

Foundations of Artificial Intelligence

F2. Automated Planning: Planning Formalisms

Malte Helmert

University of Basel

April 29, 2026

Foundations of Artificial Intelligence

April 29, 2026 — F2. Automated Planning: Planning Formalisms

F2.1 Four Formalisms

F2.2 STRIPS

F2.3 ADL, SAS⁺ and PDDL

F2.4 Summary

Automated Planning: Overview

Chapter overview: automated planning

- ▶ F1. Introduction
- ▶ **F2. Planning Formalisms**
- ▶ F3. Delete Relaxation
- ▶ F4. Delete Relaxation Heuristics
- ▶ F5. Abstraction
- ▶ F6. Abstraction Heuristics

F2.1 Four Formalisms

Four Planning Formalisms

- ▶ A description language for state spaces (**planning tasks**) is called a **planning formalism**.
- ▶ We introduce four planning formalisms:
 - ① STRIPS (Stanford Research Institute Problem Solver)
 - ② ADL (Action Description Language)
 - ③ SAS⁺ (Simplified Action Structures)
 - ④ PDDL (Planning Domain Definition Language)
- ▶ STRIPS and SAS⁺ are the most simple formalisms; in the next chapters, we only consider these.

F2.2 STRIPS

STRIPS: Basic Concepts

basic concepts of STRIPS:

- ▶ STRIPS is the **most simple** common planning formalism.
- ▶ state variables are **binary** (true or false)
- ▶ **states** s (based on a given set of state variables V) can be represented in two equivalent ways:
 - ▶ as **assignments** $s : V \rightarrow \{\mathbf{F}, \mathbf{T}\}$
 - ▶ as **sets** $s \subseteq V$,
where s encodes the set of state variables that are **true** in s
- We will use the set representation.
- ▶ **goals** and **preconditions of actions** are given as sets of variables that must be **true** (values of other variables do not matter)
- ▶ **effects of actions** are given as sets of variables that are **set to true** and **set to false**, respectively

STRIPS Planning Task

Definition (STRIPS Planning Task)

A **STRIPS** planning task is a 4 tuple $\Pi = \langle V, I, G, A \rangle$ with

- ▶ V : finite set of **state variables**
- ▶ $I \subseteq V$: the **initial state**
- ▶ $G \subseteq V$: the set of **goals**
- ▶ A : finite set of **actions**,
where for all actions $a \in A$, the following is defined:
 - ▶ $pre(a) \subseteq V$: the **preconditions** of a
 - ▶ $add(a) \subseteq V$: the **add effects** of a
 - ▶ $del(a) \subseteq V$: the **delete effects** of a
 - ▶ $cost(a) \in \mathbb{N}_0$: the **costs** of a

German: STRIPS-Planungsaufgabe, Zustandsvariablen, Anfangszustand, Ziele, Aktionen, Add-/Delete-Effekte, Kosten
remark: action costs are an extension of “traditional” STRIPS

State Space for STRIPS Planning Task

Definition (state space induced by STRIPS planning task)

Let $\Pi = \langle V, I, G, A \rangle$ be a STRIPS planning task.

Then Π induces the state space $\mathcal{S}(\Pi) = \langle S, A, cost, T, s_1, S_G \rangle$:

- ▶ set of states: $S = 2^V$ (= power set of V)
- ▶ actions: actions A as defined in Π
- ▶ action costs: *cost* as defined in Π
- ▶ transitions: $s \xrightarrow{a} s'$ for states $s, s' \in S$ and action $a \in A$ iff
 - ▶ $pre(a) \subseteq s$ (preconditions satisfied)
 - ▶ $s' = (s \setminus del(a)) \cup add(a)$ (effects are applied)
- ▶ initial state: $s_1 = I$
- ▶ goal states: $s \in S_G$ for state s iff $G \subseteq s$ (goals reached)

German: induziert den Zustandsraum

Example: Blocks World in STRIPS

Example (A Blocks World Planning Task in STRIPS)

$\Pi = \langle V, I, G, A \rangle$ with:

- ▶ $V = \{on_{R,B}, on_{R,G}, on_{B,R}, on_{B,G}, on_{G,R}, on_{G,B}, on-table_R, on-table_B, on-table_G, clear_R, clear_B, clear_G\}$
 - ▶ $I = \{on_{G,R}, on-table_R, on-table_B, clear_G, clear_B\}$
 - ▶ $G = \{on_{R,B}, on_{B,G}\}$
 - ▶ $A = \{move_{R,B,G}, move_{R,G,B}, move_{B,R,G}, move_{B,G,R}, move_{G,R,B}, move_{G,B,R}, to-table_{R,B}, to-table_{R,G}, to-table_{B,R}, to-table_{B,G}, to-table_{G,R}, to-table_{G,B}, from-table_{R,B}, from-table_{R,G}, from-table_{B,R}, from-table_{B,G}, from-table_{G,R}, from-table_{G,B}\}$
- ...

