

# Foundations of Artificial Intelligence

## F1. Automated Planning: Introduction

Malte Helmert

University of Basel

April 27, 2026

# Automated Planning: Overview

## Chapter overview: automated planning

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

# Classification

classification:

## Automated Planning

environment:

- static vs. dynamic
- deterministic vs. nondeterministic vs. stochastic
- fully observable vs. partially observable
- discrete vs. continuous
- single-agent vs. multi-agent

problem solving method:

- problem-specific vs. **general** vs. learning

(applications also in more complex environments)

# Introduction

# Automated Planning

## What is Automated Planning?

“Planning is the art and practice of thinking before acting.”

— P. Haslum

↔ finding **plans** (sequences of actions)  
that lead from an initial state to a goal state

our topic in this course: **classical planning**

- **general** approach to finding solutions  
for **state-space search problems** (Part B)
- **classical** = static, deterministic, fully observable
- **variants**: probabilistic planning, planning under partial observability, online planning, . . .

# Planning: Informally

given:

- state space description in terms of suitable problem description language (**planning formalism**)

required:

- a **plan**, i.e., a solution for the described state space (sequence of actions from initial state to goal)
- or a proof that no plan exists

distinguish between

- **optimal planning**: guarantee that returned plans are optimal, i.e., have minimal overall cost
- **suboptimal planning** (**satisficing**): suboptimal plans are allowed

# What is New?

Many previously encountered problems are planning tasks:

- blocks world
- missionaries and cannibals
- 15-puzzle

**New:** we are now interested in **general** algorithms, i.e., the developer of the search algorithm **does not know** the tasks that the algorithm needs to solve.

- ↪ no problem-specific heuristics!
- ↪ **input language** to model the planning task

# Repetition: State Spaces



# Formal Models for State-Space Search

To cleanly study search problems we need a **formal model**.

## Nothing New Here!

This section is a **repetition** of Section B1.2 of the chapter “State-Space Search: State Spaces”.

# State Spaces

## Definition (state space)

A **state space** or **transition system** is a 6-tuple  $\mathcal{S} = \langle S, A, cost, T, s_1, S_G \rangle$  with

- finite set of **states**  $S$
- finite set of **actions**  $A$
- **action costs**  $cost : A \rightarrow \mathbb{R}_0^+$
- **transition relation**  $T \subseteq S \times A \times S$  that is **deterministic in  $\langle s, a \rangle$**  (see next slide)
- **initial state**  $s_1 \in S$
- set of **goal states**  $S_G \subseteq S$

**German:** Zustandsraum, Transitionssystem, Zustände, Aktionen, Aktionskosten, Transitions-/Übergangsrelation, deterministisch, Anfangszustand, Zielzustände

# State Spaces: Terminology & Notation

## Definition (transition, deterministic)

Let  $\mathcal{S} = \langle S, A, cost, T, s_1, S_G \rangle$  be a state space.

The triples  $\langle s, a, s' \rangle \in T$  are called **(state) transitions**.

We say  $\mathcal{S}$  **has the transition**  $\langle s, a, s' \rangle$  if  $\langle s, a, s' \rangle \in T$ .

We write this as  $s \xrightarrow{a} s'$ , or  $s \rightarrow s'$  when  $a$  does not matter.

Transitions are **deterministic** in  $\langle s, a \rangle$ : it is forbidden to have both  $s \xrightarrow{a} s_1$  and  $s \xrightarrow{a} s_2$  with  $s_1 \neq s_2$ .

# Graph Interpretation

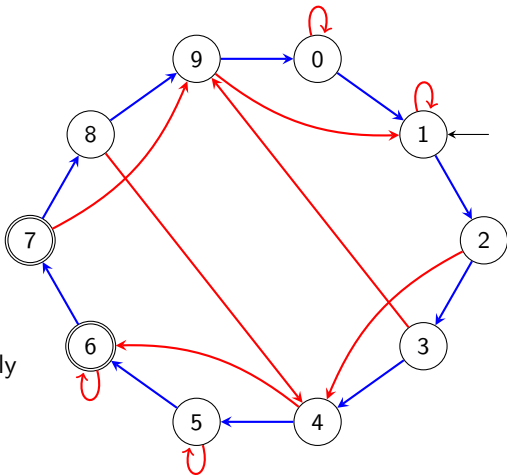
state spaces are often depicted as **directed, labeled graphs**

- **states:** graph vertices
- **transitions:** labeled arcs
  
- **initial state:** incoming arrow
- **goal states:** double circles
- **actions:** the arc labels
- **action costs:** described separately  
(or implicitly = 1)

# Graph Interpretation

state spaces are often depicted as **directed, labeled graphs**

- **states**: graph vertices
- **transitions**: labeled arcs (here: colors instead of labels)
- **initial state**: incoming arrow
- **goal states**: double circles
- **actions**: the arc labels
- **action costs**: described separately (or implicitly = 1)



# State Spaces: Terminology

## terminology:

- predecessor, successor
- applicable action
- path, length, costs
- reachable
- solution, optimal solution

**German:** Vorgänger, Nachfolger, anwendbare Aktion, Pfad, Länge, Kosten, erreichbar, Lösung, optimale Lösung

# Compact Descriptions

# State Spaces with Declarative Representations

How do we represent state spaces in the computer?

**previously:** as black box

**now:** as **declarative description**

reminder: Chapter B2

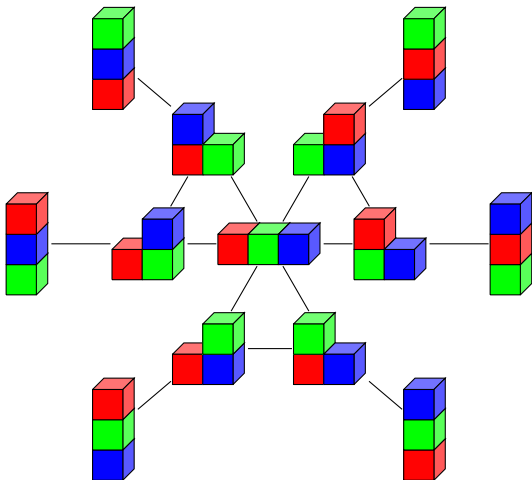
## State Spaces with Declarative Representations

represent state spaces **declaratively**:

- **compact** description of state space as input to algorithms  
     $\rightsquigarrow$  state spaces **exponentially larger** than the input
- algorithms directly operate on compact description  
     $\rightsquigarrow$  allows automatic reasoning about problem:  
        reformulation, simplification, abstraction, etc.



# Reminder: Blocks World



problem:  $n$  blocks  $\rightsquigarrow$  more than  $n!$  states

# Compact Description of State Spaces

## How to describe state spaces compactly?

### Compact Description of Several States

- introduce **state variables**
- states: assignments to state variables
- ↪ e.g.,  $n$  binary state variables can describe  $2^n$  states
- **transitions** and **goal states** are compactly described with a logic-based formalism

different variants: different **planning formalisms**

# Summary

# Summary

- **planning:** search in **general** state spaces
- **input:** compact, declarative description of state space