

# Foundations of Artificial Intelligence

## 6. State-Space Search: Representation of State Spaces

Thomas Keller and Florian Pommerening

University of Basel

March 8, 2023

# Foundations of Artificial Intelligence

March 8, 2023 — 6. State-Space Search: Representation of State Spaces

## 6.1 Representation of State Spaces

## 6.2 Explicit Graphs

## 6.3 Declarative Representations

## 6.4 Black Box

## 6.5 Summary

## State-Space Search: Overview

### Chapter overview: state-space search

- ▶ 5.–7. Foundations
  - ▶ 5. State Spaces
  - ▶ 6. Representation of State Spaces
  - ▶ 7. Examples of State Spaces
- ▶ 8.–12. Basic Algorithms
- ▶ 13.–19. Heuristic Algorithms

## 6.1 Representation of State Spaces

## Representation of State Spaces

- ▶ practically interesting state spaces are often huge ( $10^{10}$ ,  $10^{20}$ ,  $10^{100}$  states)
- ▶ How do we represent them, so that we can efficiently deal with them algorithmically?

three main options:

- 1 as explicit (directed) graphs
- 2 with declarative representations
- 3 as a black box

## 6.2 Explicit Graphs

## State Spaces as Explicit Graphs

### State Spaces as Explicit Graphs

represent state spaces as explicit directed graphs:

- ▶ vertices = states
- ▶ directed arcs = transitions

↪ represented as adjacency list or adjacency matrix

Example (explicit graph for bounded inc-and-square)  
ai06-bounded-inc-and-square.graph

## State Spaces as Explicit Graphs

### State Spaces as Explicit Graphs

represent state spaces as explicit directed graphs:

- ▶ vertices = states
- ▶ directed arcs = transitions

↪ represented as adjacency list or adjacency matrix

Example (explicit graph for 8-puzzle)  
ai06-puzzle8.graph

## State Spaces as Explicit Graphs: Discussion

discussion:

- ▶ impossible for large state spaces (too much space required)
- ▶ if spaces small enough for explicit representations, solutions easy to compute: Dijkstra's algorithm  $O(|S| \log |S| + |T|)$
- ▶ interesting for time-critical all-pairs-shortest-path queries (examples: route planning, path planning in video games)

## 6.3 Declarative Representations

## State Spaces with Declarative Representations

### State Spaces with Declarative Representations

represent state spaces declaratively:

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

Example (declarative representation for 8-puzzle)

puzzle8-domain.pddl + puzzle8-problem.pddl

## 6.4 Black Box

## State Spaces as Black Boxes

### State Spaces as Black Boxes

Define an **abstract interface** for state spaces.

For state space  $\mathcal{S} = \langle S, A, cost, T, s_I, S_\star \rangle$

we need these methods:

- ▶ **init()**: generate initial state  
result: state  $s_I$
- ▶ **is\_goal( $s$ )**: test if  $s$  is a goal state  
result: **true** if  $s \in S_\star$ ; **false** otherwise
- ▶ **succ( $s$ )**: generate applicable actions and successors of  $s$   
result: sequence of pairs  $\langle a, s' \rangle$  with  $s \xrightarrow{a} s'$
- ▶ **cost( $a$ )**: gives cost of action  $a$   
result:  $cost(a) \in \mathbb{N}_0$

**remark:** we extend the interface later in a small but important way

## State Spaces as Black Boxes: Example and Discussion

### Example (Black Box Representation for 8-Puzzle)

demo: `puzzle8.py`

- ▶ **in the following:** focus on black box model
- ▶ **explicit graphs** only as illustrating examples
- ▶ **near end of semester:** declarative state spaces (classical planning)

## 6.5 Summary

## Summary

- ▶ state spaces often **huge** ( $> 10^{10}$  states)  
~> **how to represent?**
- ▶ **explicit graphs:** adjacency lists or matrices;  
only suitable for small problems
- ▶ **declaratively:** compact description as input  
to search algorithms
- ▶ **black box:** implement an abstract interface