AI Session 2

Jan 22, 2021

Agenda

- Recap
- Situation
- Search-based problem-solving

Recap

- Grading based on XP (weekly exercises)
- AI
  - thinking/acting like humans / rationally (?)
  - A wide subfield of CS, several applications
- Intelligent agents
  - act in their (task) environment based on perceptions “intelligently”
  - there is a wide array of different environments and ways to classify them

Situation

- Exercises: nicely done!
- Situation concerning project topics?

Solving Problems

- Simple reflex agents map a perception to an action.
- Often, a more sophisticated (?) agent has some goal(s) that may not be reachable by a single action. How to act “intelligently,” then? Some future considerations and planning are required!
    - Logical deduction, reasoning – left to a future session
    - Planning a sequence of actions by *searching*
        - “blindly”
        - with some information
    - when an agent has a plan, it can proceed action by action.
        - generally, replanning may be required along the way

Uninformed search

Defining Problems Suitably

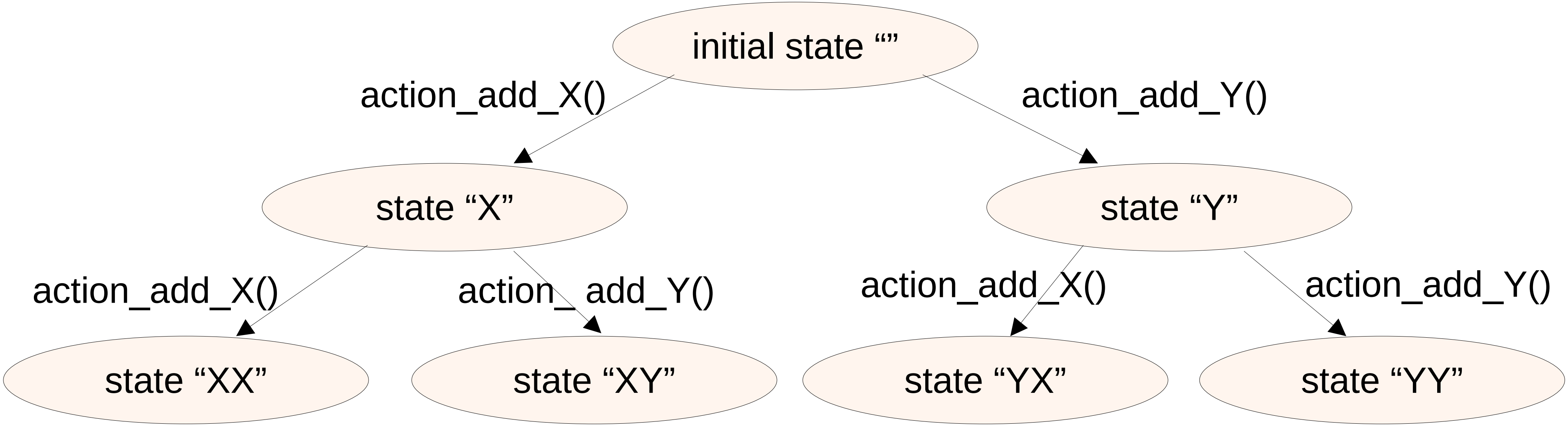
- A problem may often be considered as a searching problem with elements like
  - **states** containing **initial state** and **goal state**
  - **successor functions** returning, based on a state, available actions and resulting states
  - **test of** a **goal** / the goal having been reached
  - **step costs**, **path costs**
- Environment affects the problem formulation
  - let us consider here an “easy” case, in which the environment is deterministic and observable

Understanding the Limitations

- All the algorithms are not guaranteed to find a solution (even when there is such)
- Setting goals and selecting algorithms reasonably is important
  - Often, “good enough” results and approximations must suffice in practice, since many problems are, actually, computationally very demanding (impossible to solve exactly)

