

# Foundations of Artificial Intelligence

## 34. Automated Planning: Planning Formalisms

Malte Helmert

University of Basel

April 29, 2019

# Automated Planning: Overview

## Chapter overview: automated planning

- 33. Introduction
- 34. Planning Formalisms
- 35.–36. Planning Heuristics: Delete Relaxation
- 37. Planning Heuristics: Abstraction
- 38.–39. Planning Heuristics: Landmarks

# 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<sup>+</sup> ([Simplified Action Structures](#))
  - 4 PDDL ([Planning Domain Definition Language](#))
- STRIPS and SAS<sup>+</sup> are the most simple formalisms; in the next chapters, we restrict our considerations to these.

# STRIPS

# STRIPS: Basic Concepts

basic concepts of STRIPS:

- STRIPS is the **most simple** common planning formalism.
- state variables are **binary** (true or false)

# 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.

# 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  $\Pi$  induces the state space  $\mathcal{S}(\Pi) = \langle S, A, \text{cost}, T, s_0, S_* \rangle$ :

- set of states:  $S = 2^V$  (= power set of  $V$ )
- actions: actions  $A$  as defined in  $\Pi$
- action costs:  $\text{cost}$  as defined in  $\Pi$
- transitions:  $s \xrightarrow{a} s'$  for states  $s, s'$  and action  $a$  iff
  - $\text{pre}(a) \subseteq s$  (preconditions satisfied)
  - $s' = (s \setminus \text{del}(a)) \cup \text{add}(a)$  (effects are applied)
- initial state:  $s_0 = I$
- goal states:  $s \in S_*$  for state  $s$  iff  $G \subseteq s$  (goals reached)

German: durch STRIPS-Planungsaufgabe  
induzierter Zustandsraum

## Example: Blocks World in STRIPS

## Example (A Blocks World Planning Task in STRIPS)

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

- $V = \{on_{A,B}, on_{A,C}, on_{B,A}, on_{B,C}, on_{C,A}, on_{C,B}, on-table_A, on-table_B, on-table_C, clear_A, clear_B, clear_C\}$
- $I = \{on_{C,A}, on-table_A, on-table_B, clear_C, clear_B\}$
- $G = \{on_{A,B}, on_{B,C}\}$
- $A = \{move_{A,B,C}, move_{A,C,B}, move_{B,A,C}, move_{B,C,A}, move_{C,A,B}, move_{C,B,A}, to-table_{A,B}, to-table_{A,C}, to-table_{B,A}, to-table_{B,C}, to-table_{C,A}, to-table_{C,B}, from-table_{A,B}, from-table_{A,C}, from-table_{B,A}, from-table_{B,C}, from-table_{C,A}, from-table_{C,B}\}$

...

# 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:

- $pre(move_{A,B,C}) = \{on_{A,B}, clear_A, clear_C\}$
- $add(move_{A,B,C}) = \{on_{A,C}, clear_B\}$
- $del(move_{A,B,C}) = \{on_{A,B}, clear_C\}$
- $cost(move_{A,B,C}) = 1$

# 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:

- $pre(to-table_{A,B}) = \{on_{A,B}, clear_A\}$
- $add(to-table_{A,B}) = \{on-table_A, clear_B\}$
- $del(to-table_{A,B}) = \{on_{A,B}\}$
- $cost(to-table_{A,B}) = 1$

# 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:

- $pre(from-table_{A,B}) = \{on-table_A, clear_A, clear_B\}$
- $add(from-table_{A,B}) = \{on_{A,B}\}$
- $del(from-table_{A,B}) = \{on-table_A, clear_B\}$
- $cost(from-table_{A,B}) = 1$

# 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<sup>+</sup>) to STRIPS

# ADL, SAS<sup>+</sup> 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<sup>+</sup>

## basic concepts of SAS<sup>+</sup>:

- very similar to STRIPS: state variables not necessarily binary, but with given **finite domain** (cf. CSPs)
- states are **assignments** to these variables (cf. CSPs)

# Basic Concepts of SAS<sup>+</sup>

## basic concepts of SAS<sup>+</sup>:

- 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

# Basic Concepts of SAS<sup>+</sup>

## basic concepts of SAS<sup>+</sup>:

- 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 (↔ even more compact)
- other features like **numeric variables** and **derived variables (axioms)** for defining “macros” (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

# 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<sup>+</sup>**: 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)