Principles of AI Planning

14. Nondeterministic planning

Albert-Ludwigs-Universität Freiburg

Bernhard Nebel and Robert Mattmüller

January 11, 2019

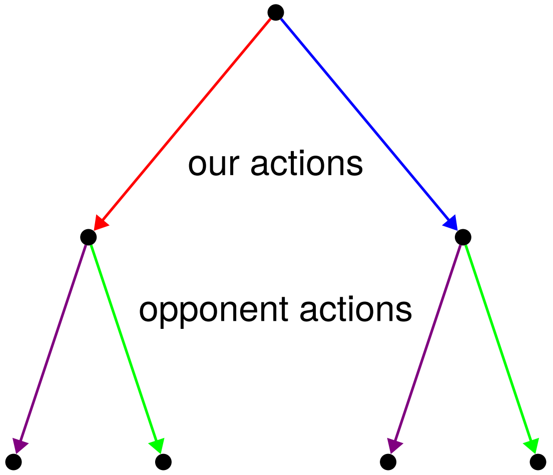
Motivation

Nondeterministic planning

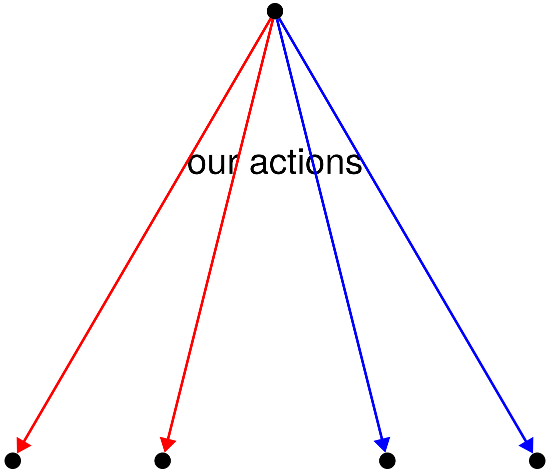
- The world is not predictable.
- AI robotics:
  - imprecise movement of the robot
  - other robots
  - human beings, animals
  - machines (cars, trains, airplanes, lawn-mowers, . . .)
  - natural phenomena (wind, water, snow, temperature, . . .)
- Games: other players are outside our control.
  - To win a game (reaching a goal state) with certainty, all possible actions by the other players have to be anticipated (a winning strategy of a game).
  - The world is not predictable because it is unknown: we cannot observe everything.

In this lecture, we will only deal with uncertain operator outcomes, not with partial observability.

Nondeterminism in games



Nondeterminism in games



Motivation
Transition systems and planning tasks
Plans

Nondeterministic planning

- In deterministic planning we have assumed that the only changes taking place in the world are those caused by us and that we can exactly predict the results of our actions.
- Other agents and processes, beyond our control, are formalized as (demonic) nondeterminism.
- Implications:
  1. The future state of the world cannot be predicted.
  2. We cannot reliably plan ahead: no single operator sequence achieves the goals.
  3. In some cases it is not possible to achieve the goals with certainty no matter which outcomes the actions have, but only under certain fairness assumptions.

Motivation
Transition systems and planning tasks
Plans

A note on the term *nondeterminism*

Nondeterminism occurs in three different kinds in computer science:

- demonic nondeterminism as in nondeterministic planning, i.e., on has to be prepared that the worst-case happens;
- angelic nondeterminism as in nondeterministic automata, where always the best choice is taken;
- don’t care nondeterminism as in the variable choice in solving CSPs, where the choice does not influence the final outcome, but can make a difference in runtime.

Motivation
Transition systems and planning tasks
Plans

Transition systems and planning tasks

Motivation
Transition systems and planning tasks
Transition systems
Operators
Planning tasks
Plans

Transition systems with nondeterminism (cf. Chapter 2)

Definition (transition system)

A nondeterministic transition system is a 5-tuple $\mathcal{T} = \langle S, L, T, s_0, S_\star \rangle$ where

- S is a finite set of states,
- L is a finite set of (transition) labels,
- $T \subseteq S \times L \times S$ is the transition relation,
- $s_0 \in S$ is the initial state, and
- $S_\star \subseteq S$ is the set of goal states.

Note: $T \subseteq S \times L \times S$ allows nondeterministic operators with more than one possible outcome.

Nondeterministic operators

Definition (nondeterministic operator)

Let V be a set of finite-domain state variables. A nondeterministic operator in unary nondeterminism normal form with conjunctive precondition and unconditional effects, or nondeterministic operator for short, is a pair $o = \langle \chi, E \rangle$, where

- χ is a conjunction of atoms over V (the precondition), and
- $E = \{e_1, \dots, e_n\}$ is a finite set of possible effects of o, each e_i being a conjunction of atomic finite-domain effects over V.

Nondeterministic operators

Definition (nondeterministic operator application)

Let $o = \langle \chi, E \rangle$ be a nondeterministic operator and s a state.

Applicability of o in s is defined as in the deterministic case, i.e., o is applicable in s iff $s \models \chi$ and the change set of each effect $e \in E$ is consistent.

If o is applicable in s, then the application of o in s leads to one of the states in the set $app_o(s) := \{app_{\langle \chi, e \rangle}(s) \mid e \in E\}$ nondeterministically.

Nondeterministic operators

Example

$\textit{put-on-block}(A, B) = \langle \chi, \{e_1, e_2\} \rangle$ where

- $\chi = \{\textit{handempty} \mapsto \textit{false}, \textit{clear-B} \mapsto \textit{true}, \textit{pos-A} \mapsto \textit{hand}\}$,
- $e_1 = \{\textit{handempty} \mapsto \textit{true}, \textit{clear-B} \mapsto \textit{false}, \textit{pos-A} \mapsto \textit{on-B}\}$,
- $e_2 = \{\textit{handempty} \mapsto \textit{true}, \textit{pos-A} \mapsto \textit{table}\}$.