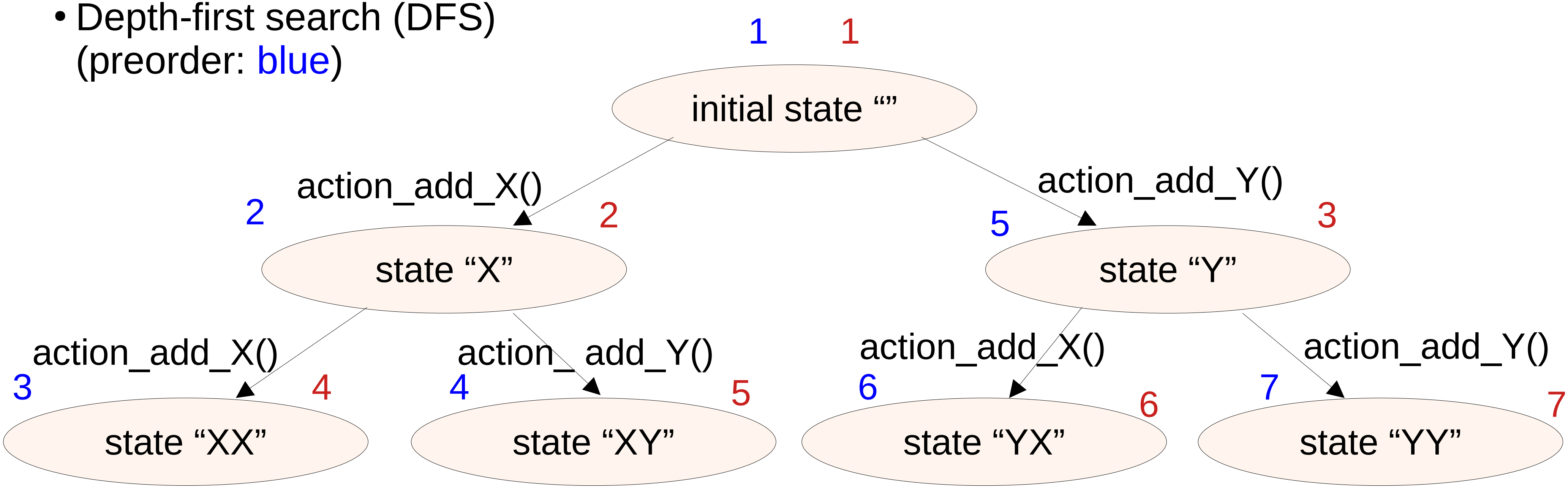
Search Tree and Tree Search

- Often useful way to consider the state space
  - defined by the initial state and the successor function
  - tree search can be used to solve problems
  - Search strategy: in which order should the nodes be expanded?



Blind (Uninformed) Search Strategies

- Breadth-first search (BFS) (search order: red)
- Depth-first search (DFS) (preorder: blue)



Informed Search

- If we have some additional problem-specific information, we may get results more efficiently than with blindly generating successor states and testing if a goal has been found.
  - We may be able to use some heuristic and test good-looking paths early.

Heuristic (Informed) Search Strategies

Seeing some nodes as more promising than others

- Greedy best-first search
  - graph search using a heuristic function estimating “distance to goal”
  - picking always a node with minimum heuristic value to be expanded next
- A*
  - a node is evaluated based on the combination of the actual path cost to the node and the value of the heuristic function
  - with a heuristic function satisfying certain requirements, guaranteed to be optimal and complete
  - Used heavily (as different variants), e.g., in games (and everywhere) for finding paths/routes

Local Search / Optimization

- Sometimes, the actual path leading to the goal is not so interesting (not needed as a solution to the problem).
  - finding designs or layouts of high quality
  - creating schedules
  - routing vehicles
  - optimizing configurations
- Local methods may cope quite well even with infinite state spaces and without high memory needs

Local Search / Optimization

- E.g.,
  - Hill-climbing (greedy local search)
    - always proceed to the neighbor state of highest (lowest) value and stop, when no better neighbors
    - gets easily stuck in global maxima
    - variants like stochastic hill-climbing, first-choise hill-climbing, random-restart hill-climbing...
  - Genetic algorithms

Exercises

It is time to embark upon the challenges of this week (check Moodle)!