Haskell Tips
You can turn any function that takes two inputs into an infix operator:

\[ \text{mod } 7 \ 3 \ \text{is the same as } \ 7 \ `\text{mod}` \ 3 \]

takeWhile returns all initial elements of a given list, as long as a given condition is met, and then it stops:

\[ \text{takeWhile even [2, 6, 4, 7, 2, 8]} \]

Search in State Spaces

- Many problems in Artificial Intelligence can be mapped onto searches in particular state spaces.
- This concept is especially useful if the system (our “world”) can be defined as having a finite number of states, including an initial state and one or more goal states.
- Optimally, there are a finite number of actions that we can take, and there are well-defined state transitions that only depend on our current state and current action.

Search in State Spaces

- To some extent, it is also possible to account for state changes that the algorithm itself does not initiate.
- For example, a chess playing program can consider its opponent’s future moves.
- However, it is necessary that the set of such actions and their consequences are well-defined.
- While computers are able to play chess at a very high level, it is impossible these days to build a robot that, for instance, is capable of reliably carrying out everyday tasks such as going to a supermarket to buy groceries.

Search in State Spaces

Let us consider an easy task in a very simple world with our robot being the only actor in it:

- The world contains a floor and three toy blocks labeled A, B, and C.
- The robot can move a block (with no other block on top of it) onto the floor or on top of another block.
- These actions are modeled by instances of a schema, move(x, y).
- Instances of the schema are called operators.

Search in State Spaces

- The robot’s task is to stack the toy blocks so that A is on top of B, B is on top of C, and C is on the floor.
- For us it is clear what steps have to be taken to solve the task.
- The robot has to use its world model to find a solution.
- Let us take a look at the effects that the robot’s actions exert on its world.
Search in State Spaces

- In order to solve the task efficiently, the robot should “look ahead”, that is, simulate possible actions and their outcomes.
- Then, the robot can carry out a sequence of actions that, according to the robot’s prediction, solves the problem.
- A useful structure for such a simulation of alternative sequences of action is a directed graph.
- Such a graph is called a state-space graph.

State-Space Graphs

- To solve a particular problem, the robot has to find a path in the graph from a start node (representing the initial state) to a goal node (representing a goal state).
- The resulting path indicates a sequence of actions that solves the problem.
- The sequence of operators along a path to a goal is called a plan.
- Searching for such a sequence is called planning.
- Predicting a sequence of world states from a sequence of actions is called projecting.

Decision Trees

A decision tree is a special case of a state-space graph.

It is a rooted tree in which each internal node corresponds to a decision, with a subtree at these nodes for each possible outcome of the decision.

Decision trees can be used to model problems in which a series of decisions leads to a solution.

The possible solutions of the problem correspond to the paths from the root to the leaves of the decision tree.
Decision Trees

Example: The n-queens problem
How can we place n queens on an n x n chessboard so that no two queens can capture each other?

A queen can move any number of squares horizontally, vertically, and diagonally.
Here, the possible target squares of the queen Q are marked with an x.

September 18, 2018 Introduction to Artificial Intelligence Lecture 5: Search in State Spaces I

Backtracking in Decision Trees

Idea: Start at the root of the decision tree and move downwards, that is, make a sequence of decisions, until you either reach a solution or you enter a situation from where no solution can be reached by any further sequence of decisions.

In the latter case, backtrack to the parent of the current node and take a different path downwards from there. If all paths from this node have already been explored, backtrack to its parent.

Continue this procedure until you find a solution or establish that no solution exists (there are no more paths to try out).

September 18, 2018 Introduction to Artificial Intelligence Lecture 5: Search in State Spaces I