Applied to a state where the agent is holding block A and block B is clear, this operator leads to one of two possible successor states. Either A gets stacked on B successfully, or A is dropped to the table.

Nondeterministic planning task

Definition (nondeterministic planning task)

A (fully observable) nondeterministic planning task is a 4-tuple $\Pi = \langle V, I, O, \gamma \rangle$ where

- V is a finite set of finite-domain state variables,
- I is an initial state over V,
- O is a finite set of nondeterministic operators over V, and
- γ is a conjunctions of atoms over V describing the goal states.

Remark: In the following, we will always assume that our nondeterministic planning tasks are fully observable.

Mapping planning tasks to transition systems

Definition (induced transition system)

Every nondeterministic planning task $\Pi = \langle V, I, O, \gamma \rangle$ induces a corresponding nondeterministic transition system $\mathscr{T}(\Pi) = \langle S, L, T, s_0, S_\star \rangle$:

- S is the set of all states over V,
- L is the set of operators O,
- $T = \{\langle s, o, s' \rangle \mid s \in S,\ o \text{ applicable in } s,\ s' \in app_o(s)\}$,
- $s_0 = I$, and
- $S_\star = \{s \in S \mid s \models \gamma\}$

Plans

What is a plan?

In nondeterministic planning, plans are more complicated objects than in the deterministic case:

The best action to take may depend on nondeterministic effects of previous operators.

Nondeterministic plans thus often require branching. Sometimes, they even require looping.

What is a plan?

Example (Branching)

(Part of) a plan for winning the game Connect Four can be described as follows:

- Place a tile in the 4th column.
  - If opponent places a tile in the 1st, 4th or 7th column, place a tile in the 4th column.
  - If opponent places a tile in the 2nd or 5th column, place a tile in the 2nd column.
  - If opponent places a tile in the 3rd or 6th column, place a tile in the 6th column.

There is no non-branching plan that solves the task (= is guaranteed to win the game).

What is a plan?

Example (Looping)

A plan for building a card house can be described as follows:

1. Build a wall with two cards.
   If the structure falls apart, redo from start.
2. Build a second wall with two cards.
   If the structure falls apart, redo from start.
3. Build a ceiling on top of the walls with a fifth card.
   If the structure falls apart, redo from start.
4. Build a wall on top of the ceiling with two cards.
   If the structure falls apart, redo from start.

There is no non-looping plan that solves the task (unless the planning agent is very dextrous).

What is a plan?

- Plans should be allowed to branch. Otherwise, most interesting nondeterministic planning tasks cannot be solved.
- We may or may not allow plans to loop.
  - Non-looping plans are preferable because they guarantee that the goal is reached within a bounded number of steps.
  - Where non-looping plans are not possible, looping plans may be adequate because they at least guarantee that the goal will be reached eventually unless nature is unfair.

We will now introduce the formal concepts necessary to define branching and looping plans.

Nondeterministic plans: formal definition

Definition (strategy)

Let $\Pi = \langle V, I, O, \gamma \rangle$ be a nondeterministic planning task with state set S and goal states $S_\star$.

A strategy for Π is a function $\pi : S_\pi \to O$ for some subset $S_\pi \subseteq S$ such that for all states $s \in S_\pi$ the action $\pi(s)$ is applicable in s.

The set of states reachable in $\mathcal{T}(\Pi)$ starting in state s and following π is denoted by $S_\pi(s)$.

Nondeterministic plans: formal definition

Definition (weak, closed, proper, and acyclic strategies)

Let $\Pi = \langle V, I, O, \gamma \rangle$ be a nondeterministic planning task with state set S and goal states $S_\star$, and let π be a strategy for Π. Then π is called

- weak iff $S_\pi(s_0) \cap S_\star \neq \emptyset$,
- closed iff $S_\pi(s_0) \subseteq S_\pi \cup S_\star$,
- proper iff $S_\pi(s') \cap S_\star \neq \emptyset$ for all $s' \in S_\pi(s_0)$, and
- acyclic iff there is no state $s' \in S_\pi(s_0)$ such that s' is reachable from s' following π in a strictly positive number of steps.

Note: *Proper* implies *closed* and *acyclic* together with *closed* implies *proper*.

Nondeterministic plans: formal definition

- Strategies in nondeterministic planning correspond to applicable operator sequences in deterministic planning.
- In deterministic planning, a plan is an applicable operator sequence that results in a goal state.
- In nondeterministic planning, we define different notions of “resulting in a goal state”.

Nondeterministic plans: formal definition

Definition

Let $\Pi = \langle V, I, O, \gamma \rangle$ be a nondeterministic planning task with state set S and goal states $S_\star$.

- A strategy for Π is called a weak plan for Π iff it is weak.
- A strategy for Π is called a strong cyclic plan for Π iff it is proper.
- A strong cyclic plan for Π is called a strong plan for Π iff it is acyclic.

Summary and outlook

We extended the deterministic (classical) planning formalism:

- operators can be nondeterministic

Remark: We could also introduce nondeterminism in the initial situation by allowing more than one initial state, but this can be easily compiled into our formalism. (How?)

As a consequence, plans can contain

- branches and
- loops.

In the following chapter, we consider the strong planning problem and the strong cyclic planning problem and discuss some algorithms.