Planning

Chapter 11
Russell and Norvig

Planning

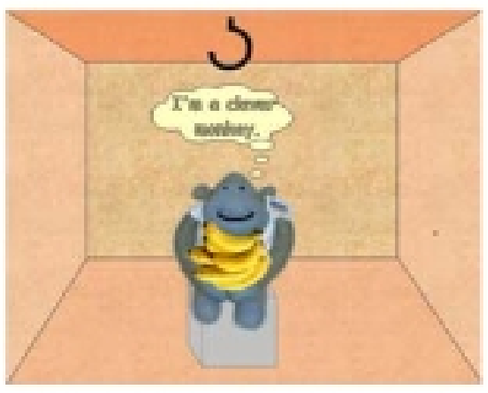
Monkey & Bananas

- A hungry monkey is in a room. Suspended from the roof, just out of his reach, is a bunch of bananas. In the corner of the room is a box. The monkey desperately wants the bananas but he can't reach them. What shall he do?

Planning

Monkey & Bananas (2)

- After several unsuccessful attempts to reach the bananas, the monkey walks to the box, pushes it under the bananas, climbs on the box, picks the bananas and eats them.



- The hungry monkey is now a happy monkey.

Planning

Planning

- To solve this problem the monkey needed to devise a plan, *a sequence of actions that would allow him to reach the desired goal.*
- Planning is a topic of traditional interest in Artificial Intelligence as it is an important part of many different AI applications, such as robotics and intelligent agents.
- To be able to plan, a system needs to be able to reason about the individual and cumulative effects of a series of actions. This is a skill that is only observed in a few animal species and only mastered by humans.
- The planning problems we will be discussing today are mostly Toy-World problems but they can be scaled up to real-world problems such as a robot cruising a space.

Planning

Planning vs. Problem Solving

- Planning is a special case of Problem Solving (Search).
- Problem solving searches a *state-space* of possible *actions*, starting from an *initial state* and following any path that it believes will lead it the *goal state*. Recording the actions along the path from the initial state to the goal state gives you a plan.
- Planning has two distinct features:
  1. The planner does not have to solve the problem in order (from initial to goal state). It can suggest actions to solve any sub-goals at anytime.
  2. Planners assume that most parts of the world are independent so they can be stripped apart and solved individually (turning the problem into practically sized chunks).

Planning

Planning using STRIPS

- The "classical" approach most planners use today is derived from the STRIPS language.
- STRIPS was devised by SRI in the early 1970s to control a robot called Shakey.
- Shakey's task was to negotiate a series of rooms, move boxes, and grab objects.
- The STRIPS language was used to derive plans that would control Shakey's movements so that he could achieve his goals.
- The STRIPS language is very simple but expressive language that lends itself to efficient planning algorithms.
- The representation we will use in Prolog is derived from the original STRIPS representation.

Planning

STRIPS Representation

- Planning can be considered as a logical inference problem:
  - a plan is inferred from facts and logical relationships.
- STRIPS represented planning problems as a series of *state descriptions* and *operators* expressed in first-order predicate logic.

State descriptions represent the state of the world at three points during the plan:

- *Initial state*, the state of the world at the start of the problem;
- *Current state*, and
- *Goal state*, the state of the world we want to get to.

Operators are actions that can be applied to change the state of the world.

- Each operator has *outcomes* i.e. how it affects the world.
- Each operator can only be applied in certain circumstances. These are the *preconditions* of the operator.

Planning

Planning in Prolog

- As STRIPS uses a logic based representation of states it lends itself well to being implemented in Prolog.
- To show the development of a planning system we will implement the Monkey and Bananas problem in Prolog using STRIPS.
- When beginning to produce a planner there are certain *representation considerations* that need to be made:
  - How do we represent the state of the world?
  - How do we represent operators?
  - Does our representation make it easy to:
    - check preconditions;
    - alter the state of the world after performing actions; and
    - recognise the goal state?

Planning

Representing the World

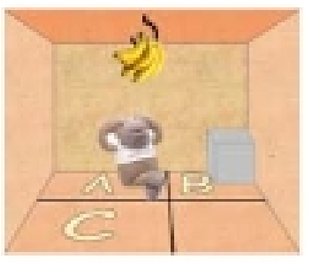
- In the Monkey&Banana problem we have:
  - *objects*: a monkey, a box, the bananas, and a floor.
  - *locations*: we'll call them a, b, and c.
  - *relations of objects to locations*. For example:
    - the monkey is at location a;
    - the monkey is on the floor;
    - the bananas are hanging;
    - the box is in the same location as the bananas.
- To represent these relations we need to choose appropriate predicates and arguments:
    - at(monkey,a).
    - on(monkey,floor).
    - status(bananas,hanging).
    - at(box,X), at(bananas,X).

Planning

Initial and Goal State

- Once we have decided on appropriate state predicates we need to represent the Initial and Goal states.
- Initial State:

```
on(monkey, floor),
on(box, floor),
at(monkey, a),
at(box, b),
at(bananas, c),
status(bananas, hanging).
```



- Goal State:

```
on(monkey, box),
on(box, floor),
at(monkey, c),
at(box, c),
at(bananas, c),
status(bananas, grabbed).
```

- Only this last state can be known without knowing the details of the Plan (i.e. how we're going to get there).

Planning

Representing Operators

- STRIPS operators are defined as:
  - **NAME**: How we refer to the operator e.g. go(Agent, From, To).
  - **PRECONDITIONS**: What states need to hold for the operator to be applied. e.g. [at(Agent, From)].
  - **ADD LIST**: What new states are added to the world as a result of applying the operator e.g. [at(Agent, To)].
  - **DELETE LIST**: What old states are removed from the world as a result of applying the operator. e.g. [at(Agent, From)].
- We will specify operators within a Prolog predicate `opn/4`:

```
opn( go(Agent,From,To),          <-- Name
     [at(Agent, From)],          <-- Preconditions
     [at(Agent, To)],            <-- Add List
     [at(Agent, From)] ).        <-- Delete List
```

Planning

The Frame Problem

- When representing operators we make the assumption that the only effects our operator has on the world are those specified by the add and delete lists.
- In real-world planning this is a hard assumption to make as we can never be absolutely certain of the extent of the effects of an action.
  - This is known in AI as the *Frame Problem*.
- Real-World systems, such as Shakey, are notoriously difficult to plan for because of this problem. Plans must constantly adapt based on incoming sensory information about the new state of the world otherwise the operator preconditions will no longer apply.
- The planning domains we will be working in our Toy-Worlds so we can assume that our framing assumptions are accurate.

Planning