Example: Blocks World in STRIPS

Example (A Blocks World Planning Task in STRIPS)

move actions encode moving a block from one block to another

example:

- ▶ $pre(move_{R,B,G}) = \{on_{R,B}, clear_R, clear_G\}$
- ▶ $add(move_{R,B,G}) = \{on_{R,G}, clear_B\}$
- ▶ $del(move_{R,B,G}) = \{on_{R,B}, clear_G\}$
- ▶ $cost(move_{R,B,G}) = 1$

Example: Blocks World in STRIPS

Example (A Blocks World Planning Task in STRIPS)

to-table actions encode moving a block from a block to the table

example:

- ▶ $pre(to-table_{R,B}) = \{on_{R,B}, clear_R\}$
- ▶ $add(to-table_{R,B}) = \{on-table_R, clear_B\}$
- ▶ $del(to-table_{R,B}) = \{on_{R,B}\}$
- ▶ $cost(to-table_{R,B}) = 1$

Example: Blocks World in STRIPS

Example (A Blocks World Planning Task in STRIPS)

from-table actions encode moving a block from the table to a block

example:

- ▶ $pre(\text{from-table}_{R,B}) = \{on\text{-table}_R, clear_R, clear_B\}$
- ▶ $add(\text{from-table}_{R,B}) = \{on_{R,B}\}$
- ▶ $del(\text{from-table}_{R,B}) = \{on\text{-table}_R, clear_B\}$
- ▶ $cost(\text{from-table}_{R,B}) = 1$

Why STRIPS?

- ▶ STRIPS is **particularly simple**.
- ↪ simplifies the design and implementation of planning algorithms
- ▶ often cumbersome for the user to model tasks directly in STRIPS
- ▶ **but:** STRIPS is equally “powerful” to much more complex planning formalisms
- ↪ automatic “compilers” exist that translate more complex formalisms (like ADL and SAS⁺) to STRIPS

F2.3 ADL, SAS⁺ and PDDL

Basic Concepts of ADL

basic concepts of ADL:

- ▶ Like STRIPS, ADL uses propositional variables (true/false) as state variables.
- ▶ preconditions of actions and goal are **arbitrary logic formulas** (action applicable/goal reached in states that satisfy the formula)
- ▶ in addition to STRIPS effects, there are **conditional effects**: variable v is only set to true/false if a given logical formula is true in the current state

Basic Concepts of SAS⁺

basic concepts of SAS⁺:

- ▶ very similar to STRIPS: state variables not necessarily binary, but with given **finite domain** (cf. CSPs)
- ▶ states are **assignments** to these variables (cf. CSPs)
- ▶ preconditions and goals given as **partial assignments**
example: $\{v_1 \mapsto a, v_3 \mapsto b\}$ as preconditions (or goals)
 - ▶ If $s(v_1) = a$ and $s(v_3) = b$, then the action is applicable in s (or goal is reached)
 - ▶ values of other variables do not matter
- ▶ effects are **assignments to subset** of variables
example: effect $\{v_1 \mapsto b, v_2 \mapsto c\}$ means
 - ▶ In the successor state s' , $s'(v_1) = b$ and $s'(v_2) = c$.
 - ▶ All other variables retain their values.

Basic Concept of PDDL

- ▶ PDDL is the standard language used in practice to describe planning tasks.
- ▶ descriptions in (restricted) predicate logic instead of propositional logic ($\neg \rightarrow$ even more compact)
- ▶ other features like **numeric variables** and **derived variables (axioms)** for defining complex logical conditions (formulas that are automatically evaluated in every state and can, e.g., be used in preconditions)
- ▶ There exist defined PDDL fragments for STRIPS and ADL; many planners only support the STRIPS fragment.

example: blocks world in PDDL

F2.4 Summary

Summary

planning formalisms:

- ▶ **STRIPS:** particularly simple, easy to handle for algorithms
 - ▶ binary state variables
 - ▶ preconditions, add and delete effects, goals: sets of variables
- ▶ **ADL:** extension of STRIPS
 - ▶ **logic formulas** for complex preconditions and goals
 - ▶ **conditional effects**
- ▶ **SAS⁺:** extension of STRIPS
 - ▶ state variables with **arbitrary finite domains**
- ▶ **PDDL:** input language used in practice
 - ▶ based on predicate logic (more compact than propositional logic)
 - ▶ only partly supported by most algorithms (e.g., STRIPS or ADL fragment)