

Foundations of Artificial Intelligence

33. Automated Planning: Introduction

Malte Helmert

University of Basel

April 24, 2019

Classification

classification:

Automated Planning

environment:

- **static** vs. dynamic
- **deterministic** vs. non-deterministic vs. stochastic
- **fully** vs. partially vs. not **observable**
- **discrete** vs. continuous
- **single-agent** vs. multi-agent

problem solving method:

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

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** (Chapters 5–19)
- **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

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

Repetition: State Spaces

About This Section

Nothing New Here!

This section is a **repetition** of Section 5.2 of the chapter “State-Space Search: State Spaces”.

Formalization of State Spaces

preliminary remarks:

- to cleanly study search problems we need a **formal model**
- fundamental concept: **state spaces**
- state spaces are (labeled, directed) **graphs**
- **paths** to goal states represent **solutions**
- **shortest paths** correspond to **optimal solutions**

State Spaces

Definition (state space)

A **state space** or **transition system** is a 6-tuple

$\mathcal{S} = \langle S, A, cost, T, s_0, S_\star \rangle$ with

- S : finite set of **states**
- A : finite set of **actions**
- $cost : A \rightarrow \mathbb{R}_0^+$ **action costs**
- $T \subseteq S \times A \times S$ **transition relation**; **deterministic** in $\langle s, a \rangle$
(see next slide)
- $s_0 \in S$ **initial state**
- $S_\star \subseteq S$ set of **goal states**

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

State Spaces: Transitions, Determinism

Definition (transition, deterministic)

Let $\mathcal{S} = \langle S, A, cost, T, s_0, S_\star \rangle$ 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 $s_1 \neq s_2$.

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**

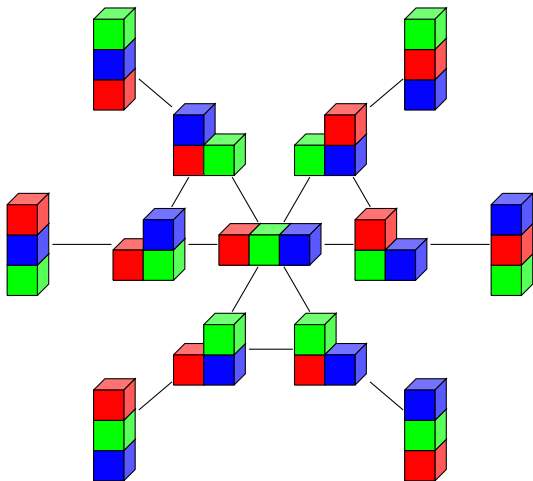
reminder: Chapter 6

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** 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