversification

Böhl, Gaggl, Rusovac, Representative Answer Sets: Collecting Something of Everything (ECAI 2023)

Case Study: Reproduce IPC Learning Competition

Augment PDDL domains with legality constraints.

Generate training and testing instances
following the spread of IPC 2023 learning track[1].

For each domain and instance size, run instance generator
with 30 minute time limit, 4 GiB memory limit.

[1] Taitler, Alford, Espasa, Behnke, Fišer, Gimelfarb, Pommerening, Sanner, Scala, Schreiber, Segovia-Aguas, Seipp, The 2023 International Planning Competition (AI Magazine 2024)

Results: Instance Generation

More than enough instances for small numbers of objects

We needed 6 of 1057 generated Blocksworld instances with 26 objects.

No instances for large (out of time) and very large (out of memory) numbers of objects.

IPC had Blocksworld instances with up to 500 objects.

	training instances	testing instances
per domain	99 / 99 (two exceptions)	$\ll$ 90 / 90
total	805 / 891	340 / 810

Setup: Cross-Validation

Train each planner on the generated and on the IPC learning track[1] instances.

Test each planner variant on both kinds of instances.

IPC planners on IPC instances	ASP planners on IPC instances
IPC planners on ASP instances	ASP planners on ASP instances

[1] Taitler, Alford, Espasa, Behnke, Fišer, Gimelfarb, Pommerening, Sanner, Scala, Schreiber, Segovia-Aguas, Seipp, The 2023 International Planning Competition (AI Magazine 2024)

Results: Cross-Validation

mean coverage

IPC-IPC	ASP-IPC
368.7 / 810	354.2 / 810
203.2 / 340	208.7 / 340
IPC-ASP	ASP-ASP

strictly better
domain coverage

IPC-IPC	ASP-IPC
23 cases	10 cases
8 cases	9 cases
IPC-ASP	ASP-ASP

IPC-IPC = IPC planners on IPC instances	ASP-IPC = ASP planners on IPC instances
IPC-ASP = IPC planners on ASP instances	ASP-ASP = ASP planners on ASP instances

Summary

Translation from PDDL axioms to ASP is reasonably easy.

ASP allows us to guess and verify legal initial states and has approaches for creating diverse answer sets.

Scaling not as good as that of domain-specific generators but still suitable for training learning-based planners.