Formal Representations of Classical Planning Domains

Claudia Grundke, Gabriele Röger, Malte Helmert

PDDL domain descriptions overapproximate domains. This is fine for solving individual tasks but it is a **problem e. g. for generalized planning or automated instance generation**.

We want to

- **describe domains precisely**,
- stay compatible with existing **PDDL** benchmarks,
- **efficiently check** if a task belongs to the domain.

Basic Idea

- All tasks of a domain share **same first-order goal** and only **differ in the set of objects and the initial state**.
- Task belongs to a domain if **derived query predicate** *legal*() is true in the initial state.
- **PDDL axioms** characterize legal tasks:

$$\begin{aligned} \mathit{above}(x,y) &\leftarrow \mathit{on}(x,y) \lor \exists z(\mathit{on}(x,z) \\ &\quad \land \mathit{above}(z,y)) \\ \mathit{illegal}() &\leftarrow \exists x\, \mathit{above}(x,x) \\ \mathit{illegal}() &\leftarrow \ldots \\ \mathit{legal}() &\leftarrow \neg \mathit{illegal}() \end{aligned}$$

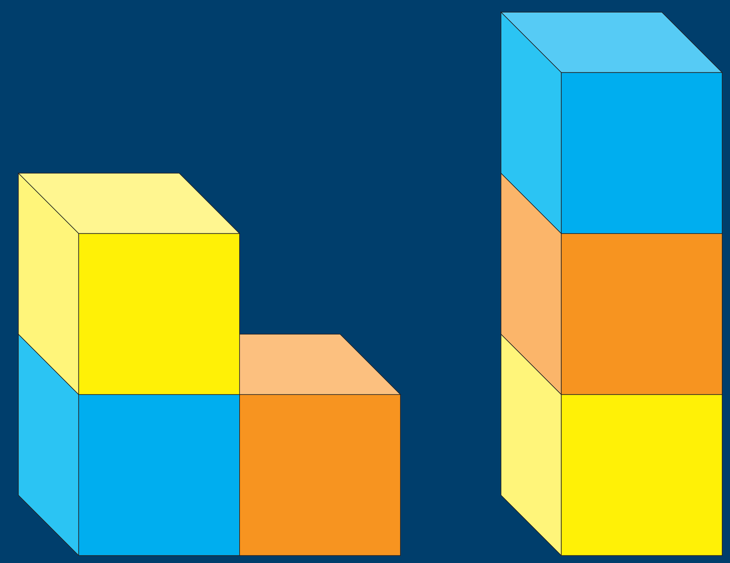
Paper contains examples suitably augmenting existing IPC benchmarks.

Domain-wide Goal

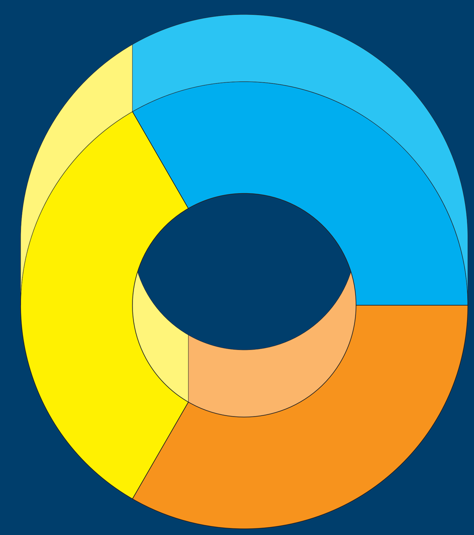
- Shared goal ensures **desirable conformity** of tasks.
- Not too restrictive because we can **move STRIPS goal into initial state**:
  - For STRIPS goal $\mathit{on}(A,B) \land \mathit{on}(B,C) \land \mathit{on}(C,D)$
  - Add to initial state: $\mathit{on}^{g}(A,B), \mathit{on}^{g}(B,C), \mathit{on}^{g}(C,D)$
  - Goal of all Blocksworld tasks then is $\forall x, x'(\mathit{on}^{g}(x,x') \rightarrow \mathit{on}(x,x'))$

University of Basel

Which tasks belong to a planning domain?



Blocksworld



Blocksworld?

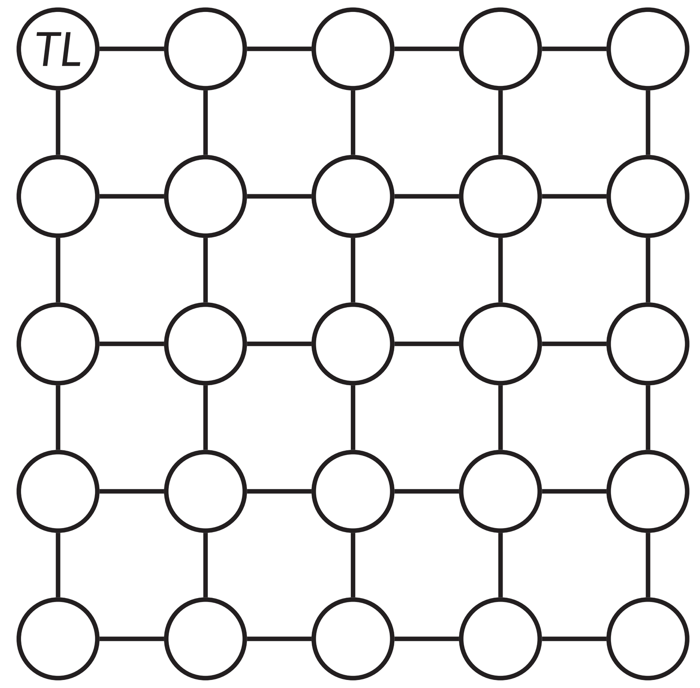
If there exists a **polynomial-time decision** algorithm, we can capture it using an extended form of **PDDL axioms.**

Take a picture to download the full paper.

Adding Order

- Extension to PDDL: Axioms may exploit an **arbitrary linear order** $<$ over all objects.
- Set of axioms must be **order-invariant**, i. e. the evaluation result of the axioms may not depend on the exact order.

Example:
Characterize grids based on edge relation E.



- Use linear order to distinguish one corner.

$$\begin{aligned} \mathit{corner}(x) &\leftarrow \mathit{degree}_{\geq 2}(x) \land \neg \mathit{degree}_{\geq 3}(x) \\ \mathit{TL}(x) &\leftarrow \mathit{corner}(x) \land \\ &\quad \neg \exists y\, (\mathit{corner}(y) \land y < x) \end{aligned}$$

- Smaller neighbor lies to the right, larger neighbor below.

$$\begin{aligned} \mathit{rightOfTL}(x) &\leftarrow \exists c, y\, (\mathit{TL}(c) \land \\ &\quad E(x,c) \land E(y,c) \\ &\quad \land x < y) \\ \mathit{belowOfTL}(x) &\leftarrow \exists c, y\, (\mathit{TL}(c) \land \\ &\quad E(x,c) \land E(y,c) \\ &\quad \land y < x) \end{aligned}$$

Complexity & Expressiveness

- **Complexity**: Legality can be decided in polynomial time (in the size of the task description) for a fixed domain.
- **Expressiveness**: Our formalism captures P (Immerman-Vardi Theorem), i. e. if any polynomial-time algorithm can decide if a task is legal, we can express it.